Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

On the atout ticket learning problem for neural networks and its application in securing federated learning exchanges

Abdelhak Bouayad[a,*], Mohammed Akallouch[a], Abdelkader El Mahdaouy[a], Hamza Alami[b], Ismail Berrada[a]

[a] College of Computing, Mohammed VI Polytechnic University, Benguerir, Morocco
[b] Faculty of Sciences Dhar EL Mahraz, Sidi Mohamed Ben Abdellah University, Morocco

ARTICLE INFO

Keywords:
Federated learning
Deep learning
Privacy-preserving machine learning
Trusted execution environments

ABSTRACT

Artificial Neural Networks (ANNs) have become the backbone of many real-world applications, including distributed applications relying on Federated Learning (FL). However, several vulnerabilities/attacks have emerged in recent years, affecting the benefits of using ANNs in FL, such as reconstruction attacks and membership inference attacks. These attacks can have severe impacts on both the societal and professional levels. For instance, inferring the presence of a patient's private health record in a medical study or a clinic database violates the patient's privacy and can have legal or ethical consequences. Therefore, protecting the data and model from malicious attacks in FL systems is important. This paper introduces the Atout Ticket Learning (ATL) problem. This new problem consists of identifying sensitive parameters (atout tickets) of a neural network model, which, if modified, will increase the model's loss by at least a given threshold ϵ. First, we formulate ATL as an ℓ_0-norm minimization problem, and we derive a lower bound on the number of atout tickets needed to achieve a model degradation of ϵ. Second, we design the Atout Ticket Protocol (ATP) as an effective solution for privacy-preserving in FL systems using atout tickets, along with the benefit of noise perturbations and secure aggregation techniques. Finally, we experiment ATP against FL reconstruction attacks using new selection strategies, namely Inverting Gradients, Deep Leakage, and Improved Deep Leakage. The results show that ATP is highly robust against these attacks.

1. Introduction

The previous few years have shown an amazing increase withinside the use of Artificial Neural Networks (ANNs) (see Fig. 1). ANN models have been successfully applied in several domains and today have become a powerful tool in Machine Learning (ML) [1,2].

An ANN, often referred to as a "neural network", is a category of machine learning models inspired by biological neural circuits. At its core, a neural network consists of interconnected neurons. Each neuron receives inputs that are multiplied by adjustable weights and summed with a bias term. An activation function then determines the neuron's output. Neural networks process input values through layers: an "input layer" receives initial data, one or more "hidden layers" apply functions to these values, and an "output layer" produces a result, like a prediction. The network's parameters are iteratively adjusted to improve pattern recognition. In deep learning models, weights are crucial parameters that modify input data within network layers. During training, weights are updated based on input data, following a process consisting of initialization, forward propagation, backpropagation, and weight adjustment.

The goal of weight adjustment is to find an optimal set that allows the model to accurately map inputs to outputs, capturing underlying data patterns. Trained models use these weights to make predictions on new data, with prediction quality influenced by the efficacy of weight adjustments during training.

ANNs are known to be data-hungry in the sense that the input data size for training sizeable models grows exponentially with the number of parameters. This can pose significant challenges in terms of data availability and privacy. Traditional centralized ML approaches involve collecting all data in a single location, which can lead to privacy concerns and compliance issues, especially with sensitive data such as medical records. Distributed ML [3] handles this data issue by distributing the machine learning workload (data and models) across multiple machines, making it the most widely adopted and deployed ML technology in industrial production. Among the various distributed

* Corresponding author.
E-mail addresses: abdelhak.bouayad@um6p.ma (A. Bouayad), mohammed.akallouch@usmba.ac.ma (M. Akallouch), abdelkader.mahdaouy@um6p.ma (A. El Mahdaouy), hamza.alami@um6p.ma (H. Alami), ismail.berrada@um6p.ma (I. Berrada).

https://doi.org/10.1016/j.jisa.2024.103891

Available online 9 October 2024

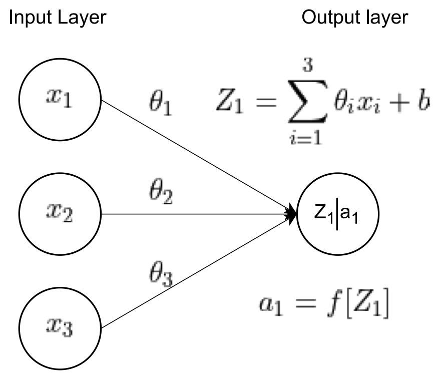


Fig. 1. Structure of a single neuron in a neural network.

ML approaches, FL has gained prominence. FL allows multiple parties to collaboratively train a model without sharing their raw data, thus enhancing privacy and security. Fundamentally, in the data parallelism approach, data is partitioned across worker nodes that apply the same algorithm to different data chunks, whereas in the model parallelism approach, worker nodes operate on different parts of the ML model.

In terms of privacy-preserving, FL could be an efficient distributed ML framework for securing users' private data. In fact, considering the raw data being the sensitive part of the system, FL can provide an efficient solution for preserving users' privacy by ensuring that exchanged model parameters during the training phase do not leak any information about users' data. However, many recent works have shown that distributed ML systems, and in particular FL systems, present many privacy issues and vulnerabilities to a variety of attacks such as reconstruction attacks [4–7], feature information attacks [8,9], and membership inference attacks [10–16].

To overcome these issues, several solutions have been proposed, such as Secure Aggregation (SA) [17] and Differential Privacy (DP) [18]. The main intuitive ideas behind these approaches are either to hide all model parameters or to alter all these parameters in order to prevent attackers from exploiting them. In the case of SA approaches, the overhead of these protocols creates a crucial bottleneck in scaling secure FL to a considerable number of users [19]. Especially in a network of N users, a well-optimized protocol for SA generally requires pairwise random masks to be generated among each pair of users. Consequently, the overhead grows quadratically with the number of users (i.e. $O(N^2)$). This first class of approaches also greatly impacts the processing time as they typically require cryptography-based solutions, which can be a limiting factor for resource-constrained devices. On the other hand, DP solves the security problem by altering the model parameters, which causes a trade-off between convergence performance and privacy protection (adding noises). This second class of approaches greatly impacts the model performance as the accuracy drops when the effectiveness of the approach increases. In conclusion, the proposed approaches either give up utility to gain privacy or give up privacy to gain privacy.

This paper aims to investigate approaches that can leverage the advantages of both the model's perturbation and SA while avoiding the negative impact of DP on the model performance and reducing the computation overhead that occurs by encrypting client models. A trivial idea consists of altering and encrypting only a small fraction of model parameters during exchanges. This leads us to address the following non-trivial research questions: (**RQ1**) Given a trained neural network, can we identify a subset of sensitive parameters of the network that should be altered to decrease the model's performance by a given threshold? If yes (**RQ2**), What is the minimum size of such a subset? and (**RQ3**) How to leverage this subset of sensitive parameters to design efficient protocols for securing FL exchanges? Note that the two first questions are fundamental, and answering them may have applications beyond FL settings.

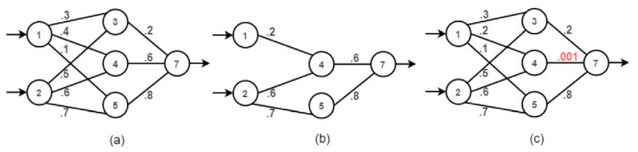


Fig. 2. Sparse Neural Network Learning vs. Atout Ticket Learning: (a) Initial Network, (b) SNNL, and (c) ATL.

In this paper, we provide positive answers to the previous questions. Firstly, we introduce the Atout Ticket Learning (ATL) problem, formulated as a sparsity-constrained optimization problem, which aims to identify a subset of sensitive parameters of a neural network model. Modifying/altering these sensitive parameters, hereinafter called Atout Tickets[1] (AT), will increase the expected loss at least by a threshold ϵ. The notion of the threshold ϵ is employed here to quantify the sensitivity of model parameters and the expected degradation resulting from the alteration of its corresponding subset of AT (application-dependent). Secondly, we derive a theoretical lower-bound on the number of AT, expressed in terms of the threshold ϵ and the model's parameters magnitudes. Since ATL is an NP-hard problem, the obtained lower-bound can be leveraged to infer and design heuristics for identifying AT. Finally, we introduce the Atout Ticket Protocol (ATP) as a privacy-preserving protocol for securing users' data during FL exchanges. ATP relies on both encryption and alteration of an extremely low number of AT to secure clients' private data. The aggregation of client models' AT is performed in a Trusted Entity (TE). The central server aggregates the non-sensitive parameters of the client's models as well as the altered AT and the averaged parameters by the TE.

1.1. SNNL vs. ATP

The scope of this paper lies at the borders of two hot areas of machine learning: (1) Privacy-Preserving FL [20], which explores how to protect the data privacy and to secure ML models and (2) Sparse Neural Network Learning (SNNL) [21] where the objective is to identify a small subset of the available parameters while preserving the same level of accuracy.

SNNL and ATL problems have fundamentally different objectives. In fact, in SNNL, the objective is to find a sub-network that will replace the initial network while preserving the model accuracy whereas the initial network is preserved in ATL and only a subset of model parameters are modified to reach a model alteration that fit the threshold ϵ. Although the SNNL and ATL problems are two different optimization problem, the approaches to solve both are related to the ℓ_0 minimization problem which is generally NP-hard, even for the simple convex objective function. In the case of ℓ_0-norm minimization, the proposed algorithms in the literature can be categorized into four classes [22]: (i) greedy descent methods, (ii) convex approximate methods, (iii) non-convex approximate methods, and (iv) exact methods. In fact, a tangible progress has been made in the literature, leading to several SNNL techniques: (i) model downsizing [23,24], (ii) low-rank matrix factorization [25,26], (iii) compression [27,28], (iv) parameter sharing [29], (v) quantization [27,30], and (vi) pruning (sparsification) [31]. Pruning-based methods have been an active area of research over the last three decades with more than 300 publications [32–43], to name a few. Through a regression toy example with a linear activation function, Fig. 2 highlights these differences. The initial network is given in

[1] In the tarot deck, the atout (meaning "assets") cards serve as trump cards (21 trump cards plus the fool).

Table 1
The output values of networks f_a, f_b, and f_c.

Input	f_a	f_b	f_c
$\begin{pmatrix}1\\0\end{pmatrix}$	0.38	0.12	0.1404
$\begin{pmatrix}0\\1\end{pmatrix}$	1.02	0.92	0.6606
$\begin{pmatrix}1\\1\end{pmatrix}$	1.4	1.04	0.801

Fig. 2(a), the SNNL in Fig. 2(b) and the ATL in Fig. 2(c), hereinafter referred as f_a, f_b and f_c respectively. As we can see in f_b, neuron 3 is dropped as well as the connection between neurons 1 and 5 from f_a. The ATL network f_c is exactly the same as f_a except for the atout ticket 6 corresponding to the parameter of the connection between neurons 4 and 7, which haven been replaced by the value 0.001 (other values are possible) in contrast to deferential privacy (DP) which adds noises to all model parameters. Table 1 summarizes the network outputs for three given inputs. We notice that changing one atout ticket leads to a high degradation of the output which is in general bigger than the f_b network. From a privacy-preserving perspective in distributed ML, this atout ticket should be hidden to the network (encrypted) for the resilience to ML attacks.

1.2. Paper contributions

The key contributions of the paper are summarized as follows:

- Introduction of the Atout Ticket Learning (ATL) problem. While ATL is a recognized challenge, our work uniquely adapts and applies it within the FL environment to enhance security.
- Theoretical Insights: We provide a theoretical lower bound on the number of atout tickets needed to degrade the model performance by a specified threshold ϵ.
- Development of the Atout Ticket Protocol (ATP): We introduce ATP as a novel protocol designed to secure FL exchanges, incorporating atout tickets, noise perturbations, and secure aggregation techniques.
- Comprehensive evaluation: We conduct extensive experiments to evaluate the resilience of our proposed framework against various FL reconstruction attacks, including Inverting Gradients, Deep Leakage, and Improved Deep Leakage

To the best of our knowledge, the ATL problem and its application for securing FL exchanges are first time investigated in this paper.

The rest of the paper is organized as follows. Section 2 recalls notations and concepts used in the paper. Section 3 introduces the ATL problem and the main theoretical results. Section 4 highlights the Atout Ticket Protocol as one an application of ATL in securing FL exchanges. Section 5 is dedicated to experimentation. Section 6 presents the related work. Finally, the last section concludes the paper and draws future perspectives.

2. Background

In this section, without loss of generality, we consider the class of Feed Forward Neural Networks (FFNNs). A FFNN defines a set of N_θ neurons with synapse parameters θ that are ranged over a series of L+2 layers $\{l_0, l_1, \dots, l_L, l_{L+1}\}$ such that l_0 is the input layer, $\{l_1, \dots, l_L\}$ are the hidden layers and l_{L+1} is the output layer. N_θ^l is the number of neurons in layer l, with $N_\theta = \sum_{l=0}^{L+1} N_\theta^l$. N_θ^0 corresponds to d for layer l_0 (the dimension of the input vector $X \in \mathbb{R}^d$), while we assume that layer l_{L+1} has exactly one neuron ($N_\theta^{L+1} = 1$).

Formally, we define a FFNN as a mapping $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}$ parameterized by a tensor $\theta \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_{L+1}}$. We also represent θ as a set of tensors $\{\theta^{l_1}, \theta^{l_2}, \dots, \theta^{l_{L+1}}\}$, where $\theta^l \in \mathbb{R}^{N_\theta^l \times N_\theta^{l-1}}$ is the layer parameters of synapses connecting layer $l-1$ neurons to layer l neurons.

As $N_\theta^{L+1} = 1$ for the output layer, then θ^{L+1} will be simply refered using vector notations: $\theta^{L+1} = (\theta_i^{L+1})$. Finally, we define the ℓ_0, ℓ_p, and ℓ_∞-norms of tensors as usual ($|\,|$ is the absolute value):

$$\|\theta\|_0 = \mathbf{card}\big(\{\theta_{ji}^l \mid \theta_{ji}^l \neq 0\}\big), \quad \|\theta\|_\infty = \mathbf{max}\big(\big|\theta_{ji}^l\big|\big),$$

$$\text{and } \|\theta\|_p = \left(\sum_{l=1}^{L+1} \sum_{j=1}^{N_\theta^l} \sum_{i=1}^{N_\theta^{l-1}} \left|\theta_{ji}^l\right|^p\right)^{1/p}$$

For neural network computations, given an input $X = (x_j) \in \mathbb{R}^d$, the output of a neuron j of the input layer l_0 corresponds to the input x_j ($y_j^{(0)} = x_j$), while this output $y_j^{(l)}$ at layer $l \geq 1$ is:

$$y_j^{(l)} = \varphi(S_j^{(l)}) \quad \text{with} \quad S_j^{(l)} = \sum_{i=1}^{N_\theta^{l-1}} \theta_{ji}^l y_i^{(l-1)} \tag{1}$$

$S_j^{(l)}$ corresponds to the linear operation of j while φ represents the non-linear activation of the neuron. We assume that φ is a K_θ-Lipschitz function (K_θ is the Lipschitz constant). As the output layer l_{L+1} has only one neuron, then the neural computation of f_θ can be written as follows:

$$f_\theta(X) = y_1^{(L+1)} = \sum_{j=1}^{N_\theta^L} \theta_j^{L+1} y_j^{(L)}(X) \tag{2}$$

2.1. Recovery problem

The recovery problem [44] of target function $f : \mathbb{R}^d \mapsto \mathbb{R}$, consists in estimating an approximation of f by a neural network. Given an $\epsilon > 0$, f_θ is said to be an ϵ-approximation of f if for all $X \in \mathbb{R}^d$:

$$|f(X) - f_\theta(X)| \leq \epsilon \tag{3}$$

Under some assumptions, it is well known that FFNNs are universal approximators, i.e. for any function f, and any $\epsilon > 0$, we can find f_θ, a neural ϵ-approximation of f. The recovery problem is in principle a nonlinear problem, and commonly, we would like to recover f having some priori information about it. This is generally done using a dataset $\mathcal{D}$ corresponding to a finite set of data points $\{(X_1, Y_1), \dots, (X_m, Y_m)\}$. Samples of $\mathcal{D}$ are assumed to be drawn in an IID (Independent and Identically Distributed) fashion from a probability law which is generally unknown. In order to define the best available approximation f_θ of f, a loss function $\mathcal{L}$ is used to measure the loss/discrepancy between f and f_θ on the dataset $\mathcal{D}$. In this case, the empirical risk $\mathcal{R}(f_\theta)$ of f_θ corresponds to the expectation $\mathbb{E}_\mathcal{D}$ of the measured losses over $\mathcal{D}$, and it is defined as follows:

$$\mathcal{R}(f_\theta) = \mathbb{E}_\mathcal{D}[\mathcal{L}(f_\theta(X), Y)] = \frac{1}{|\mathcal{D}|} \sum_{(X,Y)\in\mathcal{D}} \mathcal{L}(f_\theta(X), Y)$$

For the rest of the paper, we assume a given FFNN f_θ approximating a target function f, such that the parameter tensor $\theta \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_{L+1}}$ is known in advance (FFNN is already trained).

2.2. Atout masks

In order to identify sensitive parameters (aka atout tickets) of θ, we define the atout mask as a tensor $\mathcal{M} \in \mathbb{T}^{l_1 \times l_2 \times \cdots \times l_{L+1}}$ such that $\mathbb{T}$ is a subset of $\mathbb{R}$. Similarly to θ, we represent $\mathcal{M}$ as a set of tensors $\{\mathcal{M}^{l_1}, \mathcal{M}^{l_2}, \dots, \mathcal{M}^{l_{L+1}}\}$. The intuitive idea is that zero values of $\mathcal{M}_{ji}^l$ highlight parameters of θ that should be skipped, i.e, corresponding to potential non-sensitive parameters (non-atout tickets). $\mathcal{M}_\mathbb{T}(\theta)$ will denote the set of all atout masks $\mathcal{M}$ of θ.

Applying a mask $\mathcal{M}$ to θ will result on the tensor $\tilde{\theta} = \theta \oplus \mathcal{M}$, where $\oplus$ is the tensor sum operator ($\tilde{\theta}_{ji}^l = \theta_{ji}^l + \mathcal{M}_{ji}^l$). In this case, $f_{\tilde{\theta}}$ is also a FFNN with same structure (graph) of f_θ, as the only differences are on the synapse parameters. Finally, the class of all FFNNs derived from f_θ corresponds to the set $\mathcal{F}(f_\theta) = \{f_{\theta\oplus\mathcal{M}} \mid \mathcal{M} \in \mathcal{M}_{\mathbb{T}}(\theta)\}$.

The definition of the atout masks is quite general in the sense that it can handle different parameter configurations. In fact, for some given parameters:

- Parameter inhibitions (e.g. dropouts) are simply modeled by a mask $\mathcal{M}^l_{ji} = -\theta^l_{ji}$.
- Additive noises are modeled by a mask $\mathcal{M}^l_{ji} = \epsilon^l_{ji}$ such that $\epsilon^l_{ji} \sim \mathcal{N}(0, N)$.
- Multiplicative noises are modeled by a mask $\mathcal{M}^l_{ji} = \theta^l_{ji} \times (\epsilon^l_{ji} - 1)$ such that $\epsilon^l_{ji} \sim \mathcal{N}(0, N)$.

2.3. Federated leaning

FL is a novel machine learning paradigm that enables multiple participants to collaboratively train a global model without exposing their private data [45]. This approach preserves data privacy by allowing each participant to keep their data locally and only share the model's updates with a centralized server. The server aggregates the updates and distributes the updated model to the participants, who continue to train it on their local data. This process repeats until the model converges or satisfies a predefined criterion. FL has potential applications in various domains where data is distributed, sensitive, or large-scale, such as healthcare, finance, education, and social media. It offers a promising solution for collaborative machine learning in the era of big data and privacy concerns. However, FL also poses some challenges that need to be addressed, such as dealing with heterogeneous data, reducing communication overhead, ensuring robustness against malicious attacks, and achieving fairness among participants.

2.4. Secure aggregation

In the domain of Federated Learning, Secure Aggregation (SA) protocols are devised to safeguard the aggregation process of locally trained models or updates from participating clients. The primary objective is to thwart any attempts by unauthorized entities or malicious actors to intercept individual client models or data during the aggregation process. Typically, these protocols employ cryptographic methodologies like homomorphic encryption or secure multi-party computation. These techniques facilitate the confidential and privacy-preserving aggregation of client models, ensuring that the contributions of each client remain confidential and inaccessible to unauthorized parties

2.5. Differential privacy

Differential Privacy (DP) [46] is a mathematical method designed to quantify and minimize the risk of identifying individual data points within a dataset. It achieves this by introducing random noise into the data or computations, ensuring that the output remains useful while obscuring the contribution of any single data point. In the context of FL [45], DP techniques can be applied to the updates sent by clients to the central server, ensuring that the aggregated updates do not disclose sensitive information about individual data points.

2.6. Secure entities

A secure entity (or trusted execution environment) is a computing environment that isolates code and data from the operating system using hardware-based isolation or a trusted hypervisor. This ensures that even users with physical or root access cannot access or tamper with the enclave's memory or execution.

There are different implementations of secure entities for various devices and data centers, such as Intel SGX [47], AMD SEV [48], and AWS Nitro Enclaves. These implementations have different tradeoffs between security and convenience, and may suit various applications.

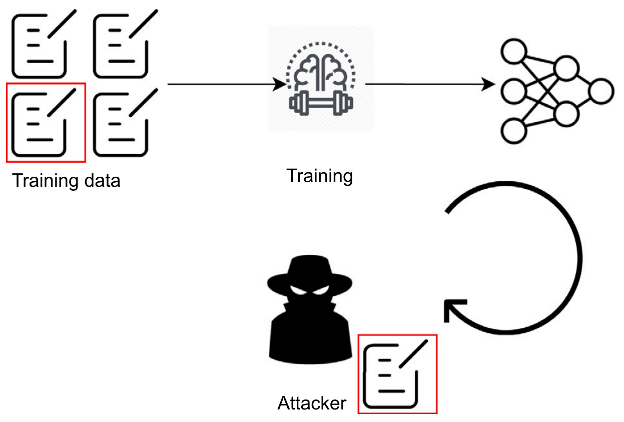


Fig. 3. Reconstruction attack.

Intel SGX provides strong security guarantees by using remote attestation to verify that a known code is running within the enclave. It requires developer SDKs to build enclave-compatible applications.

AMD SEV provides convenience by encrypting the whole memory of a virtual machine(VM), without requiring developer SDKs. However, it has a larger attack surface and needs extra precautions to protect against threats.

AWS Nitro Enclaves are based on the Nitro System, which has a lightweight hypervisor and offloads most functionality to Nitro Cards. These enclaves run as isolated VMs alongside an existing EC2 instance, and can only be accessed by an application in the same instance.

All major cloud providers offer secure enclave services, using either AMD or Intel-based processors, or their platforms

2.7. Privacy threats on machine learning

To extract training data, attacks infer information through the weights of the ANNs, and an attacker can employ an optimization algorithm capable of acquiring the training inputs. To execute the attack, the attacker initially generates a pair of "dummy" inputs and labels randomly and subsequently conducts the usual forward and backward procedures. Subsequently, the attacker utilizes the acquired dummy weights from the dummy data. Instead of optimizing model weights as in typical training, the attacker optimizes the dummy inputs and labels to minimize the distance between dummy weights and real weights. Matching the weights renders the dummy data proximate to the original dataset. Upon the completion of optimization, the private training data, encompassing both inputs and labels, becomes fully exposed.

2.7.1. Reconstruction attacks

Reconstruction attacks (Fig. 3) are a type of attack that aims to reconstruct the data that was used to train an ML model. These attacks can be performed in two ways: white box and black box. In a white box attack, the attacker has access to the training parameters of the model and uses them to infer the original data. In a black box attack, the attacker does not have access to the training parameters but queries the model with inputs and uses the outputs to infer the original data.

2.7.2. Membership inference attack (MIA)

The membership inference attack, first introduced by Shokri et al. [10], focuses on distinguishing the presence or absence of a specific data record within a dataset 4. In this attack scenario, assuming the adversary has access to data like that used for training the target model, they employed this data to train a'shadow' model that mimics the target and an attack model dedicated to deducing membership. The attack model predicts whether the target model has encountered

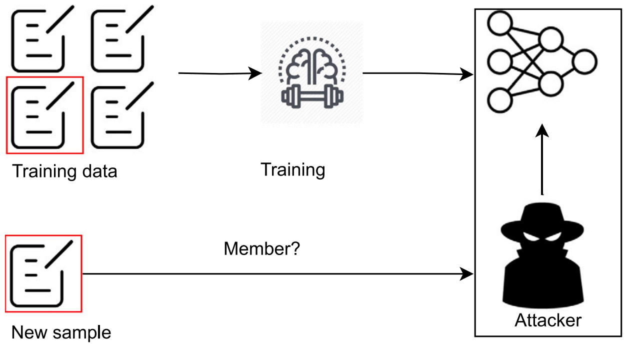


Fig. 4. Membership inference attack.

a given example during training based on the output probabilities it generates. Shokri et al. [10] executed this attack in a black-box setting against models trained in the cloud using Google Prediction API and Amazon ML, revealing that the attack can achieve high accuracy on specific datasets, presenting substantial privacy risks for participants in sensitive datasets.

3. Atout Ticket Learning

In this section, we answer the following two research questions: **(RQ1)** Given a trained neural network, can we identify a subset of sensitive parameters of the network, having significant effects on the model's prediction? and **(RQ2)** What is the minimum size of such a subset? Our motivations for studying the previous questions is to investigate approaches that can leverage the advantages of both DP and SA for securing FL exchanges. However, these two questions are fundamental and answering them may have applications beyond FL settings.

Subsets of sensitive parameters can be formally represented using atout masks. The sensitivity of a model's parameters can be quantified using a threshold ϵ. Generally, this threshold corresponds to the desired increase in the model loss (or any metric) when altering sensitive parameters, and thus depends on the intended application and the desired confidence.

3.1. Atout Ticket Learning (ATL) problem

RQ1 and **RQ2** induce the Atout Ticket Learning (ATL) problem, expressed as an ℓ_0-norm optimization. ATL problem consists in finding a minimal atout mask in term of size such that the empirical risk of f_θ will increase/decrease in $f_{\theta\oplus\mathcal{M}}$ at least by a given threshold ϵ. Thus, $f_{\theta\oplus\mathcal{M}}$ can be seen as an approximation of f_θ with an error greater than a given threshold ϵ, where $\mathcal{M}_{\mathbb{T}}(\theta)$ is the solution space.

Problem 1 (*Atout Ticket Learning Using the Empirical Risk*). Given $\mathbb{T} \subseteq \mathbb{R}$, and a threshold $\epsilon > 0$:

$$\min_{\mathcal{M}\in\mathcal{M}_{\mathbb{T}}(\theta)} \quad \|\mathcal{M}\|_0 \tag{4}$$

$$\text{subject to} \quad |\mathcal{R}(f_\theta) - \mathcal{R}(f_{\theta\oplus\mathcal{M}})| \geq \epsilon \tag{5}$$

While $\mathcal{R}(f_\theta)$ is considered as a constant, the ATL problem is inherently NP-hard (non-convexity of the ℓ_0-norm and the loss function). Note that Problem 1 is parameterized with the threshold ϵ.

Definition 1 (*Atout Tickets*). Let $\mathcal{M} \in \mathcal{M}_{\mathbb{T}}(\theta)$ a solution (if exists) of Problem 1 for a given threshold ϵ. Parameters θ^l_{ji} of the network f_θ such that $\mathcal{M}^l_{ji} \neq 0$ are called the Atout Tickets (AT) of f_θ.

Several strategies can be explored to approximate the solutions of the ATL problem. A naive strategy is an exhaustive search by enumerating all elements of $\mathcal{M}_{\mathbb{T}}$. This assumes that $\mathbb{T}$ is finite and of reasonable size, which is usually not the case. Another strategy is to consider a convex relaxation of the ATL problem (e.g. replace ℓ_0 by ℓ_1 or ℓ_2-norm). These strategies quickly fail with large networks having millions/billions of parameters.

Rather than trying to solve the Problem 1 directly, we are interested in answering **RQ2**: can we derive a theoretical lower-bound for the minimum number of atout tickets needed to solve the ATL problem? The aim is to characterize factors influencing the choice of atout masks. So, before answering **RQ2**, one can notice that, if we assume that the loss function $\mathcal{L}$ is the absolute value, and f_θ is approximating a function f such that $\forall (X,Y) \in \mathcal{D}$, we have $f(X) = Y$, $f_\theta(X) \leq f(X)$ and $f_{\theta\oplus\mathcal{M}}(X) \leq f(X)$, then:

$$\begin{aligned}
|\mathcal{R}(f_\theta) - \mathcal{R}(f_{\theta\oplus\mathcal{M}})| &= |\frac{1}{|\mathcal{D}|}\sum_{(X,Y)\in\mathcal{D}} \mathcal{L}(f_\theta(X), f(X)) \\
&\quad - \frac{1}{|\mathcal{D}|}\sum_{(X,Y)\in\mathcal{D}} \mathcal{L}(f_{\theta\oplus\mathcal{M}}(X), f(X))| \\
&= |\frac{1}{|\mathcal{D}|}\sum_{(X,Y)\in\mathcal{D}} f_\theta(X) - f_{\theta\oplus\mathcal{M}}(X)| \\
&\leq max_{(X,Y)\in\mathcal{D}}\{|f_\theta(X) - f_{\theta\oplus\mathcal{M}}(X)|\}
\end{aligned}$$

Thus, instead of studying the Problem 1, we will study the following Problem 2.

Problem 2 (*Atout Ticket Learning Using Function Approximation*). Given $\mathbb{T} \subseteq \mathbb{R}$, and $\epsilon > 0$:

$$\min_{\mathcal{M}\in\mathcal{M}_{\mathbb{T}}(\theta)} \quad \|\mathcal{M}\|_0 \tag{6}$$

$$\text{subject to} \quad |f_\theta(X) - f_{\theta\oplus\mathcal{M}}(X)| \geq \epsilon, \quad \forall X \in \mathbb{R}^d \tag{7}$$

Next, under some assumptions, we provide a positive answer to **RQ2** for the Problem 2.

3.2. Main results

Without loss of generality, we will assume that the output of f_θ is bounded by C_θ. In fact, most used activation functions (Sigmoid, Tanh, …) are bounded. Although the ReLu output is not bounded, we usually use L1 or L2 penalties in the objective function.

Theorem 1. *Under the boundedness assumption, we have (K_θ is the Lipschitz constant of f_θ):*

$$\begin{aligned}
|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| \leq C_\theta \times \Bigg(& \|\mathcal{M}^{L+1}\|_\infty \times \|\mathcal{M}^{L+1}\|_0 \\
+ & \sum_{i=1}^{L} (K_\theta^{L+1-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \\
\times & \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j)) \Bigg)
\end{aligned}$$

Proof. See Appendix B.2 □

Theorem 1 sets an upper-bound on the error made when $f_{\theta\oplus\mathcal{M}}(X)$ is approximating $f_\theta(X)$. This upper-bound depends heavily on the layer parameter and the mask magnitudes.

Corollay 1 (*Lower-bound for ATL Problem*). *Given an $\epsilon > 0$, if $\mathcal{M}$ is a solution of the Problem 2 for a given f_θ, and $K_\theta \times \|\theta\|_\infty \times N_\theta \neq 1$ then:*

$$\frac{\epsilon \times \left(1 - K_\theta \times \|\theta\|_\infty \times N_\theta\right)}{C_\theta \times \left(1 - (K_\theta \times \|\theta\|_\infty \times N_\theta)^{L+1}\right)} \leq \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0$$

Proof. See Appendix B.3 □

As highlighted by Corollary 1, the lower-bound of $\|\mathcal{M}\|_0$ depends strongly on the magnitude of the layer parameters and the mask parameters. This suggests that layerwise magnitude-based selection strategies could be very effective. This result is consistent with recent findings [32,49] which show that the choice of simple pruning on the basis of parameter magnitudes, in the case of sparse neural network learning, results in surprisingly powerful unstructured selection strategies.

4. Atout Ticket Protocol as a privacy-preserving protocol for federated learning

In the previous section, we have introduced the ATL problem and derived a lower-bound on the number of AT needed for a given threshold ϵ, by answering questions **RQ1** and **RQ2**, respectively. As one potential application of these results, we show, in this section, how AT can be leveraged to secure FL exchanges, and therefor answer **RQ3**.

In FL settings (Fig. 6), a central server is delegated to send a global ML model to a bunch of clients, participating in the training. The training phase consists of multiple rounds until the model convergence or a given condition is met. In each round, the clients submit their locally updated models to the server who aggregates them and sends back the aggregated model to the clients. Although the client data is not leaving their local devices, FL presents several privacy issues and vulnerabilities [5,6,9,11–13,16,50].

In order to mitigate these issues, we propose the Atout Ticket Protocol (ATP). The intuitive idea behind ATP is that an extremely low number of parameters (AT) of a client model should be exchanged over the network in two secure forms: an altered version with the server and an encrypted version with a secure aggregator. Thus, ATP exploits advantages of both perturbation and SA techniques as:

- A few portion of the model's parameters are encrypted, and therefor do not affect memory usage and computation performance, and
- The model's parameters are only hidden during the network transfer and can be recovered on the server side. This approach is fundamentally different from pure perturbation techniques such as Deferential Privacy.

ATP is organized around 3 components (i) client strategies for selecting AT, (ii) AT alteration strategies, and (iii) ATP exchanges. For the threat model, ATP assumes that K participants ($k \geq 2$) are training a machine learning model using a given FL algorithm. The non-trusted party is the server or central entity that initializes the model and aggregates the results from all participants. We assume that the server is honest-but-curious: a legitimate participant in a communication protocol that will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages. Our approach does not consider the case where the server is malicious [51]. The primary goal of the attacker is to reconstruct the training data of a target participant using the parameters of that participant's model during a round of global model updates. We also assumes that all other participants are honest. Finally, the threat model is shown in Fig. 5.

4.1. Layerwise Magnitude-based Selection (LMS) strategies

LMS strategies of ATP allow FL clients to identify atout tickets corresponding to locally updated models θ. As the ATL problem is NP-hard, and based on the results of Corollary 1, we opt for layerwise magnitude-based selection strategies. Note that, for a comparison of different strategies, one can use the methodology introduced by Lee et al. [52]. Thus, we have designed six new classes of heuristics (structured and unstructured) to define the set of atout tickets:

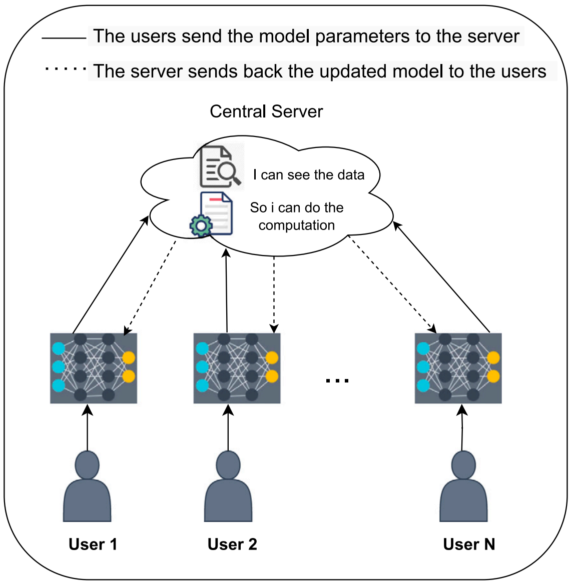

The users send the model parameters to the server
The server sends back the updated model to the users
Central Server
I can see the data
So i can do the computation
...
User 1
User 2
User N

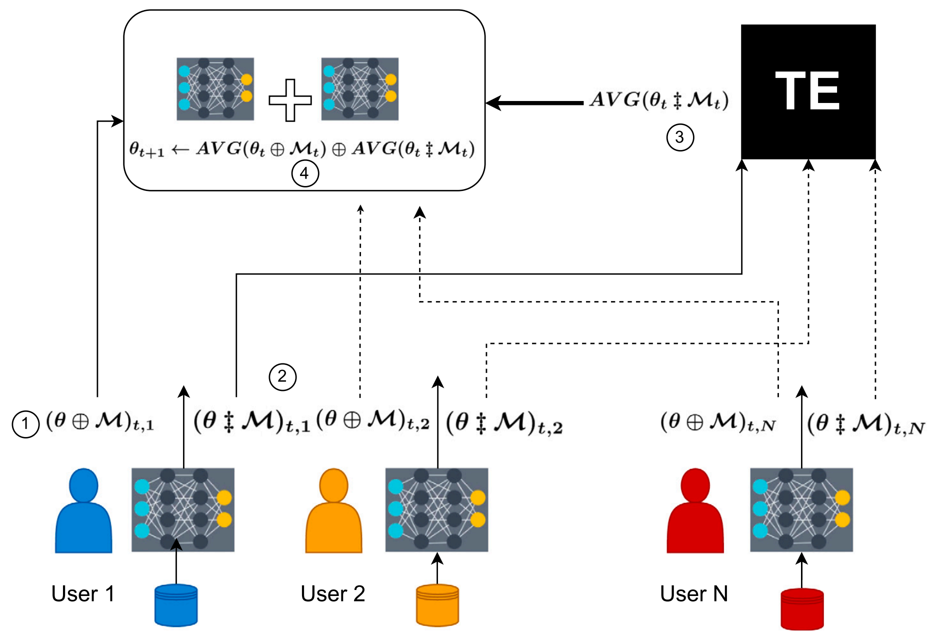
Fig. 5. Threat model for FL.

- **Global random strategy** (H_0). Unlike the global threshold on the parameter magnitudes of [53,54], we randomly choose α% model parameters to meet the global sparsity constraint. In our case, α is set to 2% (the optimal value found by experimentation).
- **Convolutional random strategy** (H_1). Rather than considering all network layers, we randomly choose α% parameters only for convolutional layers. In our case, α is set to 2% (the optimal value found by experimentation).
- **Global uniform strategy** (H_2). For every layer, the top-k parameters with the highest magnitude are chosen. In our case, k is set to the number of neurons of the layer (number of the kernels).
- **Convolutional uniform strategy** (H_3). For every convolution layer, the top-k parameters with the highest magnitude are chosen. The extension proposed by Evci et al. [32] of Mocanu et al. [55] method also takes convolutional layers into account. However, the number of chosen parameters per layer is scaled to be proportional to $1 - \frac{N_\theta^{l-1} + N_\theta^l + w^l + h^l}{N_\theta^{l-1} \times N_\theta^l \times w^l \times h^l}$, where w^l and h^l denote the width and height of the lth convolutional layer's kernel, respectively. In our case, the selection is based on the parameter magnitude, and k corresponds to the number of neurons of the layer.
- **One shot strategy** (H_4). This family of strategies is similar to H_0 and H_1 but only one layer is chosen. In our experiment, we randomly chose α% parameters of the first, second and last convolutional layers separately (α is set to 2%).
- **One shot strategy** + (H_5). This strategy is similar to the previous one except that we change the parameters of the last fully connected layer.

Handcrafted strategies, like the work of Gale et al. [49], can also be considered. Nevertheless, we did not cover them in our experiments.

4.2. AT alteration strategy

Once atout tickets are identified by a chosen LMS strategy, the next step corresponds to set the atout mask values. We opt for a mixed strategy combining inhibition and additive noise masks. Firstly, we initialize the mask $\mathcal{M}$ to $\mathcal{M}_0$ (the empty mask). Secondly, we generate a noise ϵ_{ji}^l, according to a chosen distribution (Table 2). Finally, we



$\theta_{t+1} \leftarrow AVG(\theta_t \oplus \mathcal{M}_t) \oplus AVG(\theta_t \ddagger \mathcal{M}_t)$
4
$AVG(\theta_t \ddagger \mathcal{M}_t)$
3
TE
2
1 $(\theta \oplus \mathcal{M})_{t,1}$
$(\theta \ddagger \mathcal{M})_{t,1}$
$(\theta \oplus \mathcal{M})_{t,2}$
$(\theta \ddagger \mathcal{M})_{t,2}$
$(\theta \oplus \mathcal{M})_{t,N}$
$(\theta \ddagger \mathcal{M})_{t,N}$
User 1
User 2
User N

Fig. 6. Atout Ticket Protocol (ATL). ① Noise is introduced to the sensitive parameters before they are dispatched to the central server, ② These parameters are then encrypted and sent to the Trusted Entity (TE), ③ The TE decrypts, aggregates, and sends the encrypted aggregate parameters back to the central server, ④ Finally, non-sensitive parameters are aggregated and combined with the received parameters from the TE through element-wise addition.

Table 2
Mask normal distributions.

Hypothesis	Attacks	ϵ^l_{ji}
H_0, H_1, H_2, H_3	DLG, iDLG, IG	$\mathcal{N}(0, min(\|\theta^l\|))$
H_4	IG	$\mathcal{N}(0, min(\|\theta^l\|))$
	DLG, iDLG	$-\|\mathcal{N}(\|\|var(\theta^l)\|\|_\infty, min(\|\theta^l\|))\|$
H_5	DLG, iDLG	$-\|\mathcal{N}(\|\|var(\theta^l)\|\|_\infty, max(\|\theta^l\|))\|$

update the mask for the atout ticket given by the heuristic accordingly: $\mathcal{M}^l_{ji} = -\theta^l_{ji} + \epsilon^l_{ji}$.
where, $min(\theta^l)$ (resp. $max(\theta^l)$) is the lowest (resp. highest) value of layer l, and $var(\theta^l)$ is the variance vector of layer l neurons. In the case of Deep Leakage from Gradients (DLG) and improved Deep Leakage from Gradients (iDLG) attacks, in addition to the training data, the attackers try to extract the labels based on the sign of the gradients. To strengthen our approach, we support our previous strategies with H_5 which is similar to H_4 except that it exploits the sign of the gradients in addition to their magnitudes. In fact, as it has been proven in [56], a smaller Fisher Information Loss implies a larger variance.

4.3. Atout Ticket Protocol (ATP) exchanges

ATP highlights a new era of privacy-preserving distributed ML which combines encryption and differential-privacy. ATP assumes that three entities are involved in the exchanges: (i) a honest-but-curious FL server (ii) legitimate clients and (iii) a Trusted Entity (TE). TE can be either a hardware trusted execution environment such as Intel's SGX or a purely software solution. Before describing the ATP exchanges, we define the operator $\theta \ddagger \mathcal{M}$ as follows:

$$\theta \ddagger \mathcal{M} = \begin{cases} 0, & if\ \mathcal{M}^l_{ij} = 0 \\ Encrypted(-\mathcal{M}^l_{ij}), & otherwise \end{cases}$$

ATP exchanges are depicted in Fig. 6 as follows:

- Client side (cf. Algorithm 1):
  1. A client receives the parameters from the server.
  2. It trains the model locally on its own dataset.
  3. It issues a remote attestation protocol to verify the trustworthy of the trusted entity and to exchange the encryption keys.
  4. It computes $\theta \oplus \mathcal{M}$ (differential-privacy part) and $\theta \ddagger \mathcal{M}$ (encrypted part). Then, it sends $\theta \oplus \mathcal{M}$ to the server and $\theta \ddagger \mathcal{M}$ to the TE.
- TE side (cf. Algorithm 2):
  1. TE receives $\theta \ddagger \mathcal{M}$ from each client.
  2. TE decrypts non-zero values of $\theta \ddagger \mathcal{M}$.
  3. TE averages $\theta \ddagger \mathcal{M}$, then sends $AVG(\theta \ddagger \mathcal{M})$ to the server.
- Server side (cf. Algorithm 3):
  1. The server receives $\theta \oplus \mathcal{M}$ from each client.
  2. The server receives $AVG(\theta \ddagger \mathcal{M})$ from the TE.
  3. It averages $\theta \oplus \mathcal{M}$, then element-wise adds it to $AVG(\theta \ddagger \mathcal{M})$.
  4. It sends back the updated model to all clients.

Algorithm 1: Client Update

```
Input: Global Model $\theta$
Output: Parameters $\theta \oplus \mathcal{M}$ and $\theta \ddagger \mathcal{M}$
B ; // The number of batches
E ; // The number of epochs
foreach epoch from 1 to E do
    foreach $b \in B$ do
        $\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta, b)$
    end
end
Generate the appropriate mask $\mathcal{M}$
Compute $\theta \oplus \mathcal{M}$ and send it to the server
Compute $\theta \ddagger \mathcal{M}$ and send it to the TE
```

Algorithm 2: TE Averaging

```
1  Input: Clients $C = \{c_1, c_2, ...c_N\}$
2  Output: $\theta \ddagger \mathcal{M}_t$
3  foreach round $t = 1, 2, 3...$ do
4      Receive $\theta \ddagger \mathcal{M}_{t,j}$ from each Client $C_j$
5      foreach $\theta \ddagger \mathcal{M}_{t,j} \in \mathcal{M}_t$ do
6          $\theta \ddagger \mathcal{M}_{t,j} \leftarrow decrypt(\theta \ddagger \mathcal{M}_{t,j})$
7      end
8      $AVG(\theta \ddagger \mathcal{M}_t) \leftarrow \frac{1}{N}\sum_{j=1}^{N} \theta \ddagger \mathcal{M}_{t,j}$
9      Send $AVG(\theta \ddagger \mathcal{M}_t)$
10 end
11 to the Server
```

Algorithm 3: Server Averaging

```
1  Input: Clients $C = \{c_1, c_2, ...c_N\}$
2  Output: Parameter $\theta$
3  foreach round $t = 1, 2, 3...$ do
4      Receive $(\theta \oplus \mathcal{M})_{t,j}$ from each client $c_j$
5      Receive $AVG(\theta \ddagger \mathcal{M})_t$
6  from the TE
7  Compute $AVG(\theta \oplus \mathcal{M})_t \leftarrow \frac{1}{N}\sum_{j=1}^{N} (\theta \oplus \mathcal{M})_{t,j}$
8  Compute $\theta_{t+1} \leftarrow AVG(\theta \oplus \mathcal{M})_t \oplus AVG(\theta \ddagger \mathcal{M})_t$
9  Send the global model $\theta_{t+1}$ to the clients
```

The additional overhead, added by ATP compared to standard FL exchanges, is negligible in terms of computation and communication. In fact, the computation of atout ticket does not require model retraining. Furthermore, the amount encrypted atout tickets do not exceed a few kilobytes. Through a series of experiments, we show that the number of atout tickets is less than 10^{-4} of the total number of parameters. In term of additional communication exchanges, ATP adds the TE messages $\theta \ddagger \mathcal{M}$ in each round. However, these tensors are extremely sparse, and their overhead cost is negligible when an efficient storage scheme is applied [57]. Note that, in our settings, TE can also be hosted on the server using hardware enclaves as the case of Intel SGX [47] enclaves. Furthermore, the used noise for altering the AT of client models is subtracted when the server aggregates $AVG(\theta \oplus \mathcal{M})$ and $AVG(\theta \ddagger \mathcal{M})$. This is crucial for avoiding performance degradation resulted by applying DP on AT of client models.

Theorem 2. *At each round, the values computed by Algorithm 3 of the server corresponds to the average of locally updated client models.*

Proof. See Appendix B.4. □

5. Experiments

In this section, we report the results of the ATP protocol using the six Layer-wise Magnitude-based Selection (LMS) strategies against FL reconstruction attacks. For FL reconstruction attacks, namely Inverting Gradients (**IG**) [7], Deep Leakage (**DLG**) [5], and Improved Deep Leakage (**iDLG**) [6], we investigate the attacker's ability in reconstructing raw data. For that, we implement the three attacks with the same settings as those provided by the authors in their corresponding papers. In our experiments, four datasets (**MNIST** [58], **CIFAR-10/100** [59], and **LFW** [60]) and three models (**ResNet20**, **ConvNet** and **LeNet**) are used. To test the performance of LMS strategies, we keep the same metrics as those used in the aforementioned attacks, namely Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Foveated Mean Squared Error (FMSE), and Cross-entropy loss (Loss). The experiment results are computed by mean and standard deviation for the first 100 images of all used datasets. *The source code is available at github.*[2]

5.1. Experiment settings

Experiments are implemented using PyTorch framework [61], and are conducted using one Tesla P40 GPU with 24 GB GPU memory. In what follows, we describe our experiment settings.

Datasets. In our experiments, we use the following datasets:

- **MNIST** [58] dataset corresponds to handwritten digit images of ten categories ranging from 0 to 9. It contains 60,000 training images and 10,000 testing images.
- **CIFAR-10** [59] dataset consists of 60 000 32 × 32 color images, categorized into 10 classes, with 6000 images per class. There are 50 000 training images and 10 000 test images.
- **CIFAR-100** dataset has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.
- **LFW** [60] dataset contains 13,233 images of faces collected from the web. It includes 5749 identities with 1680 people having two or more images.

Models. The used architectures in the experiments are the following:

- **LeNet** consists of three convolutional layers each with segmoid activation function, followed by one fully-connected layer.
- **ConvNet** consists of eight sets of convolution, batch normalization, relu activation layers, followed by one pooling and one fully-connected layer.
- **ResNet20** consists of 20 convolutional layers, 8 batch normalization layers, and one fully-connected layer.

Metrics. To test the performance of our approach, we use the same metrics as those used in the aforementioned attacks:

- **Mean Square Error**(MSE) measures the squared error between estimated values and the actual values.
- **Peak Signal-to-Noise Ratio** (PSNR) is an image quality metric. The PSNR computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and the modified images. The higher the PSNR, the better the quality of the reconstructed image.
- **Foveated Mean Squared Error** (FMSE) is a video quality metric which calculates the difference between frames of the original and distorted sequences.
- **Cross-entropy loss** (Loss) measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

Model hyperparameters. Table 3 presents the total number of parameters of each model according to the used dataset. For the reconstruction attacks, the used hyperparameters are presented in Tables 4 and 5. N is the hidden unit size, LR is the learning rate, and TV is the total variation, Max iteration is the maximal number of iterations per optimization step, history size is the update history size.

Overcoming implementation hurdles. Overcoming implementation hurdles in FL systems can be achieved through several key strategies. First, we emphasize on the importance of optimizing data handling. In this context, employing standardized pre-processing techniques ensures consistency across diverse datasets, while data normalization and augmentation help mitigate discrepancies and enhance model performance. Second, selecting flexible CNN architectures that can be

[2] https://github.com/bouayad42/SecureFL.

Table 3
The total number of parameters of the used models.

Model	Dataset	Total number of parameters
ResNet20	CIFAR-10	272 474
ConvNet	CIFAR-10	2 904 970
	CIFAR-100	85 036
LeNet	MNIST	13 426
	LFW	4 429 117

Table 4
Hyperparameters for DLG and iDLG attacks.

Architecture	Dataset	N	LR	Iterations	History size	Max iterations
	MNIST	588	1.0	300	100	20
LeNet	CIFAR-100	768	1.0	300	100	20
	LFW	768	1.0	300	100	20

Table 5
Hyperparameters for IG attack.

Architecture	Local LR	Local step size	TV	LR	LR decay	Iterations
ResNet20 & ConvNet	1e−4	5	1e−6	1.0	3/8, 5/8, 7/8	24 000

easily adapted to different datasets is essential. Furthermore, leveraging transfer learning by fine-tuning pre-trained models on specific datasets can also improve efficiency and accuracy. Lastly, addressing Non-IID data challenges through advanced aggregation methods is recommended to effectively manage variability in data distribution across users. These approaches collectively provide a robust framework for tackling common implementation challenges in federated learning.

5.2. Experimental results

A reconstruction attack is a privacy attack on aggregate data that reconstructs sensitive information of a raw dataset from public aggregate information in the compute party server. In this case, the adversary's goal is to use its knowledge of the model's parameters to reconstruct raw private data. Reconstruction attacks require white-box access to the ML model. That is, we need have access to the model's parameters. The idea of the mentioned attacks is simple, and an adversarial user can leverage the model's parameters in order to extract the sensitive data that the model has been trained on. First, the attacker has to know the shape of the original dataset's samples and then initialize a sample dataset of the same ground truth shape. After that, the attacker trains a model on the randomly created dataset. At that point, the attacker's goal is to optimize the objective function, which consists in minimizing the distance between the ground truth gradients and the randomly generated gradients. In what follows, we report the results of ATP against the three evaluated reconstruction attacks. For each attack, the corresponding masks are introduced in Table 2.

5.2.1. IG attack

The IG attack is performed on two models, namely ResNet20 and ConvNet, using the CIFAR-10 dataset. Table 6 reports the obtained results. **AT** refers to the number of atout tickets of a model when applying an LMS strategy (heuristic). For the sake of simplicity, the threshold expresses the attack's model degradation, and it is computed as the MSE loss increment from the baseline loss of a model. For the IG attack, the experiments show a significant resilience against this attack. In all LMS strategies, the attacker could not extract/recover any image by just using intermediate parameters of the FL system participants. The results in Table 6 show also that heuristic 4 ($H4$) is the optimal choice due to its low requirements for parameter encryption. The threshold ϵ in this case is low. In addition, the results show that shallow networks are more sensitive to perturbation than deep network architectures.

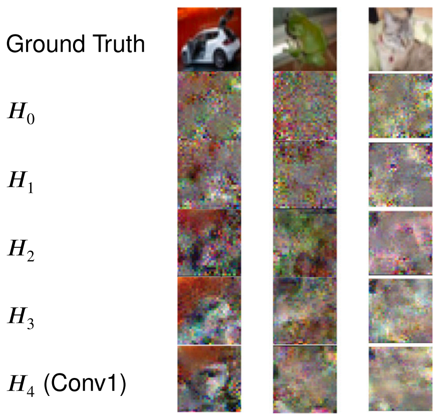


Fig. 7. Examples of IG reconstruction using ConvNet.

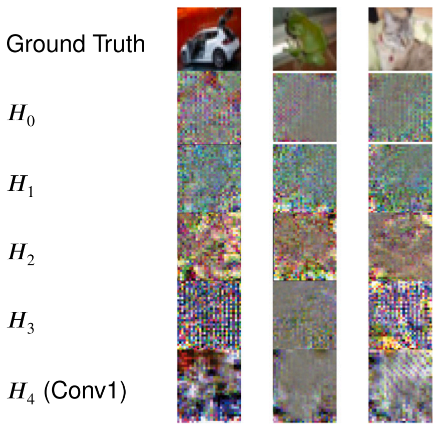


Fig. 8. Examples of IG reconstruction using ResNet20.

Besides, it prevents the attacker from gaining any information about the training data. Hence, it is an optimization in terms of communication, privacy, and utility. Finally, Table 7 presents the best results achieved for the IG attack using different metrics.

5.2.2. DLG attack

DLG is an optimization algorithm that regenerates the training samples iteratively by optimizing the input to produce suitable gradients for a user. The objectives here is to recover both the input image and the corresponding label. Table 8 reports the obtained results for the DLG attack. ACC refers to the accuracy of the extracted label performed by the attacker. From heuristic H_0 to heuristic H_4, our approach succeeded to prevent the attacker from extracting any information except the training labels, which we dealt with it using the heuristic H_5 that prevent the attacker from extracting the training labels. Therefore the best heuristic that guarantees privacy, preserves utility, and maintain computation complexity is heuristic H_5. The best found results are presented in Table 9.

5.2.3. iDLG attack

iDLG is an optimization of DLG, especially in extracting the training labels. The authors discovered a new method which relay mainly on the sign of the gradients. This method succeeded to extract the training labels with 100% accuracy. The obtained results of ATL against iDGL are detailed in Tables 10 and 11. In the case of heuristic H_5, we also change the sign and the magnitude of the model's parameters according to Table 2. The obtained results for iDLG are similar to those of DLG.

The overall results reveal that all models incurred a loss in terms of performance for all conducted experimentation using different LMS

Table 6
ATP resilience against IG attack on CIFAR-10.

Heuristic	Network	Layer	MSE	FMSE	PSNR	AT	~AT (%)	ϵ
IG Baseline	ResNet20	–	0.9430 (±0.46)	6.81E−01 (±3.61)	0.7228 (±2.07)	–	0.000	–
	ConvNet	–	0.050 (±0.04)	1.55E−09 (±1.77E−09)	14.68 (±4.0)	–	0.000	–
H_0	ResNet20	Conv + FC	1.6380 (±0.74)	9.78E+01 (±9.82E+01)	−1.72 (±1.97)	5290	2	0.69
	ConvNet	Conv + FC	1.0665 (±0.46)	9.06E−08 (±4.64E−08)	−0.88 (±1.87)	58 350	2	1.01
H_1	ResNet20	Conv	1.7115 (±0.8)	2.26E+01 (±1.75E+01)	−1.79 (±2.33)	5276	1.936	0.76
	ConvNet	Conv	1.2061 (±0.52)	2.93E−07 (±2.78E−07)	−0.46 (±1.74)	57 903	1.993	1.15
H_2	ResNet20	Conv + FC	1.8226 (±0.7)	4.565 (±9.2)	−2.20 (±1.9)	698	0.256	1.34
	ConvNet	Conv + FC	2.1802 (±0.9)	1.8504E−07 (±1.3)	−3.18 (±1.4)	1610	0.055	2.13
H_3	ResNet20	Conv	1.7855 (±0.7)	4.137 (±2.2E+01)	−2.21 (±1.9)	688	0.252	0.84
	ConvNet	Conv	2.1742 (±0.7)	1.709E−07 (±1.03E−7)	−3.14 (±1.4)	1600	0.055	2.12
H_4	ResNet20	Conv 1	0.8846 (±0.3)	3.135E−01 (±9.1E−01)	0.96 (±1.9)	9	0.003	0.05
		Conv 2	1.0290 (±0.4)	8.408E−01 (±1.8)	0.33 (±2.07)	47	0.017	0.97
		Last Conv	1.7142 (±0.8)	2.487 (±3.2)	−1.86 (±2.1)	738	0.270	0.77
	ConvNet	Conv 1	0.2738 (±0.1)	8.027E−09 (±8.01E−9)	6.44 (±2.7)	35	0.001	0.22
		Conv 2	0.2731 (±0.1)	9.017E−09 (±6.4E−9)	6.40 (±2.7)	1475	0.050	0.023
		Last Conv	0.8116 (±0.2)	3.470E−08 (±1.2E−08)	1.15 (±1.4)	11 797	0.406	0.76

Table 7
Best found results over IG attack experiments on CIFAR-10 dataset, using ResNet20 and ConvNet networks.

Metric	ConvNet	ResNet
MSE	H_2: 2.1802 (±0.9)	H_2: 1.8226 (±0.7)
FMSE	H_1: 2.93E−07 (±2.78E−07)	H_0: 9.78E+01 (±9.82E+01)
PSNR	H_2: −3.18 (±1.4)	H_3: −2.21 (±1.9)
AT	H_4, Conv 1: 35	H_4, Conv 1: 9
ϵ	H_2: 2.13	H_2: 1.34

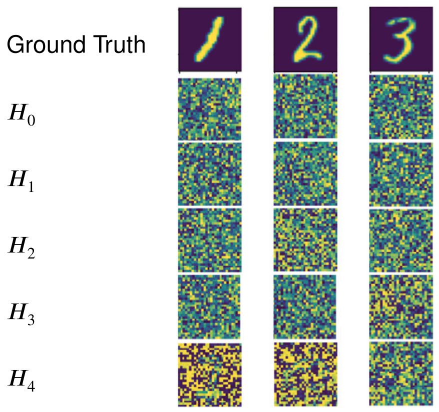


Fig. 9. Examples of DLG reconstruction on MNIST.

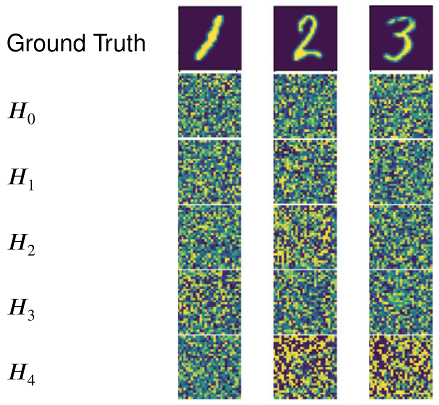


Fig. 10. Examples of iDLG attack on MNIST.

strategies, and therefore failed to construct the original images. Figs. 7 and 8 illustrates a sample of reconstructed images using IG attack on CIFAR-10 for ResNet20 and ConvNet models. Figs. 9 and 10 illustrate the failure of reconstruction for both DLG and iDLG on MNIST. On one hand, the overall results show that the percentage of AT (AT (%)) is extremely small for all employed heuristics and models. Thus, ATP requires an extremely small fraction of model's sensitive parameters (AT) to be encrypted for resilience to reconstruction attacks. In terms of the minimal number of ATs, the hypothesis H_4 (Conv 1) seems to be the most efficient one. The ResNet20 architecture is too vulnerable to a slight change in the model parameters: altering only 9 high magnitude parameters in the first convolutional layer makes the model unable to perform the reconstruction. This is also valid for ConvNet. In contrast, for ConvNet, as we can see in Fig. 7, images are started to be visually recognizable. Therefore, and from what being said, the shallow network is more sensitive to alterations than a deep network. Thus, for IG attack, heuristic H_3 seems to be the most effective way to protect users' data as it depends on a plausible number of atout tickets and it works for both shallow and deep networks. On the other hand, the hypothesis H_5 is more effective in preventing label reconstruction, and might require hiding more ATs.

5.2.4. IG ablation study

In order to achieve a better understanding of the effectiveness of our proposed approaches, we provide an ablation study for the IG attack on H_0. The results are presented in Table 12. Our ablation study consists of changing the random distribution and its parameters for the noise masks $\mathcal{M}$. For our proposed method, we used a normal distribution $\mathcal{N}(0, min(|\theta^l|))$ as a main random distribution. However, in this study, we used $min(\theta^l)$ and $max(\theta^l)$ parameters for both distributions $\mathcal{N}$ and $\mathcal{U}$ (normal and uniform distributions). As illustrated in Table 12, the normal distribution $\mathcal{N}$ with $min(\theta^l)$ has achieved, in general, good performance compared to the other distributions.

6. Related work

6.1. Privacy concerns in machine learning

In This section, we will mainly review three categories of attacks that relate to our work: reconstruction attacks, feature estimation attacks, and membership inference attacks. Some attacks leverage access to model parameters to perform the attack (white-box attack), while other attacks only use the model prediction to perform the attack (black-box attack).

Table 8
ATL resilience against DLG attack performed over CIFAR-100, MNIST and LFW datasets, using LeNet network.

Heuristic	Dataset	Layer	LOSS	MSE	ACC	AT	∼ AT (%)	ϵ
DLG baseline	CIFAR-100	–	206.37	0.00036	100%	–	–	–
	MNIST	–	91.07	8.7995E−09	100%	–	–	–
	LFW	–	220.62	5.3622E−05	100%	–	–	–
H_0	CIFAR-100	Conv + FC	1773.3 (±473.1)	6.5E+21 (±1.91E+23)	84%	1700	2	6.5E+21
	MNIST	Conv + FC	247.6 (±255.07)	7.73E+23 (±2.44E+25)	88%	269	2	7.73E+23
	LFW	Conv + FC	8.77E+04 (±1.50E+03)	1.22E+29 (±3.77E+30)	71%	88 582	2	1.22E+29
H_1	CIFAR-100	Conv	285.6 (±446.3)	3.6E+15 (±4.8E+16)	82%	163	0.19	3.6E+15
	MNIST	Conv	132.1 (±236.7)	7.7E+22 (±2.4E+24)	90%	150	1.11	7.7E+22
	LFW	Conv	2.85E+02 (±4.25E+02)	6.66E+22 (±1.78E+24)	69%	163	0.03	6.66E+22
H_2	CIFAR-100	Conv + FC	519.7 (±491.3)	1.6E+23 (±3.2E+24)	84%	136	0.15	1.6E+23
	MNIST	Conv + FC	137.13 (±258.4)	2.3E+22 (±5.8E+23)	88%	46	0.34	2.3E+22
	LFW	Conv + FC	5.96E+03 (±4.17E+02)	1.29E+23 (±3.21E+24)	80%	4 415 268	99	1.29E+23
H_3	CIFAR-100	Conv	194.08 (±393.5)	2.5E+19 (±2.9E+24)	86%	36	0.04	2.5E+19
	MNIST	Conv	101.2 (±252.6)	8.23E+21 (±2.5E+26)	90%	36	0.26	8.23E+21
	LFW	Conv	2.60E+02 (±4.17E+02)	1.59E+20 (±3.05E+21)	82%	36	0.0008	1.59E+20
H_4	CIFAR-100	Conv 1	221.2 (±468.8)	4.33E+20 (±8.7E+21)	84%	12	0.01	4.33E+20
		Conv 2	218.05 (±450.5)	4.8E+20 (±1.5E+22)	84%	12	0.01	4.8E+20
		Last Conv	601.08 (±511.3)	1.2E+17 (±2.1E+18)	84%	12	0.01	1.2E+17
	MNIST	Conv 1	256.9 (±89.8)	13E+23 (±2.9E+24)	91%	12	0.08	1.3E+24
		Conv 2	96.1 (±250.4)	9.2E+23 (±2.8E+25)	89%	12	0.08	9.2E+23
		Last Conv	105.2 (±253.8)	9.4E+23 (±2.8E+25)	89%	12	0.08	9.4E+23
	LFW	Conv 1	306.6 (±485)	6.54E+18 (±4.17E+19)	95%	12	0.0002	6.54E+18
		Conv 2	0.71 (±0.72)	5.13 (±7.35)	83%	12	0.0002	5.1263
		Last Conv	444.23 (±571.9)	8.9E+09 (±3.5E+10)	89%	12	0.0002	9.4E+23
H_5	CIFAR-100	Last FC	2.07 (±1.45)	9.9E+23 (±2.3E+25)	27%	780	0.9	9.9E+23
	MNIST	Last FC	1757 (±1229)	9.73E+20 (±3.10E+22)	31%	784	4.3	7.5E+17
	LFW	Last FC	Nan	5.02E+29 (±1.5E+31)	8%	441 523	10	5.02E+29

Table 9
Best found results over DLG attack experiments on CIFAR-100, MNIST, and LFW datasets, using LeNet network.

Dataset	LOSS		MSE		ACC	
	Baseline	AT	Baseline	AT	Baseline	AT
CIFAR-100	206.37	H_0: 1773.3 (±473.1)	0.00036	H_5: 9.9E+23 (±2.3E+25)	100%	H_5: 27%
MNIST	91.07	H_5: 1757 (±1229)	8.7995E−09	H_0: 7.73E+23 (±2.44E+25)	100%	H_5: 31%
LFW	220.62	H_5: Nan	5.3622E−05	H_5: 5.02E+29 (±1.5E+31)	100%	H_5: 8%

Table 10
ATL resilience against iDLG attack performed over CIFAR-100, MNIST and LFW datasets.

Heuristic	Dataset	Layer	LOSS	MSE	ACC	AT	∼ AT (%)	ϵ
iDLG baseline	CIFAR-100	–	43.19	0.00078	100%	–	–	–
	MNIST	–	17.32	1.8874E−07	100%	–	–	–
	LFW	–	18.31	3.3356E−05	100%	–	–	–
H_0	CIFAR-10	Conv + FC	1599 (±206.3)	187.5E+20 (±5.9E+20)	100%	1700	2	1.875E+22
	MNIST	Conv + FC	164.5 (±89.5)	6.3E+21 (±2.36E+23)	100%	269	2	6.3E+21
	LFW	Conv + FC	8.76E+04 (±1.44E+03)	2.58E+12 (±6.29E+13)	100%	88 582	2	2.58E+12
H_1	CIFAR-10	Conv	106.7 (±160.4)	2.24E+10 (±3.2E+11)	100%	163	0.19	2.24E+10
	MNIST	Conv	57.98 (±66.9)	4.39E+18 (±1.38E+20)	100%	150	1.11	4.39E+18
	LFW	Conv	5.80E+01 (±8.58E+01)	2.16E+15 (±6.78E+16)	100%	163	0.03	2.16E+15
H_2	CIFAR-10	Conv + FC	241.2 (±170.5)	3.7E+14 (±7.15E+15)	100%	136	0.15	3.7E+14
	MNIST	Conv + FC	48.28 (±73.25)	6.9E+18 (±2.1E+20)	100%	46	0.34	6.9E+18
	LFW	Conv + FC	6.05E+03 (±4.29E+02)	1.15E+21 (±2.26E+22)	100%	4 415 268	9	1.15E+21
H_3	CIFAR-10	Conv	62.10 (±152.1)	1.24E+13 (±1.67E+14)	100%	36	0.04	1.24E+13
	MNIST	Conv	27.7 (±68.9)	1.2E+20 (±3.1E+21)	100%	36	0.26	1.2E+20
	LFW	Conv	3.96E+01 (±1.08E+02)	7.67E+18 (±1.90E+20)	100%	36	0.0008	7.67E+18
H_4	CIFAR-10	Conv 1	38.98 (±144.4)	2.4E+14 (±3.4E+15)	100%	12	0.01	2.4E+14
		Conv 2	37.22 (±132)	8.6E+18 (±2.1E+20)	100%	12	0.01	8.6E+18
		Last Conv	439.8 (±323.1)	5.7E+22 (±1.3E+24)	100%	12	0.01	5.7E+22
	MNIST	Conv 1	13.12 (±63.8)	9.6E+18 (±1.9E+20)	100%	12	0.08	9.6E+18
		Conv 2	16.17 (±69.6)	1.6E+16 (±5.1E+17)	100%	12	0.08	1.6E+16
		Last Conv	27.9 (±83.1)	4.4E+15 (±1.38E+17)	100%	12	0.08	4.4E+15
	LFW	Conv 1	27.49 (±136.6)	1.02E+18 (±6.6E+18)	100%	12	0.0002	1.02E+18
		Conv 2	0.26 (±0.46)	1.49 (±3.59)	40%	12	0.0002	0.47
H_5	CIFAR-10	Last FC	1.74 (±1.37)	2,03E+21 (±6.4E+22)	21%	780	0.9	2.03E+21
	MNIST	Last FC	1632 (±1185)	3.93E+22 (±1.26E+24)	15%	784	4.3	5.5E+16
	LFW	Last FC	Nan	9.0E+17 (±2.5E+19)	40%	441 523	10	9E+17

Table 11
Best found results over iDLG attack experiments on CIFAR-100, MNIST, and LFW datasets, using LeNet network.

Dataset	LOSS		MSE		ACC	
	Baseline	AT	Baseline	AT	Baseline	AT
CIFAR-100	43.19	H_0: 1599 (±206.3)	0.00078	H_4, last: 5.7E+22 (±1.3E+24)	100%	H_5: 21%
MNIST	17.32	H_5: 1632 (±1185)	1.8874E−07	H_5: 3.93E+22 (±1.26E+24)	100%	H_5: 15%
LFW	18.31	H_5: Nan	3.3356E−05	H_2: 1.15E+21 (±2.26E+22)	100%	H_5: 40%

Table 12
Ablation study performances against IG attack on CIFAR-10 for heuristic H_0.

Network	Law	MSE	FMSE	PSNR
ConvNet	$\mathcal{N}(0, min(\|\theta^l\|))$	1.07 (±0.46)	9.06E−08 (±4.64E−08)	−0.88 (±1.87)
–	$\mathcal{N}(0, max(\|\theta^l\|))$	0.27 (±0.4)	3.4E−09 (±2.7E−07)	0.07 (±1.86)
–	$\mathcal{U}(0, min(\|\theta^l\|))$	1.04 (±0.5)	7.5E−05 (±5.3E−04)	0.8 (±3.4)
–	$\mathcal{U}(0, max(\|\theta^l\|))$	1.01 (±0.5)	1.98 (±1.57)	0.55 (±2.32)
ResNet20	$\mathcal{N}(0, min(\|\theta^l\|))$	1.6380 (±0.74)	9.78E+01 (±9.82E+01)	−1.72 (±1.97)
–	$\mathcal{N}(0, max(\|\theta^l\|))$	1.70 (±0.79)	7.32E+01 (±6.71E+01)	−1.82 (±2.16)
–	$\mathcal{U}(0, min(\|\theta^l\|))$	1.56 (±0.78)	1.42 (±2.43)	−1.52 (±1.86)
–	$\mathcal{U}(0, max(\|\theta^l\|))$	1.34 (±0.39)	8.34E+03 (±1.27E+04)	−1.13 (±1.09)

Reconstruction attacks. Phong et al. [4] demonstrated theoretically and empirically the feasibility of attacking a single neuron or a single-layer network. Moreover, a curious but honest server can reconstruct the clients' training inputs using only a small portion of the gradients. Zhu et al. [5] introduced the Deep Leakage (DLG) attack that uses an optimization-based approach to reconstruct training samples by attempting to match the client's gradient. Zhao et al. [6], through the Improved Deep Leakage (iDLG) attack, improved the single image reconstruction speed of Zhu et al. [5] approach by adding a label restoration step. Geiping et al. [7] proved that inverting gradients is possible not only for shallow networks but also for deep networks (Inverting Gradients (IG) attack). Pasquini et al. [62] proposed a novel attack that involves the server sending different malicious parameters to each user. The attack's goal is to breach the security of FL's Secure Aggregation (SA) and uncover a target user's model updates. Wen et al. [63] established an attack that modifies the last layer of the model sent to a user. The attack isolates a target data point by increasing its gradient and decreasing others'. The attacker's goal is to get a gradient from a single sample. Boenisch et al. [64] introduced a new method of reconstructing user data: a malicious server secretly alters its shared model's weights, sends them to the users, and then uses the gradients it receives to recover the user data. The method achieved a high success rate of over 50% for the ImageNet [65] dataset and over 25% for the IMDB-sentiment-analysis dataset [66].

Feature information attacks. Ateniese et al. [8] trained a model based on the target model to infer some private information from the training set. Hitaj et al. [9] proved that a malicious participant in a collaborative learning can take advantage of the GAN properties and the model's parameters to extract information from the training dataset. However, this attack is limited to samples of the same class. Aly et al. [67] examined how forecasting models can expose global properties and privacy risks in smart grid systems. They showed that by querying an LSTM model with black box access, they can infer the global properties of an honest user, such as the number of children, occupants, and the existence of desktop and console devices. Gu and Bai [68] investigated how an edge user's private data labels can be estimated by a cloud server that is honest but curious, by proposing a method to infer the label distribution from the FL process in edge computing.

Membership Inference Attacks (MIA). Shokri et al. [10] pioneered the MIA. Their idea is to train several meta-models that mimic the behavior of the target model. The trained models are used to predict whether a certain input was involved in the training of the target model or not. The technique proposed by Salem et al. [11] relies on a single shadow model instead of multiple shadow models and assumes no knowledge of target model structure or data distribution. Hayes et al. [12] used GANs to recognize private training datasets. Melis et al. [13] performed the attack in collaborative learning by leveraging the periodic updates (gradients) coming from the participants during the training phase. Wang et al. [14] considered a malicious server in the FL system and leveraged GAN with a multi-task discriminator to recover user-specific private data. Sablayrolles et al. [16] demonstrated that the attack depends more on the loss function than on the classifier parameters. Nasr et al. [15] studied the possibility of performing the attack in the case of FL where the attacker can exploit only model updates or modify the target model in order to extract more information. Carlini et al. [69] argued that average-case metrics, such as accuracy or ROC-AUC, are not adequate for measuring worst-case privacy in MIA. The authors suggested evaluating attacks by their true-positive rate (TPR) at low false-positive rates (FPR), a common method in security. Their LiRA, which uses per-example difficulty scores and Gaussian likelihood estimates, outperformed previous attacks in the low-FPR regime. Ye et al. [70] introduced a hypothesis testing framework for MIA, ensuring consistent comparisons and revealing data point vulnerabilities. The authors also introduced a high-performance MIA using reference models for enhanced privacy protection.

6.2. Privacy-preserving machine learning

In the past few years, privacy-preserving machine learning has attracted widespread interest due to its vulnerability to model attacks. Mainly, two lines of model defense can be identified: Differential Privacy (DP) and cryptography-based methods.

DP is an obfuscation mechanism where the primary goal is to reduce the precision of the data or the model by adding noise to prevent leaking private information. This is achieved by developing differentially-private ML models and optimization mechanisms [71,72]. In order to secure distributed and collaborative learning models, several FL-based DP methods have been introduced [45,73–79]. However, these methods generally decrease the model performance and are less effective with a large number of participants. In the context of Intrusion Detection system (IDS) [80] evaluated DP techniques in FL for enhancing IDS in industrial IoT (IIoT). The authors included assessing DP-enabled FL for IoT attack detection, analyzing various FL aggregation methods with DP, and providing a comprehensive performance analysis of different DP mechanisms. Using the ToN_IoT dataset, adapted for Non-IID data, the study shows that the Fed+ aggregation function significantly improves accuracy, particularly in Non-IID contexts, while maintaining acceptable accuracy levels with DP techniques

Cryptography-based methods can be categorized into two primary classes: encryption and Multi-Party Computation (MPC). Encryption methods predominantly leverage homomorphic encryption [81] to secure the training data through encryption. In contrast, MPC methods

Table 13
Comparison analysis with state-of-the-art approaches.

Approach	Key features	Strengths	Weaknesses/gaps	Our contribution
Differential Privacy (DP) [45,73–79]	– Adds noise to data or model updates. – Provides quantifiable privacy guarantees for privacy.	– Strong privacy guarantees. – Simple to implement in various ML algorithms. – Mathematical rigor provides confidence in privacy protection.	– Reduced accuracy due to noise – not be suitable for all types of data or models.	– Integrates DP principles with randomization. – Enhances privacy while maintaining accuracy.
Cryptography [85,85–93]	– Encrypts data and computations to ensure confidentiality. – Uses cryptographic protocols for secure aggregation of model updates.	– High security. – Prevents data leakage during processing. – Ensures confidentiality.	– High computational overhead. – Scalability issues, especially in large-scale systems.	– Utilizes TEEs to optimize cryptographic methods. – Reduces computational overhead and improves scalability.
Trusted Execution Environments (TEEs) [94–99]	– Secure enclaves for computation. – Protects data during processing.	– Strong security – Protects against hardware and software attacks. – Isolates sensitive computations.	– Limited memory. – Limitted computational power.	– Combines TEEs with randomization techniques. – Offers robust protection. – Ensures scalability.
Our approach	– Combines randomization and TEEs – Adds noise to secure enclaves – Secure aggregation	– Enhanced privacy and security – Scalable solution. – Robust against various attacks.	– Combines the strengths of DP, cryptography, and TEEs. – Mitigates the individual weaknesses of DP, cryptography, and TEEs by integrating their strengths.	– Provides a comprehensive solution. – Balances privacy, security, and performance.

employ protocols that facilitate collaborative computation over individual inputs while preserving their privacy. Notable MPC protocols include Yao's garbled circuit protocol [82], Shamir's protocol [83], and the Ben-Or-Goldwasser-Wigderson (BGW) protocol [84].

Encryption methods. In a collaborative learning system, Li et al. [85] used fully multi-key homomorphic encryption over encrypted data where datasets are encrypted with different keys. Graepel et al. [100] demonstrated that it is possible to train an ML model on encrypted data. Nevertheless, this approach can only work when the training algorithm can be expressed as a low-degree polynomial. Bost et al. [101] applied this technique in three classifiers: Hyperplane Decision, Naive Bayes, and Decision Trees. Gilad-Bachrach et al. [86] proposed CryptoNets designed to enable model training on encrypted input data. Li et al. [85] use fully multi-key homomorphic encryption as a solution to train an ML model over encrypted data in a collaborative learning system where datasets are encrypted with different keys. Ma et al. [87] proposed a privacy-preserving FL scheme called xMK-CKKS. This latter is suited for IoT scenarios where devices process data locally and share encrypted model updates with a server. The scheme uses a novel multi-key homomorphic encryption scheme requiring aggregated public key and decryption shares from all participants.

Multi-Party Computation (MPC). MPC is involved in distributed learning where non-colluding parties use encryption and oblivious transfer to privately train a model without revealing private data. Danner and Jelasity [89] proposed a secure sum protocol using a tree topology. SecureML [88] is an MPC-based protocol that trains a collaborative ML model using two non-collusive servers where participants' private data is shared between them. A secure multiparty aggregation protocol for FL proposed by Bonawitz et al. [90] in order to aggregate the parameters of the participants in an encrypted manner. Therefore, the server only gets the aggregated results. In [91], DisBezant is introduced by Xindi et al. as an algorithm for privacy-preserving, Byzantine-robust FL in IoT-enabled maritime transportation systems (MTS). It enables collaborative ship-based model training, safeguarding data, and countering attacks. The authors used techniques including assessing gradient credibility and secure aggregation via a central server while maintaining dataset diversity. Zhou et al. [92] proposed LEGO, a toolkit that enables secure and efficient machine learning using MPC. They showed that LEGO can lower the costs of communication, computation, and offline phases by applying associated triplets, linear activation functions, GPU-accelerated computing, and optimized offline protocols. These approaches' main goal is to prevent an honest but curious server from learning additional information apart from the aggregated results. Despite the fact that these approaches achieved a considerable performance in terms of privacy, they have many drawbacks either in communication complexity [102] or in computation time [90]. Gehlhar et al. [93] introduced a framework named SAFEFL, designed to assess FL methods that protect against privacy inference and poisoning attacks. SAFEFL leveraged MPC-based techniques to establish a distributed aggregator setup and featured a communicator interface linking PyTorch and MP-SPDZ [103]. The framework also conducted the implementation of diverse poisoning attacks, assessed various robust aggregation techniques, and evaluated the computational overhead of the private implementation of FLTrust [104] across multiple MPC protocols.

6.2.1. *Trusted Execution Entities (TEEs)*

Trusted execution environments (TEEs) can protect model parameters from attackers in the rich operating system execution environment (REE). However, due to the limited memory size of TEEs, most existing approaches that aim to secure deep learning models split the model training workflow into different parts. Ohrimenko et al. [94] used TEEs to prevent side-channel attacks on a model trained on a central server using SGX [47]. However, this approach is different from the FL approach since it requires training a model on the aggregated data of all participants inside the enclave. Chen et al. [95] assumed that customers have a TEE built into their system, which most participants do not. Mo et al. [96] proposed DarkneTZ to protect the last layer using TEEs but left the initial layers unsecured. However, it has been proven that the first layers can embed some unintended data into the model parameters [105], and this makes the model vulnerable to private data leaks. Mo et al. [97] proposed a layer-wise method where the server trains only the sensitive layers of the model within the TEEs, and then trains the remaining layers in the REE. However, this approach has some drawbacks in terms of training time and communication, as the layers of the model depend on each other, and each layer needs to wait for the previous layer's output. Natarajan et al. [98] introduced CHEX-MIX, a new protocol for privacy-preserving and Byzantine-robust machine learning inference in the cloud. The protocol combines homomorphic encryption (HE) and TEEs to protect the data of users and the models of providers from malicious attacks. The protocol also removes the need for attestation and provides confidentiality of the inference code. Recently, Islam et al. [99] proposed T-Slices, a framework for privacy-preserving and accurate machine learning inference on edge devices with limited memory. T-Slices uses ARM TrustZone to protect the data and model from malicious attacks and dynamically converts the model into slices that fit into the trusted memory. T-Slices does not modify or retrain the model and preserve the original inference accuracy.

According to what has been discussed, all the proposed options either attempt to secure all model parameters within TEEs, which is impractical and causes the model to converge slowly, or to secure only the final layer, which does not provide full data privacy. As a result, our technique is a compromise of the aforementioned approaches, as it provides complete data privacy and utilizes the sensitive parameters of the whole model (see Table 13).

7. Conclusion

In this paper, we have introduced the problem of Atout Tickets Learning (ATL), expressed as an ℓ_0-norm minimization. The aim is to identify neural network models' sensitive parameters, called atout tickets. We have derived a theoretical lower-bound on the number of atout tickets needed to maintain a model degradation at least by a given threshold ϵ. As an application of ATL, we have designed the Atout Ticket Protocol (ATP) for securing FL exchanges. ATP provides several atout tickets selection strategies. Experiments have shown that the choice of an atout ticket selection strategy is application-specific. In fact, for FL reconstruction attacks, the best heuristic to use is H_3 although H_4 uses less atout tickets. Experiments have also demonstrated that shallow networks are more vulnerable to reconstruction attacks. As an extension of this work, we plan to benchmark ATP against well-known distributed ML attacks and investigate other efficient selection strategies.

CRediT authorship contribution statement

Abdelhak Bouayad: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Formal analysis, Data curation, Conceptualization. **Mohammed Akallouch:** Writing – review & editing, Writing – original draft, Validation, Software. **Hamza Alami:** Supervision. **Ismail Berrada:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We used well known public dataset.

Appendix A. Additional experiments

To evaluate our approach comprehensively, we have included additional evaluation metrics, such as the Structural Similarity Index (SSI), Mean Absolute Error (MAE), and Peak Absolute Error (PAE). The results, presented in Table 14, were obtained using the CIFAR-10 dataset and a ConvNet architecture. These results confirm the consistency of our previously obtained outcomes and demonstrate that our approach effectively prevents IG attacks from extracting high-quality data. Instead, the extracted images are noisy and appear almost random, highlighting the robustness of our method in preserving data privacy.

Appendix B. Proof of the main results of the paper

B.1. Notations

- $f_\theta(X)$ is the nominal neural function and $f_{\theta\oplus\mathcal{M}}(X)$ is the mask neural function.
- C_θ is the f_θ neuron bound, i.e. $|y = f_\theta(X)| \leq C_\theta$.
- l_0 is the input layer, $\{l_1, \ldots, l_L\}$ are the hidden layers and l_{L+1} is the output layer.
- N_θ^l is the number of neurons in layer l, with $N_\theta = \sum_{l=0}^{L+1} N_\theta^l$.
- K_θ is the Lipschitz constant of the activation function.
- $\|\mathcal{M}^l\|_0$ is s the number of non-zero elements of a layer mask $\mathcal{M}^l$.
- $\|\mathcal{M}\|_0$ is the number of non-zero elements of a mask $\mathcal{M}$.
- $\|\mathcal{M}^l\|_\infty$ is the maximum absolute value of a layer mask $\mathcal{M}^l$.
- $\|\mathcal{M}\|_\infty$ is the maximum absolute value of a mask $\mathcal{M}$.
- $\|\theta^l\|_\infty$ is the maximum absolute value of θ^l.
- $\|\theta\|_\infty$ is the maximum absolute value of θ.

B.2. Proof of Theorem 1

Theorem 1. *Under the boundedness assumption, we have (K_θ is the Lipschitz constant of f_θ):*

$$|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| \leq C_\theta \times \Bigg(\|\mathcal{M}^{L+1}\|_\infty \times \|\mathcal{M}^{L+1}\|_0 + \sum_{i=1}^{L} (K_\theta^{L+1-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j)) \Bigg)$$

Proof. See Appendix B.2 $\square$

Proof.

$$\begin{aligned}
|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| &= |\sum_{j=1}^{N_\theta^1} (\theta_j^2 \oplus \mathcal{M}_j^2)\tilde{y}_j^{(1)} - \sum_{j=1}^{N_\theta^1} \theta_j^2 y_j^{(1)}| \\
&= |\sum_{j=1}^{N_\theta^1} \theta_j^2(\tilde{y}_j^{(1)} - y_j^{(1)}) + \sum_{j=1}^{N_\theta^1} \mathcal{M}_j^2 \tilde{y}_j^{(1)}| \\
&\leq |\sum_{j=1}^{N_\theta^1} \theta_j^2(\tilde{y}_j^{(1)} - y_j^{(1)})| + |\sum_{j=1}^{N_\theta^1} \mathcal{M}_j^2 \tilde{y}_j^{(1)}| \\
&\leq |\sum_{j=1}^{N_\theta^1} \theta_j^2(\varphi(\sum_{i=1}^{N_\theta^0}(\theta_{ji}^1 \oplus \mathcal{M}_{ji}^1)y_i^{(0)}) \\
&\quad - \varphi(\sum_{i=1}^{N_\theta^0} \theta_{ji}^1 y_i^{(0)}))| + \sum_{j=1}^{N_\theta^1} |\mathcal{M}_j^2 \tilde{y}_j^{(1)}| \\
&\leq \sum_{j=1}^{N_\theta^1} (|\theta_j^2| \times K_\theta \times |\sum_{i=1}^{N_\theta^0} \mathcal{M}_{ji}^1 y_i^{(0)}|) \\
&\quad + \sum_{j=1}^{N_\theta^1} |\mathcal{M}_j^2 \tilde{y}_j^{(1)}| \\
&\leq \sum_{j=1}^{N_\theta^1} (\|\theta^2\|_\infty \times K_\theta \times \sum_{i=1}^{N_\theta^0} |\mathcal{M}_{ji}^1 y_i^{(0)}|) \\
&\quad + \sum_{j=1}^{N_\theta^1} |\mathcal{M}_j^2 \tilde{y}_j^{(1)}| \\
&\leq K_\theta \times \|\theta^2\|_\infty \times \sum_{j=1}^{N_\theta^1} \sum_{i=1}^{N_\theta^0} |\mathcal{M}_{ji}^1 y_i^{(0)}|
\end{aligned}$$

Table 14
ATP resilience against IG attack on CIFAR-10 using ConvNet architecture.

Hypothesis	Layer	MSE	FMSE	MAE	PAE	PSNR	SSI
Baseline		0.00168	0.48777	0.53944	3.81428	3.30890	0.47310
H0	Conv+FC	0.01971	1.04407	0.81775	4.02650	−0.03650	0.03645
H1	Conv	0.01174	1.04180	0.82350	3.81712	−0.05343	0.03540
H2	Conv+FC	0.00248	0.69957	0.65729	3.96868	1.68438	0.27557
H3	Conv	0.00198	0.75710	0.69628	4.00043	1.29757	0.20034
	Conv1	0.59819	3.3589E−08	0.60266	3.74643	2.34644	0.23910
H4	Conv2	0.00209	0.56261	0.57914	4.01809	2.61847	0.30136
	Last Conv	0.00844	0.91565	0.74558	4.02596	0.48556	0.15982

$$
\begin{aligned}
&+ \sum_{j=1}^{N_\theta^1} |\mathcal{M}_j^2 \tilde{y}_j^{(1)}| \\
&\le K_\theta \times \|\theta^2\|_\infty \times \|\mathcal{M}^1\|_\infty \times \|\mathcal{M}^1\|_0 \\
&\times C_\theta + \|\mathcal{M}^2\|_\infty \times \|\mathcal{M}^2\|_0 \times C_\theta
\end{aligned}
$$

Let us prove it by induction.
Base case: $L = 1$. As $N_\theta^2 = 1$ (the output layer has only one neuron), and using the result of the previous theorem, we get:

$$
\begin{aligned}
|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| \le C_\theta \times \Big(&\|\mathcal{M}^2\|_\infty \times \|\mathcal{M}^2\|_0 \\
&+ K_\theta \times \|\theta^2\|_\infty \times \|\mathcal{M}^1\|_\infty \times \|\mathcal{M}^1\|_0 \Big)
\end{aligned}
$$

Therefore the inequality is true for $L = 1$.
Induction hypothesis We assume that the inequality is true for any FFNN with L layers. That means:

$$
\begin{aligned}
&|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| \le C_\theta \times \Big(\|\mathcal{M}^{L+1}\|_\infty \times \|\mathcal{M}^{L+1}\|_0 + \\
&\sum_{i=1}^{L} (K_\theta^{L+1-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j)) \Big)
\end{aligned}
$$

Note that, the last neuron of the last layer has non activation function in this assumption. *Induction step* Now, let us show that the inequality is true for $L + 1$.

$$
\begin{aligned}
|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| = &\,| \sum_{p=1}^{N_\theta^{L+1}} (\theta_p^{L+2} \oplus \mathcal{M}_p^{L+2}) \tilde{y}_p^{(L+1)} - \\
&\sum_{p=1}^{N_\theta^{L+1}} \theta_p^{L+2} y_p^{(L+1)} | \\
&= | \sum_{p=1}^{N_\theta^{L+1}} \theta_p^{L+2} (\tilde{y}_p^{(L+1)} - y_p^{(L+1)}) + \\
&\sum_{p=1}^{N_\theta^{L+1}} \mathcal{M}_p^{L+2} \tilde{y}_p^{(L+1)} | \\
&\le \|\theta^{L+2}\|_\infty \times \sum_{p=1}^{N_\theta^{L+1}} |\tilde{y}_p^{(L+1)} - y_p^{(L+1)}| \\
&+ \|\mathcal{M}^{L+2}\|_\infty \times \|\mathcal{M}^{L+2}\|_0 \times C_\theta
\end{aligned}
$$

For $p \in [1, N_\theta^{L+1}]$, $|\tilde{y}_p^{(L+1)} - y_p^{(L+1)}|$ is the output of a FFNN ${}^p\theta$ of L layers and similar to θ excepts for the last layer ${}^p\theta^{L+1}$. In fact:

- $C_{{}^p\theta} = C_\theta$ and $K_{{}^p\theta} = K_\theta$. Here we assume that all FFNN are bounded with same value.
- ${}^p\theta^i = \theta^i$ and ${}^p\mathcal{M}^i = \mathcal{M}^i$, for any layer $i < L + 1$.
- $N_\theta^{L+2} = 1$ and $(K_\theta)^0 = 1$.
- ${}^p\theta_{1j}^{L+1} = \theta_{pj}^{L+1}$ for any $j \in [1, N_\theta^L]$, $\|{}^p\theta^{L+1}\|_\infty \le \|\theta^{L+1}\|_\infty$, and $\sum_{p=1}^{N^{L+1}} N_{{}^p\theta}^{L+1} = N_\theta^{L+1}$.
- ${}^p\mathcal{M}_{1j}^{L+1} = \mathcal{M}_{pj}^{L+1}$ for any $j \in [1, N_\theta^L]$, $\|{}^p\mathcal{M}^{L+1}\|_\infty \le \|\mathcal{M}^{L+1}\|_\infty$ and $\sum_{p=1}^{N_\theta^{L+1}} \|{}^pM^{L+1}\|_0 = \|M^{L+1}\|_0$.

As neurons $\tilde{y}_p^{(L+1)}(X)$ and $y_p^{(L+1)}(X)$ have an activation function (not linear), then:

$$
|\tilde{y}_p^{(L+1)}(X) - y_p^{(L+1)}(X)| \le K_{{}^p\theta} \times |f_{{}^p\theta\oplus{}^p\mathcal{M}}(X) - f_{{}^p\theta}(X)|
$$

By applying the induction hypothesis for each FFNN $f_{{}^p\theta}$, we have:

$$
\begin{aligned}
|\tilde{y}_p^{(L+1)} - y_p^{(L+1)}| \le K_{{}^p\theta} \times \Big(&C_{{}^p\theta} \times \|{}^p\mathcal{M}^{L+1}\|_\infty \times \|{}^p\mathcal{M}^{L+1}\|_0 + \\
&C_{{}^p\theta} \times \sum_{i=1}^{L} \big(K_{{}^p\theta}^{L+1-i} \times \|{}^p\mathcal{M}^i\|_\infty \times \|{}^p\mathcal{M}^i\|_0 \times \\
&\prod_{j=i+1}^{L+1} (\|{}^p\theta^j\|_\infty \times N_{{}^p\theta}^j) \big) \Big) \\
\le K_{{}^p\theta} \times \Big(&C_{{}^p\theta} \times \|{}^p\mathcal{M}^{L+1}\|_\infty \times \|{}^p\mathcal{M}^{L+1}\|_0 + \\
&C_{{}^p\theta} \times \|{}^p\theta^{L+1}\|_\infty \times N_{{}^p\theta}^{L+1} \times \sum_{i=1}^{L} \big(K_{{}^p\theta}^{L+1-i} \\
&\times \|{}^p\mathcal{M}^i\|_\infty \times \|{}^p\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L} (\|{}^p\theta^j\|_\infty \times N_{{}^p\theta}^j) \big) \Big) \\
\le K_\theta \times \Big(&C_\theta \times \|{}^p\mathcal{M}^{L+1}\|_\infty \times \|{}^p\mathcal{M}^{L+1}\|_0 + \\
&C_\theta \times \|{}^p\theta^{L+1}\|_\infty \times \|{}^p\theta^{L+1}\|_0 \times \sum_{i=1}^{L} \big(K_\theta^{L+1-i} \\
&\times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L} (\|\theta^j\|_\infty \times N_\theta^j) \big) \Big) \\
\le K_\theta \times &C_\theta \times \|\mathcal{M}^{L+1}\|_\infty \times \|{}^p\mathcal{M}^{L+1}\|_0 + \\
&C_\theta \times \|\theta^{L+1}\|_\infty \times N_{{}^p\theta}^{L+1} \times \sum_{i=1}^{L} \big(K_\theta^{L+2-i} \\
&\times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L} (\|\theta^j\|_\infty \times N_\theta^j) \big)
\end{aligned}
$$

By making the sum:

$$
\begin{aligned}
\sum_{p=1}^{N_\theta^{L+1}} |\tilde{y}_p^{(L+1)} - y_p^{(L+1)}| \le \sum_{p=1}^{N_\theta^{L+1}} \Big(&K_\theta \times C_\theta \times \|\mathcal{M}^{L+1}\|_\infty \times \\
&\|{}^p\mathcal{M}^{L+1}\|_0 + C_\theta \times \|\theta^{L+1}\|_\infty \\
&\times N_{{}^p\theta}^{L+1} \times \sum_{i=1}^{L} \big(K_\theta^{L+2-i} \times \|\mathcal{M}^i\|_\infty \\
&\times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L} (\|\theta^j\|_\infty \times N_\theta^j) \big) \Big) \\
&\le K_\theta \times C_\theta \times \|\mathcal{M}^{L+1}\|_\infty \times \|\mathcal{M}^{L+1}\|_0 \\
&+ C_\theta \times \|\theta^{L+1}\|_\infty \times N_\theta^{L+1} \times \\
&\sum_{i=1}^{L} \big(K_\theta^{L+2-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \\
&\prod_{j=i+1}^{L} (\|\theta^j\|_\infty \times N_\theta^j) \big) \le K_\theta \times C_\theta
\end{aligned}
$$

$$\times \|\mathcal{M}^{L+1}\|_\infty \times \|\mathcal{M}^{L+1}\|_0 + C_\theta \times \sum_{i=1}^{L} \left(K_\theta^{L+2-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j) \right)$$

So, again

$$\begin{aligned}
|f_{\theta\oplus\mathcal{M}}(X) - f_\theta(X)| &\le \|\theta^{L+2}\|_\infty \times \sum_{p=1}^{N_\theta^{L+1}} |\tilde{y}_p^{(L+1)} - y_p^{(L+1)}| \\
&+ \|\mathcal{M}^{L+2}\|_\infty \times \|\mathcal{M}^{L+2}\|_0 \times C_\theta \\
&\le \|\theta^{L+2}\|_\infty \times \Big(K_\theta \times C_\theta \times \|\mathcal{M}^{L+1}\|_\infty \\
&\times \|\mathcal{M}^{L+1}\|_0 + C_\theta \times \sum_{i=1}^{L} \big(K_\theta^{L+2-i} \\
&\times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_\infty \\
&\times \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j) \big) \Big) \\
&+ \|\mathcal{M}^{L+2}\|_\infty \times \|\mathcal{M}^{L+2}\|_0 \times C_\theta \\
&\le \|\mathcal{M}^{L+2}\|_\infty \times \|\mathcal{M}^{L+2}\|_0 \times C_\theta \\
&+ K_\theta \times C_\theta \times \|\mathcal{M}^{L+1}\|_\infty \\
&\times \|\mathcal{M}^{L+1}\|_0 \times \|\theta^{L+2}\|_\infty \times N_\theta^{L+2} + C_\theta \times \\
&\sum_{i=1}^{L} \big(K_\theta^{L+2-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \\
&\prod_{j=i+1}^{L+2} (\|\theta^j\|_\infty \times N_\theta^j) \big) \le \|\mathcal{M}^{L+2}\|_\infty \times \|\mathcal{M}^{L+2}\|_0 \times C_\theta C_\theta \times \\
&\sum_{i=1}^{L+1} \big(K_\theta^{L+2-i} \times \|\mathcal{M}^i\|_\infty \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L+2} (\|\theta^j\|_\infty \times N_\theta^j) \big)
\end{aligned}$$

Therefore, the inequality is true for $L+1$. □

B.3. Proof of Corollary 1

Corollary 1 (*Lower-bound for ATL Problem*). *Given an $\epsilon > 0$, if $\mathcal{M}$ is a solution of the Problem 2 for a given f_θ, and $K_\theta \times \|\theta\|_\infty \times N_\theta \neq 1$ then:*

$$\frac{\epsilon \times \left(1 - K_\theta \times \|\theta\|_\infty \times N_\theta\right)}{C_\theta \times \left(1 - (K_\theta \times \|\theta\|_\infty \times N_\theta)^{L+1}\right)} \le \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0$$

Proof. See Appendix B.3 □

Proof.

$$\begin{aligned}
\epsilon &\le C_\theta \times \|\mathcal{M}^{L+1}\|_\infty \times \|\mathcal{M}^{L+1}\|_0 + C_\theta \times \sum_{i=1}^{L} \big(K_\theta^{L+1-i} \times \|\mathcal{M}^i\|_\infty \\
&\qquad \times \|\mathcal{M}^i\|_0 \times \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j) \big) \\
\epsilon &\le C_\theta \times \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0 \times \left(1 + \sum_{i=1}^{L} \big(K_\theta^{L+1-i} \times \prod_{j=i+1}^{L+1} (\|\theta^j\|_\infty \times N_\theta^j)\big)\right) \\
\epsilon &\le C_\theta \times \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0 \times \left(1 + \sum_{i=1}^{L} \big(K_\theta^{L+1-i} \times \prod_{j=i+1}^{L+1} (\|\theta\|_\infty \times N_\theta)\big)\right) \\
\epsilon &\le C_\theta \times \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0 \times \left(1 + \sum_{i=1}^{L} \big(K_\theta \times \|\theta\|_\infty \times N_\theta\big)^{L+1-i}\right) \\
\epsilon &\le C_\theta \times \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0 \times \left(\sum_{i=0}^{L} \big(K_\theta \times \|\theta\|_\infty \times N_\theta\big)^{i}\right) \\
\epsilon &\le \frac{C_\theta \times \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0 \times \left(1 - \big(K_\theta \times \|\theta\|_\infty \times N_\theta\big)^{L+1}\right)}{1 - K_\theta \times \|\theta\|_\infty \times N_\theta}
\end{aligned}$$

$$\frac{\epsilon \times \left(1 - K_\theta \times \|\theta\|_\infty \times N_\theta\right)}{C_\theta \times \left(1 - (K_\theta \times \|\theta\|_\infty \times N_\theta)^{L+1}\right)} \le \|\mathcal{M}\|_\infty \times \|\mathcal{M}\|_0 \quad \square$$

B.4. Proof of Theorem 2

Theorem 2. *At each round, the values computed by Algorithm 3 of the server corresponds to the average of locally updated client models.*

Proof.

$$\begin{aligned}
\theta_{t+1} &= AVG(\theta \oplus \mathcal{M})_t \oplus AVG(\theta \ddagger \mathcal{M})_t \\
&= \frac{1}{N}\sum_{j=1}^{N} \theta \oplus \mathcal{M}_{t,j} + \frac{1}{N}\sum_{j=1}^{N} \theta \ddagger \mathcal{M}_{t,j} \\
&= \frac{1}{N}\sum_{j=1}^{N} (\theta_{t,j} - \theta_{t,j} + \varepsilon_{t,j}) + \frac{1}{N}\sum_{j=1}^{N} (\theta_{t,j} - \varepsilon_{t,j}) \\
&= \frac{1}{N}\sum_{j=1}^{N} \varepsilon_{t,j} + \frac{1}{N}\sum_{j=1}^{N} \theta_{t,j} - \frac{1}{N}\sum_{j=1}^{N} \varepsilon_{t,j} \\
&= \frac{1}{N}\sum_{j=1}^{N} \theta_{t,j} \quad \square
\end{aligned}$$

References

[1] Kumari A, Tanwar S. Al-based peak load reduction approach for residential buildings using reinforcement learning. In: 2021 international conference on computing, communication, and intelligent systems. ICCCIS, 2021, p. 972–7. http://dx.doi.org/10.1109/ICCCIS51004.2021.9397241.

[2] Kumari A, Vekaria D, Gupta R, Tanwar S. Redills: Deep learning-based secure data analytic framework for smart grid systems. In: 2020 IEEE international conference on communications workshops (ICC workshops). 2020, p. 1–6. http://dx.doi.org/10.1109/ICCWorkshops49005.2020.9145448.

[3] Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T, Rellermeyer JS. A survey on distributed machine learning. 2020.

[4] Phong L, Aono Y, Hayashi T, Wang L, Moriai S. Privacy-preserving deep learning: Revisited and enhanced. 2017, p. 100–10.

[5] Zhu L, Liu Z, Han S. Deep leakage from gradients. 2019.

[6] Zhao B, Mopuri KR, Bilen H. Idlg: Improved deep leakage from gradients. 2020, arXiv preprint arXiv:2001.02610.

[7] Geiping J, Bauermeister H, Dröge H, Moeller M. Inverting gradients-how easy is it to break privacy in federated learning? 33, 2020, p. 16937–47,

[8] Ateniese G, Felici G, Mancini LV, Spognardi A, Villani A, Vitali D. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. 2013, arXiv:1306.4447.

[9] Hitaj B, Ateniese G, Perez-Cruz F. Deep models under the GAN: Information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. CCS '17, New York, NY, USA: Association for Computing Machinery; 2017, p. 603–18.

[10] Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy. SP, IEEE; 2017, p. 3–18.

[11] Salem A, Zhang Y, Humbert M, Berrang P, Fritz M, Backes M. ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models. 2018, arXiv:1806.01246.

[12] Hayes J, Melis L, Danezis G, Cristofaro ED. LOGAN: Membership inference attacks against generative models. 2018, arXiv:1705.07663.

[13] Melis L, Song C, Cristofaro ED, Shmatikov V. Exploiting unintended feature leakage in collaborative learning. 2018, arXiv:1805.04049.

[14] Wang Z, Song M, Zhang Z, Song Y, Wang Q, Qi H. Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE INFOCOM 2019-IEEE conference on computer communications. IEEE; 2019, p. 2512–20.

[15] Nasr M, Shokri R, Houmansadr A. 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, IEEE; 2019, p. 739–53.

[16] Sablayrolles A, Douze M, Schmid C, Ollivier Y, Jégou H. White-box vs. black-box: Bayes optimal strategies for membership inference. In: International conference on machine learning. PMLR; 2019, p. 5558–67.

[17] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for federated learning on user-held data. 2016, arXiv:1611.04482.

[18] Zhao L, Wang Q, Zou Q, Zhang Y, Chen Y. Privacy-preserving collaborative deep learning with unreliable participants. 2019.

[19] Zhao JC, Sharma A, Elkordy AR, Ezzeldin YH, Avestimehr S, Bagchi S. Secure aggregation in federated learning is not private: Leaking user data at large scale through model modification. 2023, http://dx.doi.org/10.48550/arXiv.2303.12233, CoRR arXiv:2303.12233, arXiv:2303.12233.

[20] Mohassel P, Zhang Y. Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy. SP, IEEE; 2017, p. 19–38.

[21] Louizos C, Welling M, Kingma DP. Learning sparse neural networks through L_0 regularization. 2017, arXiv preprint arXiv:1712.01312.

[22] Yuan G, Ghanem B. Sparsity constrained minimization via mathematical programming with equilibrium constraints. 2016, arXiv preprint arXiv:1608.04430.

[23] Elsken T, Metzen JH, Hutter F, et al. Neural architecture search: A survey. 2019.

[24] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. 2015, arXiv preprint arXiv:1503.02531.

[25] Sainath TN, Kingsbury B, Sindhwani V, Arisoy E, Ramabhadran B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE; 2013, p. 6655–9.

[26] Zhao Q, Sugiyama M, Yuan L, Cichocki A. Learning efficient tensor representations with ring-structured networks. In: ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing. ICASSP, IEEE; 2019, p. 8608–12.

[27] Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 2015, arXiv preprint arXiv:1510.00149.

[28] Jin S, Di S, Liang X, Tian J, Tao D, Cappello F. Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression. In: Proceedings of the 28th international symposium on high-performance parallel and distributed computing. 2019, p. 159–70.

[29] Plummer BA, Dryden N, Frost J, Hoefler T, Saenko K. Shapeshifter networks: Cross-layer parameter sharing for scalable and effective deep learning. 2020, arXiv preprint arXiv:2006.10598.

[30] Choi Y, El-Khamy M, Lee J. Towards the limit of network quantization. 2016, arXiv preprint arXiv:1612.01543.

[31] Reed R. Pruning algorithms-a survey. 1993.

[32] Evci U, Gale T, Menick J, Castro PS, Elsen E. Rigging the lottery: Making all tickets winners. In: International conference on machine learning. PMLR; 2020, p. 2943–52.

[33] Frankle J, Dziugaite GK, Roy D, Carbin M. Linear mode connectivity and the lottery ticket hypothesis. In: International conference on machine learning. PMLR; 2020, p. 3259–69.

[34] Frankle J, Carbin M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. 2018, arXiv preprint arXiv:1803.03635.

[35] Frankle J, Dziugaite GK, Roy DM, Carbin M. Pruning neural networks at initialization: Why are we missing the mark? 2020, arXiv preprint arXiv:2009.08576.

[36] Kusupati A, Ramanujan V, Somani R, Wortsman M, Jain P, Kakade S, Farhadi A. Soft threshold weight reparameterization for learnable sparsity. In: International conference on machine learning. PMLR; 2020, p. 5544–55.

[37] Ramanujan V, Wortsman M, Kembhavi A, Farhadi A, Rastegari M. What's hidden in a randomly weighted neural network? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 11893–902.

[38] Zhang Y, Lin M, Chao F, Wang Y, Wu Y, Huang F, Xu M, Tian Y, Ji R. Lottery jackpots exist in pre-trained models. 2021, arXiv preprint arXiv:2104.08700.

[39] Savarese P, Silva H, Maire M. Winning the lottery with continuous sparsification. 2019, arXiv preprint arXiv:1912.04427.

[40] Horton M, Jin Y, Farhadi A, Rastegari M. Layer-wise data-free CNN compression. 2020, arXiv preprint arXiv:2011.09058.

[41] de Jorge P, Sanyal A, Behl HS, Torr PH, Rogez G, Dokania PK. Progressive skeletonization: Trimming more fat from a network at initialization. 2020, arXiv preprint arXiv:2006.09081.

[42] Raihan MA, Aamodt TM. Sparse weight activation training. 2020, arXiv preprint arXiv:2001.01969.

[43] Hoefler T, Alistarh D, Ben-Nun T, Dryden N, Peste A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. 2021, arXiv preprint arXiv:2102.00554.

[44] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media; 2009.

[45] McMahan HB, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. 2017, arXiv:1602.05629.

[46] Dwork C. Differential privacy. In: International colloquium on automata, languages, and programming. Springer; 2006, p. 1–12.

[47] McKeen F, Alexandrovich I, Berenzon A, Rozas CV, Shafi H, Shanbhogue V, Savagaonkar UR. Innovative instructions and software model for isolated execution. In: Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy. HASP '13, New York, NY, USA: Association for Computing Machinery; 2013.

[48] Kaplan D, Powell J, Woller T. AMD memory encryption. White paper, 13, 2016.

[49] Gale T, Elsen E, Hooker S. The state of sparsity in deep neural networks. 2019, arXiv preprint arXiv:1902.09574.

[50] Wang Z, Song M, Zhang Z, Song Y, Wang Q, Qi H. Beyond inferring class representatives: User-level privacy leakage from federated learning. 2018.

[51] Lam M, Wei G-Y, Brooks D, Reddi VJ, Mitzenmacher M. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. 2021, arXiv:2106.06089.

[52] Lee J, Park S, Mo S, Ahn S, Shin J. A deeper look at the layerwise sparsity of magnitude-based pruning. 2020, arXiv preprint arXiv:2010.07611.

[53] Han S, Pool J, Tran J, Dally WJ. Learning both weights and connections for efficient neural networks. 2015, arXiv preprint arXiv:1506.02626.

[54] Morcos AS, Yu H, Paganini M, Tian Y. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. 2019, arXiv preprint arXiv:1906.02773.

[55] Mocanu DC, Mocanu E, Stone P, Nguyen PH, Gibescu M, Liotta A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. 2018.

[56] Hannun A, Guo C, van der Maaten L. Measuring data leakage in machine-learning models with Fisher information. 2021, arXiv:2102.11673.

[57] Li J, Sun J, Vuduc R. Hicoo: Hierarchical storage of sparse tensors. In: SC18: international conference for high performance computing, networking, storage and analysis. IEEE; 2018, p. 238–52.

[58] LeCun Y, Cortes C. MNIST handwritten digit database. 2010, http://yann.lecun.com/exdb/mnist/.

[59] Krizhevsky A, Nair V, Hinton G. Cifar-10 (canadian institute for advanced research). 2010.

[60] Huang GB, Ramesh M, Berg T, Learned-Miller E. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical report 07-49, Amherst: University of Massachusetts; 2007.

[61] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35.

[62] Pasquini D, Francati D, Ateniese G. Eluding secure aggregation in federated learning via model inconsistency. In: Yin H, Stavrou A, Cremers C, Shi E, editors. Proceedings of the 2022 ACM SIGSAC conference on computer and communications security, CCS 2022, los angeles, CA, USA, November 7-11, 2022. ACM; 2022, p. 2429–43. http://dx.doi.org/10.1145/3548606.3560557.

[63] Wen Y, Geiping J, Fowl L, Goldblum M, Goldstein T. Fishing for user data in large-batch federated learning via gradient magnification. In: Chaudhuri K, Jegelka S, Song L, Szepesvári C, Niu G, Sabato S, editors. International conference on machine learning, ICML 2022, 17-23 July 2022, baltimore, maryland, USA. Proceedings of machine learning research, vol. 162, PMLR; 2022, p. 23668–84, URL: https://proceedings.mlr.press/v162/wen22a.html.

[64] Boenisch F, Dziedzic A, Schuster R, Shamsabadi AS, Shumailov I, Papernot N. When the curious abandon honesty: Federated learning is not private. In: 8th IEEE European symposium on security and privacy, euroS&p 2023, delft, netherlands, July 3-7, 2023. IEEE; 2023, p. 175–99. http://dx.doi.org/10.1109/EuroSP57164.2023.00020.

[65] Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: 2009 IEEE computer society conference on computer vision and pattern recognition (CVPR 2009), 20-25 June 2009, miami, florida, USA. IEEE Computer Society; 2009, p. 248–55. http://dx.doi.org/10.1109/CVPR.2009.5206848.

[66] Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning word vectors for sentiment analysis. In: Lin D, Matsumoto Y, Mihalcea R, editors. The 49th annual meeting of the association for computational linguistics: human language technologies, proceedings of the conference, 19-24 June, 2011, portland, oregon, USA. The Association for Computer Linguistics; 2011, p. 142–50, URL: https://aclanthology.org/P11-1015/.

[67] Aly H, Al-Ali A, Al-Ali A, Malluhi QM. A blackbox model is all you need to breach privacy: Smart grid forecasting models as a use case. 2023, http://dx.doi.org/10.48550/arXiv.2309.01523, CoRR a2309.01523, arXiv:2309.01523.

[68] Gu Y, Bai Y. LDIA: label distribution inference attack against federated learning in edge computing. J Inf Secur Appl 2023;74:103475. http://dx.doi.org/10.1016/j.jisa.2023.103475, URL: https://doi.org/10.1016/j.jisa.2023.103475.

[69] Carlini N, Chien S, Nasr M, Song S, Terzis A, Tramèr F. Membership inference attacks from first principles. In: 43rd IEEE symposium on security and privacy, SP 2022, san francisco, CA, USA, May 22-26, 2022. IEEE; 2022, p. 1897–914. http://dx.doi.org/10.1109/SP46214.2022.9833649.

[70] Ye J, Maddi A, Murakonda SK, Bindschaedler V, Shokri R. Enhanced membership inference attacks against machine learning models. In: Yin H, Stavrou A, Cremers C, Shi E, editors. Proceedings of the 2022 ACM SIGSAC conference on computer and communications security, CCS 2022, los angeles, CA, USA, November 7-11, 2022. ACM; 2022, p. 3093–106. http://dx.doi.org/10.1145/3548606.3560675.

[71] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. 2016.

[72] Rubinstein BIP, Bartlett PL, Huang L, Taft N. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. 2009, arXiv:0911.5708.

[73] McMahan HB, Ramage D, Talwar K, Zhang L. Learning differentially private recurrent language models. 2018, arXiv:1710.06963.

[74] Pathak M, Rane S, Raj B. Multiparty differential privacy via aggregation of locally trained classifiers. 2010, p. 1876–84.

[75] Shokri R, Shmatikov V. Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. CCS '15, New York, NY, USA: Association for Computing Machinery; 2015, p. 1310–21.

[76] Sun J, Li A, Wang B, Yang H, Li H, Chen Y. Provable defense against privacy leakage in federated learning from representation perspective. 2020, arXiv preprint arXiv:2012.06043.

[77] Hamm J, Cao Y, Belkin M. Learning privately from multiparty data. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd international conference on machine learning. Proceedings of machine learning research, vol. 48, New York, New York, USA: PMLR; 2016, p. 555–63.

[78] Wei K, Li J, Ma C, Ding M, Chen W, Wu J, Tao M, Poor HV. Personalized federated learning with differential privacy and convergence guarantee. IEEE Trans Inf Forensics Secur 2023;18:4488–503. http://dx.doi.org/10.1109/TIFS.2023.3293417.

[79] Khalil M, Esseghir M, Boulahia LM. Privacy-preserving federated learning: An application for big data load forecast in buildings. Comput Secur 2023;131:103211. http://dx.doi.org/10.1016/j.cose.2023.103211, URL: https://www.sciencedirect.com/science/article/pii/S0167404823001219.

[80] Ruzafa-Alcázar P, Fernández-Saura P, Mármol-Campos E, González-Vidal A, Hernández-Ramos JL, Bernal-Bernabe J, Skarmeta AF. Intrusion detection based on privacy-preserving federated learning for the industrial IoT. IEEE Trans Ind Inf 2023;19(2):1145–54. http://dx.doi.org/10.1109/TII.2021.3126728.

[81] Gentry C. A fully homomorphic encryption scheme (Ph.D. thesis), Stanford University, USA; 2009, URL: https://searchworks.stanford.edu/view/8493082.

[82] Yao AC. How to generate and exchange secrets (extended abstract). In: 27th annual symposium on foundations of computer science, toronto, Canada, 27-29 October 1986. IEEE Computer Society; 1986, p. 162–7. http://dx.doi.org/10.1109/SFCS.1986.25.

[83] Shamir A. How to share a secret. Commun ACM 1979;22(11):612–3. http://dx.doi.org/10.1145/359168.359176.

[84] Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon J, editor. Proceedings of the 20th annual ACM symposium on theory of computing, May 2-4, 1988, chicago, illinois, USA. ACM; 1988, p. 1–10. http://dx.doi.org/10.1145/62212.62213.

[85] Li P, Li J, Huang Z, Li T, Gao C-Z, Yiu S-M, Chen K. Multi-key privacy-preserving deep learning in cloud computing. 74, 2017, p. 76–85,

[86] Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd international conference on machine learning. Proceedings of machine learning research, vol. 48, New York, New York, USA: PMLR; 2016, p. 201–10.

[87] Ma J, Naas S, Sigg S, Lyu X. Privacy-preserving federated learning based on multi-key homomorphic encryption. Int J Intell Syst 2022;37(9):5880–901. http://dx.doi.org/10.1002/int.22818, URL: https://doi.org/10.1002/int.22818.

[88] Mohassel P, Zhang Y. SecureML: A system for scalable privacy-preserving machine learning. 2017.

[89] Danner G, Jelasity M. Fully distributed privacy preserving mini-batch gradient descent learning. In: Proceedings of the 15th IFIP WG 6.1 international conference on distributed applications and interoperable systems - volume 9038. Berlin, Heidelberg: Springer-Verlag; 2015, p. 30–44.

[90] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. CCS '17, New York, NY, USA: Association for Computing Machinery; 2017, p. 1175–91.

[91] Ma X, Jiang Q, Shojafar M, Alazab M, Kumar S, Kumari S. DisBezant: Secure and robust federated learning against Byzantine attack in IoT-enabled MTS. IEEE Trans Intell Transp Syst 2023;24(2):2492–502. http://dx.doi.org/10.1109/TITS.2022.3152156, URL: https://doi.org/10.1109/TITS.2022.3152156.

[92] Zhou Z, Fu Q, Wei Q, Li Q. LEGO: A hybrid toolkit for efficient 2PC-based privacy-preserving machine learning. Comput Secur 2022;120:102782. http://dx.doi.org/10.1016/j.cose.2022.102782, URL: https://www.sciencedirect.com/science/article/pii/S0167404822001778.

[93] Gehlhar T, Marx F, Schneider T, Suresh A, Wehrle T, Yalame H. SAFEFL: MPC-friendly framework for private and robust federated learning. IACR Cryptol ePrint Arch 2023;555, URL: https://eprint.iacr.org/2023/555.

[94] Ohrimenko O, Schuster F, Fournet C, Mehta A, Nowozin S, Vaswani K, Costa M. Oblivious multi-party machine learning on trusted processors. In: Proceedings of the 25th USeNIX conference on security symposium. SEC '16, USA: USENIX Association; 2016, p. 619–36.

[95] Chen H, Fu C, Rouhani BD, Zhao J, Koushanfar F. DeepAttest: An end-to-end attestation framework for deep neural networks. In: Proceedings of the 46th international symposium on computer architecture. ISCA '19, New York, NY, USA: Association for Computing Machinery; 2019, p. 487–98.

[96] Mo F, Shamsabadi AS, Katevas K, Demetriou S, Leontiadis I, Cavallaro A, Haddadi H. DarkneTZ: towards model privacy at the edge using trusted execution environments. New York, NY, USA: Association for Computing Machinery; 2020, http://dx.doi.org/10.1145/3386901.3388946.

[97] Mo F, Haddadi H, Katevas K, Marin E, Perino D, Kourtellis N. PPFL: Privacy-preserving federated learning with trusted execution environments. 2021.

[98] Natarajan D, Dai W, Dreslinski RG. CHEX-MIX: combining homomorphic encryption with trusted execution environments for two-party oblivious inference in the cloud. IACR Cryptol ePrint Arch 2021;1603, URL: https://eprint.iacr.org/2021/1603.

[99] Islam MS, Zamani M, Kim CH, Khan L, Hamlen KW. Confidential execution of deep learning inference at the untrusted edge with ARM TrustZone. 2023, http://dx.doi.org/10.1145/3577923.3583648.

[100] Graepel T, Lauter K, Naehrig M. ML confidential: Machine learning on encrypted data. In: Proceedings of the 15th international conference on information security and cryptology. ICISC '12, Berlin, Heidelberg: Springer-Verlag; 2012, p. 1–21.

[101] Bost R, Popa R, Tu S, Goldwasser S. Machine learning classification over encrypted data. In: Network and distributed system security symposium. san diego: NDSS symposium; 2015.

[102] So J, Guler B, Avestimehr AS. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. 2021, arXiv:2002.04156.

[103] Keller M, Sun K. Secure quantized training for deep learning. In: Chaudhuri K, Jegelka S, Song L, Szepesvári C, Niu G, Sabato S, editors. International conference on machine learning, ICML 2022, 17-23 July 2022, baltimore, maryland, USA. Proceedings of machine learning research, vol. 162, PMLR; 2022, p. 10912–38, URL: https://proceedings.mlr.press/v162/keller22a.html.

[104] Cao X, Fang M, Liu J, Gong N. Fltrust: Byzantine-robust federated learning via trust bootstrapping. 2021, http://dx.doi.org/10.14722/ndss.2021.24434.

[105] Nasr M, Shokri R, Houmansadr A. 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, IEEE; 2019.