Protecting Data from all Parties: Combining FHE and DP in Federated Learning

Arnaud Grivet Sébert\textsuperscript{*} 1  Renaud Sirdey\textsuperscript{1}  Oana Stan\textsuperscript{1}

Cédric Gouy-Pailler\textsuperscript{1}

\textsuperscript{1}Institut LIST, CEA, Université Paris-Saclay, F-91120, Palaiseau, France

Abstract

This paper tackles the problem of ensuring training data privacy in a federated learning context. Relying on Fully Homomorphic Encryption (FHE) and Differential Privacy (DP), we propose a secure framework addressing an extended threat model with respect to privacy of the training data. Notably, the proposed framework protects the privacy of the training data from all participants, namely the training data owners and an aggregating server. In details, while homomorphic encryption blinds a semi-honest server at learning stage, differential privacy protects the data from semi-honest clients participating in the training process as well as curious end-users with black-box or white-box access to the trained model. This paper provides with new theoretical and practical results to enable these techniques to be effectively combined. In particular, by means of a novel stochastic quantization operator, we prove differential privacy guarantees in a context where the noise is quantified and bounded due to the use of homomorphic encryption. The paper is concluded by experiments which show the practicality of the entire framework in spite of these interferences in terms of both model quality (impacted by DP) and computational overheads (impacted by FHE).

Keywords — Federated learning, Differential privacy, Homomorphic encryption, Quantization

\textsuperscript{*}Contact author: arnaud.grivetsebert@cea.fr
1 Introduction

Machine learning techniques have become ubiquitous in almost all fields of industry and in our daily lives. The used algorithms, and especially the most popular ones in a lot of applications, namely neural networks, require massive amounts of data to get trained and reach the impressive accuracy that made their success. This huge need for data together with the omnipresence of the Internet and the boom of communication exchanges made data the "new oil" (Clive Humby, 2006).

In parallel, confidentiality has raised greater and greater concerns as evidenced by the new regulations on data privacy (e.g. GDPR [1] in the EU) and, while all kinds of data might be considered as private, some fields, like healthcare (e.g. HIPAA [2] in the USA) or military applications, are especially sensitive to privacy breaches.

In this context, it has been shown that, once trained, a machine learning model may indirectly release some information about its training data. Many researchers have exhibited attacks on these models, assuming that the adversary has access to the parameters of the model (white-box access) or even only to the output of the inference on some queries she made to the model (black-box access) [3, 4, 5]. These attacks are even more likely with the emergence of machine learning as a service [6] in the recent years.

To deal with these issues, the most popular notion of data privacy is differential privacy (DP) which was introduced by Dwork et al. [7] and quantifies the amount of information leaked by the output of a mechanism about its input. This notion of privacy implies that the considered mechanism is probabilistic, which is traditionally achieved by applying carefully parameterized random noise to a deterministic mechanism. Since 2006, DP has been widely used in machine learning applications (see for example [8]).

Cryptography is another field that helps providing protection to sensitive data against curious adversaries, in particular at training time. Specifically, homomorphic encryption (HE) [9, 10] or multi-party computation [11, 12] allow that a server performs computations without seeing the input data, and in the case of HE, the output either. Additionally, HE can in some cases be associated to tools such as verifiable computing to bring additional computation integrity guarantees, including in the context of federated learning [13].

In 2016, McMahan et al. proposed a new paradigm of collaborative learning that they called federated learning (FL) [14]. Along with the reduction of communication load and the parallelism it allows, a claimed key advantage is the protection of data due to the fact that each client keeps its own data locally. However, although FL gives some protection to the data with regard to the server, it gives rise to a new type of potential adversaries - the other clients. Several attacks that take advantage of this new threat were proposed in the literature [15, 16].

The contribution of this paper is the design of a complete FL framework immune to a wide spectrum of threats. The above-mentioned indirect attacks on the training data are addressed via DP, either they come from participants during the training or from the end-users of the model. Another potential adversary is obviously the server. To hide the data from it, the clients send encrypted information to the server which will do the necessary computations in the encrypted-domain, without seeing either the sent information or the result of its computations, thanks to the homomorphic properties of the used cryptosystems. In order to consistently articulate DP and FHE we introduce a (to the best of our knowledge) new stochastic quantization operator based on the Poisson distribution. This, thus, allows us to prove differential privacy...
guarantees in a context where the noise is quantified and bounded due to the use of homomorphic encryption (and related techniques).

An interesting application scenario of this work concerns the medical field. In this context, we may consider several hospitals that own medical data from their patients and wish to collaborate in order to train a global model that would detect a certain disease. Classically, the patients’ data are sensitive and the hospitals do not want to share them to the other hospitals. A solution is to use an aggregation server (e.g. from an institutional entity) but the hospitals do not want to give it their data either. Note that the parameters we used (see Section 5) make our solution lie between cross-silo and cross-device FL, which can be compatible with the numbers of medical facilities in a reasonably large country.

The paper is organized as follows. In Section 2, a quick review of the literature on the issues of data privacy in FL context follows this introduction. Then, Section 3 provides the technical prerequisites necessary to understand our method, that we thoroughly explain in Section 4. The results of the experiments that we ran to illustrate the feasibility of our solution are presented in Section 5, before concluding remarks and perspectives for further work (Section 6).

2 Related work

2.1 Differentially private federated learning

Due to the additional threats from the other clients, DP has been quite popular in collaborative and especially FL frameworks.

For instance, McMahan et al. [17] trained a next word prediction algorithm in a federated way, using the text data from users’ smartphones which were protected by DP. Nevertheless, this work is hardly comparable to ours because the task for which the model is trained is very different from our targeted classification tasks (e.g. FEMNIST in our experiments), inducing very different learning parameters and thus a quite different privacy challenge.

Closer to our work is the one from [18] which presents a differentially private FL process tested on MNIST, yet a notably easier task than FEMNIST. Shokri et al. [19] also experimented a differentially private distributed learning setting - similar to a FL setting due to the use of distributed stochastic gradient descent - on MNIST, and SVHN, another famous image dataset.

Sabater et al., in [20], ensures privacy and utility of their distributed learning setting as long as each participant communicates with only a logarithmic number of other participants.

Quite importantly for our DP analysis, Abadi et al. [21] introduced the moments accountant method that enables to keep track of the privacy cost more tightly than the traditional composition theorem when many calls to the database are done, typically while training a deep neural network, not necessarily in a distributed way.

As far as utility is concerned, contrary to some of the cited works, we did not conduct a theoretical analysis of the utility of our differentially private mechanism and let our experiments demonstrate empirically the loss in accuracy.
2.2 Homomorphic encryption and federated learning

Most of the works applying HE to machine learning models focus on the inference stage (CryptoNets [22], TAPAS [23], NED [24]) and not on the training stage.

The first papers on privacy-preserving machine learning training concentrated on a centralised setting where all the data are outsourced and where, besides, the models are only linear [25, 26]. When it comes to non-linear models, the few approaches that ran a complete centralised training of neural networks on homomorphic encrypted data have impractical performances or huge cryptographic parameters ([27]).

While some authors propose solutions in the case of multi-servers either for clustering or regression, many methods employing homomorphic encryption have been recently proposed for a collaborative learning task with no central server. They mostly apply on linear models [28, 29] and, more recently, Sav et al. focused on neural networks [30].

As far as FL is concerned, only a few recent papers propose the use of homomorphic encryption to protect the clients’ data from the server ([9, 10, 31]), the first two cited works being only theoretical.

2.3 Other multi-party computation approaches for machine learning

More general than HE, multi-party computation protocols allow several agents to collaborate and compute a function on their data such that each agent knows no more than its own input and, if requested, the output, and learns nothing about the other agents’ input. The combination of the high-communication costs of the multi-party computation and the inherent distributed nature of FL makes FL methods secured by multi-party computation (e.g. [11, 12]) difficult to implement efficiently. Among these approaches, [32] interestingly makes use of DP techniques to further protect the private data.

2.4 Quantization and differential privacy

A great focus and key contribution of our paper deals with the interference between our two defensive tools, namely DP and HE. The biggest issue induced by this interference is the restriction on the number of bits representing a message to be encrypted. This implies that the noised updates must belong to a bounded and discrete range before they are encrypted and sent to the server.

For potentially other reasons than a necessary encryption of the data, some authors have studied the possibility of making a machine learning mechanism differentially private with a discrete noise.

The authors of [33] want to propose a secure and communication-efficient distributed learning framework and perform the DP analysis of their learning mechanism using a binomial noise because the effect of quantization on the Gaussian mechanism is unclear, especially after aggregation in the distributed noise generation case. The analysis is quite involved and does provide DP bounds for the multidimensional binomial mechanism, yet only for one round of learning. Indeed, the moments accountant method is not easily applicable to the binomial distribution.

In [34], Koskela et al. present a privacy accountant for discrete-valued mechanisms for non-adaptive queries using privacy loss distribution formalism and Fast Fourier Transform. In particular, they give DP guarantees for the binomial mechanism in one
dimension and extend them in the multidimensional case but with quite demanding constraints that compel them to brutally approximate the gradients by their sign in their experiment.

Cannone et al. [35] introduced the discrete Gaussian mechanism and studied its DP guarantees that scale well with composition, even in the multivariate case. Nevertheless, contrary to binomial noise, discrete Gaussian noise is not bounded as required for our framework. Besides, and more critically, the discrete Gaussian distribution is not stable by addition, thus precluding its use in a context of distributively generated noise that a collaborative learning task with untrusted server requires (see Section 4.1).

3 Preliminaries

3.1 Federated learning

FL is a decentralised framework that enables multiple agents, called clients, to collaboratively train a shared global model under the orchestration of a central server while keeping the training data localised on the client devices. After a common (server-side) arbitrary initialisation of the global model, the FL process consists of successive rounds of communication between the server and the clients.

At the beginning of each round, the server selects a subset of clients to take part in training for this round, we call these particular clients the participants. The server sends the current global model to the participants and each of them trains the model locally using its own data. The participants then communicate only the updated parameters or the updates themselves (depending on the setting) back to the server. Finally, the server aggregates these parameters/updates before accumulating them into the global model thereby concluding the round.

The most common approach to optimisation for FL is the Federated Averaging algorithm [36] where each participant runs several epochs of minibatch stochastic gradient descent (SGD) minimising a local loss function. The central server then performs a weighted averaging of the updated local models to form the updated global model. The weight associated to a participant in the average is generally the fraction of training samples owned by the participant. Algorithm 1 gives the pseudocode of the Federated Averaging algorithm version whereby the participants send the updates rather than the updated parameters. This is the version we use in our work because it is easier to constrain the norm of the updates (necessary for privacy reasons) than the norm of the parameters themselves. The notations of Algorithm 1 will be used throughout the paper.

By avoiding the need to send all data to a central server, FL helps to protect the data privacy and reduces communication costs. As a result, FL applies best in situations where data are privacy-sensitive or large in such a way that it is undesirable or infeasible to transmit them to the server.

3.2 Differential privacy

Differential privacy [4] is a gold standard concept in privacy-preserving data analysis. It provides a guarantee that under a reasonable privacy budget ($\epsilon, \delta$) two adjacent databases produce statistically indistinguishable results. The notion of adjacency is always symmetric but varies from an author to another. In our FL framework, the difference between the updated parameters and the old ones
Algorithm 1 Federated Averaging

M is the total number of clients; K is the number of participants per round t; the selected participants are indexed by \( k \) with \( D_k \) the training set of data points on participant \( k \) and \( n_k = |D_k| \); B is the local minibatch size; E is the number of local epochs; \( \eta \) is the learning rate; \( w \) are model parameters and \( L \) is the local loss function.

Server executes:

initialize \( w_0 \)

for each round \( t \) do

\( K_t \leftarrow \) random set of \( K \leq M \) clients

for each client \( k \in K_t \) in parallel do

\((u^k_{t+1}, n_k) \leftarrow \text{ClientUpdate}(k, w_t)\)

\( w_{t+1} \leftarrow w_t + \sum_{k=1}^{K} \frac{n_k}{n} u^k_{t+1} \) where \( n = \sum_{k=1}^{K} n_k \)

ClientUpdate\((k, w)\):

initialize \( w_k = w \)

\( B \leftarrow \) split \( n_k \) samples of \( D_k \) into batches of size \( B \)

for each round epoch from 1 to \( E \) do

for each batch \( b \in B \) do

\( w_k \leftarrow w_k - \eta \nabla L(w_k; b) \)

return \((w_k - w, n_k)\)

term database denotes the concatenation of all the databases of the clients. Two databases are said adjacent if they have the same number of clients and differ on a single client, all the other clients remaining unchanged. Yet, the differing clients may have totally different data, making our notion of adjacency quite conservative.

Definition 1

A randomised mechanism \( \mathcal{M} \) with output range \( \mathcal{R} \) satisfies \((\epsilon, \delta)\)-differential privacy if, for any two adjacent databases \( d, d' \) and for any subset of outputs \( S \subset \mathcal{R} \), one has

\[
P[\mathcal{M}(d) \in S] \leq e^\epsilon P[\mathcal{M}(d') \in S] + \delta.
\]

A fundamental notion in DP is the sensitivity of a mechanism, defined below. When adjacency means differing by one sample (resp. client), the sensitivity measures the influence of any single sample (resp. client) - and thus typically the adversary’s target - on the output of the mechanism.

Definition 2

Let \( \mathcal{M} \) be a randomised mechanism. Given a norm \( \| \cdot \| \), the \( \| \cdot \| \)-sensitivity of \( \mathcal{M} \) is

\[
S = \max_{d, d' \text{ adjacent}} \| \mathcal{M}(d) - \mathcal{M}(d') \|
\]

where the maximum is taken over all pairs of adjacent databases.

One of the most famous and widely used differentially private mechanism is the Gaussian mechanism of parameter \( \sigma \in \mathbb{R}_+^* \), which simply adds a noise following \( \mathcal{N}(0, \sigma^2) \), i.e. the normal law of mean 0 and standard deviation \( \sigma \), to all the components of the output.
A fundamental property of differential privacy is its immunity to post-processing, as stated in Proposition 1.

**Proposition 1 ([37])** Let $M$ be a randomized algorithm, with output range $R$, that is $(\epsilon, \delta)$-differentially private. Let $f: R \rightarrow R'$ be an arbitrary randomized mapping. Then $f \circ M$ is $(\epsilon, \delta)$-differentially private.

**Definition 3** Let $M$ be a randomised mechanism with output range $R$ and $(d,d')$ a pair of adjacent databases. Let $aux$ denote an auxiliary input. For any $o \in R$, the privacy loss at $o$ is defined as

$$c(o; M, aux, d, d') := \log \frac{P[M(aux, d) = o]}{P[M(aux, d') = o]}.$$  

We define the privacy loss random variable $C(M, aux, d, d')$ as

$$C(M, aux, d, d') := c(M(d); M, aux, d, d')$$

i.e. the random variable defined by evaluating the privacy loss at an outcome sampled from $M(d)$.

In order to determine the privacy cost $(\epsilon, \delta)$ of our protocol, we use a traditional two-fold approach. First of all, we determine the privacy loss per query and, in a second step, we compose the privacy losses of all queries to get the overall loss. The classical composition theorem (see e.g. [37]) states that the guarantees $\epsilon$ of sequential queries add up. Nevertheless, training a deep neural network requires a large amount of calls to the databases, precluding the use of this classical composition. Therefore, to obtain reasonable DP guarantees, we need to keep track of the privacy loss with a more refined tool, namely the moments accountant [21] that we introduce here.

**Definition 4** With the same notations as above, for any $l \in \mathbb{R}^*_+$, any auxiliary input $aux$ and any pair of adjacent databases $(d, d')$,

$$\alpha_M(l; aux, d, d') := \log (\mathbb{E} [\exp(lC(M, aux, d, d'))])$$

is the moment generating function of the privacy loss random variable.

The moments accountant is defined for any $l \in \mathbb{R}^*_+$ as

$$\alpha_M(l) := \max_{aux,d,d'} \alpha_M(l; aux, d, d')$$

where the maximum is taken over any auxiliary input $aux$ and any pair of adjacent databases $(d,d')$.

The moments accountant allows for a new DP composition that gives far better results than the traditional one in practice.

**Theorem 1 ([21])** Let $p \in \mathbb{N}^*$. Let us consider a mechanism $M$ defined on a set $D$ that consists of a sequence of adaptive mechanisms $M_1, \ldots, M_p$ where, for any $i \in \{1, \ldots, p\}$, $M_i: \prod_{j=1}^{i-1} R_j \times D \rightarrow R_i$. Then, for any $l \in \mathbb{R}^*_+$,

$$\alpha_M(l) \leq \sum_{i=1}^{p} \alpha_{M_i}(l).$$
Finally, parameter $\delta$ being chosen, the privacy guarantee is derived from the overall moments accountant applying the tail bound property, stated in Theorem 2 from [21].

**Theorem 2 ([21])** For any $\epsilon \in \mathbb{R}^*_+$, the mechanism $\mathcal{M}$ is $(\epsilon, \delta)$-differentially private for

$$\delta = \min_{l \in \mathbb{N}^*} \exp(\alpha_{\lambda}(l) - l\epsilon).$$

### 3.3 Homomorphic Encryption

A HE system consists in two algorithms, the encryption $\text{Enc}$ from the clear domain to the encrypted domain and the decryption $\text{Dec}$ from the encrypted domain to the clear domain.

Any (decent) HE scheme possesses the semantic security property meaning that, given $\text{Enc}(m)$ and polynomially many pairs $(m_i, \text{Enc}(m_i))$ it is hard\(^2\) to gain any information on $m$ with a significant advantage over guessing. Most importantly, a HE scheme offers two other operators $\oplus$ and $\otimes$ where

- $\text{Enc}(m_1) \oplus \text{Enc}(m_2) = \text{Enc}(m_1 + m_2) \in \Omega$
- $\text{Enc}(m_1) \otimes \text{Enc}(m_2) = \text{Enc}(m_1m_2) \in \Omega$

This shows that HE schemes allow to perform computations directly over encrypted data without decrypting them first.

When these two operators are supported without restriction by a homomorphic scheme, it is said to be a Fully Homomorphic Encryption (FHE) scheme. A FHE whose clear domain is equal to $\mathbb{Z}_2$ is Turing-complete and, as such, is in principle sufficient to perform any computation in the encrypted domain with a computational overhead depending on the security target. In practice, though, the $\oplus$ and $\otimes$ are much more computationally costly than their clear domain counterparts, which has led to the development of several approaches to HE schemes design each with their pros and cons.

#### 3.3.1 Somewhat Homomorphic Encryption (SHE)

SHE schemes, such as BGV [38] or BFV [39], provide both operators but with several constraints. Indeed, in these cryptosystems the $\otimes$ operator is much more costly than the $\oplus$ operator and the cost of the former strongly depends on the multiplicative depth of the calculation, that is the maximum number of multiplications that have to be chained (although this depth can be optimised [40]). Interestingly, most SHE schemes offer a **batching** capability by which multiple cleartexts can be packed in one ciphertext resulting in (quite massively) parallel homomorphic operations i.e.,

$$\text{Enc}(m_1, \ldots, m_\kappa) \oplus \text{Enc}(m'_1, \ldots, m'_\kappa) = \text{Enc}(m_1 + m'_1, \ldots, m_\kappa + m'_\kappa)$$

(and similarly so for $\otimes$). Typically, several hundreds such slots are available which often allows to significantly speed up encrypted-domain calculations.

---

\(^2\)“Hard” means that it requires solving a reference (conjectured) computationally hard problem on which the security of the cryptosystem hence depends.
3.3.2 Fully Homomorphic Encryption (FHE)

Fully homomorphic encryption schemes offer both the $\oplus$ and $\otimes$ operators without restrictions on multiplicative depth. At the time of writing, only the FHE-over-the-torus approach, instantiated in the TFHE cryptosystem [41], offers practical performances. In this cryptosystem, $\oplus$ and $\otimes$ have the same constant cost. On the downside, TFHE offers no batching capabilities.

The bottomline of xHE calculations practice is to use the most appropriate cryptosystem for the problem at hand. Low (multiplicative-)depth calculations (say 5 and below) are generally performed more efficiently by means of SHEs (BFV is used in the experimental results in this paper) while higher (multiplicative-)depth ones (say above 5) requires resorting to the full-blown FHE machinery of TFHE. Furthermore, to get the best of all worlds, the TFHE scheme is often hybridised with SHE by means of operators allowing to homomorphically switch among several ciphertext formats to perform each part of calculation with the most appropriate scheme (see e.g. [42]).

Additionally, it should be emphasised that the natural threat model against FHE is limited to honest-but-curious adversaries. Such an adversary, i.e. the server which performs the encrypted-domain calculations, properly executes the task it is in charge of but attempts by all possible passive means to retrieve information about the private data values. Therefore, in a machine learning context, it may make use of any data it legitimately has access to, directly or indirectly by performing some (polynomial-time) computations on them, in order to retrieve some information about the private data, yet without harming the learning (or inference) process.

4 A fully private federated learning framework

Our secure FL framework is illustrated in Figure 1. The process at each iteration is the following:

- the server sends the current model parameters to the participants
- each participant decrypts the parameters thanks to the decryption key
- each participant performs a gradient descent of the loss function using its local data
- each participant applies on the obtained updates the successive transformations required by DP and HE (clipping, noising and quantization)
- each participant encrypts the transformed updates and sends them to server
- the server aggregates (computes the mean of) the gradients in the secret domain

The two next subsections focus on the two security tools of our framework - homomorphic encryption and differential privacy.

4.1 Differential privacy protects the data against the clients and the end-users

When one wants to protect the training data by differential privacy in a collaborative learning process, the assumption of the honest-but-curious server might be mitigated and the server may not be trusted to generate the noise, because it may not be capable of sampling it or, even more simply, because it may communicate the noise to the end-users, thus annihilating the differential privacy guarantees. A common practice in
that case is to make the participants generate the noise in a distributed way \[20, 43\]. This is especially practical when one wants the resulting noise to follow a Gaussian distribution - which is one of the most popular distribution in DP - , since the Gaussian distribution is stable by addition. The participants simply need to generate a Gaussian noise with the well-chosen variance.

Furthermore, charging the participants of the noising makes our framework easily extendable to an even more secure framework where the integrity of the server computations is guaranteed by Verifiable Computing techniques \[44\]. Indeed, this would not be possible with a centralised noise generation since a participant would need to generate a tag depending on the noise to ensure its correct addition on the server. This would compel us to reveal the noise to a particular participant, thus breaking the differential privacy guarantees from its point of view. By decentralising the noising operation, we do not need to prove it was correctly done anymore since it is not the server who is in charge of it.

DP in a FL context still requires some adaptations on the non-secure process:

- clipping the updates in L2-norm with clipping bound \( S \) (substituting participant \( k \)'s vector of updates \( u^k \) by \( \min \left( 1, \frac{S}{||u^k||_2} \right) u^k \) to bound the sensitivity (since unclipped updates and thus unbounded sensitivity is incompatible with any DP guarantee)
- adding noise to the gradients (e.g. Gaussian noise)
- fix all the coefficients of the mean to \( \frac{1}{m} \), independently of the size of the participant’s dataset to bound the sensitivity
4.1.1 Problem of the limited number of bits and naive approach

In our scenario, the information sent by the participants to the server is encrypted via HE. Since HE is quite computationally heavy, we cannot represent the messages sent to the server with too many bits. This means that, contrary to the usual case where the noise is represented by a double-precision float (i.e. finite but very fine precision) and where we make the approximation that it perfectly follows the wanted distribution, we here have to take into account the finite number of bits of the message. This translates to bounds on the noise and a discretization (or quantization) of this noise (and of the updates themselves but this does not constitute an issue for the DP analysis). Therefore, the aggregated noise is not a Gaussian noise any more, but a sum of noises that follow bounded quantized Gaussian distributions. Unfortunately, the distribution of a sum of bounded quantized Gaussian variables does not have a simple expression (in particular, it is not a bounded quantized Gaussian distribution).

A first approach would be to try to compare the complicated distribution of a sum of bounded quantized Gaussian variables with the one of a sum of perfect Gaussian variables i.e. a Gaussian variable. Indeed, if the bounds are high enough and the quantization scale is fine enough, the sum of bounded quantized Gaussian variables should intuitively be close to a Gaussian variable. The final privacy cost should be the sum of the one of a classical Gaussian mechanism and a hopefully small additional privacy cost due to the approximation. Nevertheless, this analysis is quite involved and may result in overestimated DP bounds.

We will now show that, under natural practical assumptions and with the use of a novel specific quantization function, we may significantly simplify the DP analysis, with no privacy loss compared to the Gaussian mechanism.

4.1.2 Poisson quantization

We here propose a new probabilistic quantization operator that commutes with the aggregation operator and is therefore harmless for the DP guarantees of the mechanism.

In the following, $P(\lambda)$ denotes the Poisson law of parameter $\lambda \in \mathbb{R}_+^*$ whose support is $\mathbb{N}$ and whose probability mass function is $k \in \mathbb{N} \mapsto \frac{\lambda^k}{k!} e^{-\lambda}$.

We fix the quantization scale $s \in \mathbb{R}_+^*$ and the dimension $d \in \mathbb{N}^*$ of the problem (the number of parameters of the model in our case).

**Definition 5** Let $\mu \in s\mathbb{Z}$. We define the probabilistic function

$$Q_{s,\mu} : x \in [\mu; +\infty[ \rightarrow sY + \mu$$

where $Y \sim P \left( \frac{x-\mu}{s} \right)$. We call it the Poisson quantization of scale $s$ and offset $\mu$.

Similarly, we define

$$Q_{s,\mu} : x = \left( x^{(i)} \right)_{i \in [1;d]} \in [\mu; +\infty[^d \rightarrow \left( Q_{s,\mu} \left( x^{(i)} \right) \right)_{i \in [1;d]}.$$  

Given $\mu \in s\mathbb{Z}$, for all $x \in [\mu; +\infty[^d$, $Q_{s,\mu} (x) \in (s\mathbb{Z})^d$ and its mean is equal to $x$ so we can actually consider the Poisson quantization as a function of probabilistic quantization.

\footnote{Commutativity must be understood in a large sense, as the offset parameter of the quantization changes depending on the order of the operators.}
Proposition 2 shows that the Poisson quantization on the terms of a sum can be considered as a post-processing on the sum.

**Proposition 2**

Let \( m \in \mathbb{N}^* \), \( x_1, \ldots, x_m \in \mathbb{R}^d \). Let \( \mu \in \mathbb{Z} \) such that \( \mu < \min \{ x_i | i \in [1; m] \} \).

\[ Q_{s,\mu} \left( \sum_{i=m}^{\infty} x_i \right) \] has the same distribution as \[ \sum_{i=m}^{\infty} Q_{s,\mu} (x_i) \].

**Proof.**

\[ \sum_{i=m}^{\infty} Q_{s,\mu} (x_i) \sim \sum_{i=m}^{\infty} (sY_i + \mu) = s \sum_{i=m}^{\infty} Y_i + m\mu \] where, for all \( i \in [1; m] \), \( Y_i \sim P \left( \frac{x_i - \mu}{s} \right) \). By stability of the Poisson law by addition, we know that \( \sum_{i=m}^{\infty} Y_i \sim P \left( \sum_{i=m}^{\infty} x_i - m\mu \right) \). We then directly get the result.

Proposition 2 together with Proposition 1 enables us to conclude that the Poisson quantization has no influence on the DP guarantees.

Note that, since Poisson quantization is probabilistic, it may harm the accuracy of the global model. Given \( \mu \in \mathbb{Z} \) and \( x \in [\mu; +\infty[^d \), the variance of \( Q_{s,\mu} (x) \) is \( s^2 x - \mu s = s(x - \mu) \). For a small enough \( s \), this variance is very small since \( x \) is bounded, and the impact on the accuracy is very mild as it is shown by our experiments (Section 5).

An important point to notice is that Poisson quantization implies that the quantities to round have an a priori common lower bound (otherwise the sum of the rounded quantities may depend on these quantities and not only on their sum). In our case, these quantities are the noised updates. The updates are already bounded by the clipping: as we will see in Section 4.1.3, this clipping constrains the L2-norm of the updates (considered as vectors of the updates of all the parameters) and thus it also constrains the absolute value of each parameter update. As for the noises, the argument of Section 4.1.3 allows us to consider that the noises have a common lower bound.

### 4.1.3 Ignoring the boundedness of the noises

The most common algorithms to sample from a Gaussian distribution are Box-Muller transform in its Cartesian and polar forms ([45, 46]) and the so-called ziggurat algorithm ([47]). As the following shows, it is easy to see that all these algorithms actually generate values whose range have bounds which are way smaller than the range of double-precision floats.

**Box-Muller transform (Cartesian form):** The Cartesian form of the Box-Muller transform samples two independent uniform random variables \( U_1 \) and \( U_2 \) in \([0; 1]\). The random variables

\[ Z_1 = \sqrt{-2\log(U_1)} \cos(2\pi U_2) \]

and

\[ Z_2 = \sqrt{-2\log(U_1)} \sin(2\pi U_2) \]

are independent and follow a standard normal distribution (standard deviation 1 and mean 0).

Since the function \( \cos \) and \( \sin \) are bounded by \(-1\) and \(1\), we see that the maximum absolute value of \( Z_1 \) and \( Z_2 \) is reached for the minimum value of \( U_1 \) which is \( 2^{-n\text{Bits}} \) where \( n\text{Bits} \) is the number of bits used to represent an integer. For 64 bits, we get \( \sqrt{-2\log(2^{-64})} \approx 9.42 \).
Box-Muller transform (polar form): The polar form of Box-Muller transform samples two independent uniform random variables $U_1$ and $U_2$ in $[-1; 1]$ and calculates $s = U_1^2 + U_2^2$. The random variables

$$Z_1 = \sqrt{-2 \log(s)} \frac{U_1}{\sqrt{s}}$$

and

$$Z_2 = \sqrt{-2 \log(s)} \frac{U_2}{\sqrt{s}}$$

are independent and follow a standard normal distribution.

In this case, $\frac{U_1}{\sqrt{s}}$ and $\frac{U_2}{\sqrt{s}}$ always belong to $[-1; 1]$ hence the maximum absolute value reached by $Z_1$ and $Z_2$ is $\sqrt{-2 \log(s_{\text{min}})}$ where $s_{\text{min}}$ is the minimum value possibly reached by $s$, namely $s_{\text{min}} = (2^{-n\text{Bits}})^2 + (2^{-n\text{Bits}})^2 = 2^{-2n\text{Bits}+1}$. For a 64 bits processor, this gives $\sqrt{-2 \log(2^{-127})} \approx 13.27$.

Ziggurat algorithm: The ziggurat algorithm applies to monotonically decreasing probability distributions and extends to symmetric unimodal distributions like the normal one by randomly choosing on which side of the mode the sampled value will fall. The algorithm works by covering the distribution by stacked rectangular regions of same area. What matters for our problem of finding the bound of the sampling process is only the tail rectangle. In the rare case where the value did not fall into one of the other rectangles, a fallback algorithm is used to sample the value from the tail.

Let $x_{\text{tail}}$ be the abscissa of the right side of the last rectangular region before the tail. The fallback algorithm for the normal distribution samples two independent uniform random variables $U_1$ and $U_2$ from $[0; 1]$ and defines $x = -\log(U_1)/x_{\text{tail}}$, $y = -\log(U_2)$. It then tests if $2y > x^2$ and returns $x + x_{\text{tail}}$ if yes. Otherwise, it restarts with two new samples $U_1$ and $U_2$.

As a consequence, the biggest value in the monotonic case, and thus the biggest absolute value in the symmetric case, is $-\log(2^{-n\text{Bits}})/x_{\text{tail}} + x_{\text{tail}}$. In [17], Marsaglia and Tsang calculate that, for 255 rectangles, $x_{\text{tail}} \approx 3.65$. For 64 bits, we then get the following bound: 15.81.

These "artificial" bounds, that we cannot avoid in practice anyway, are justified by the very low probability of a draw outside them: less than $10^{-120}$ for the lowest bound, $9.42$, and less than $10^{-55}$ for the highest, 15.81.

To get a sample from an arbitrary normal distribution, it suffices to scale the sample of the standard normal distribution by the wanted standard deviation and then add the wanted mean. We then easily deduce the bounds of the generated noise from any normal distribution.

This discussion allows us to exhibit a lower bound for the Gaussian noises and, the unnoised updates being bounded by the clipping, for the noised updates. As a consequence we can apply Poisson quantization.

Note that the distributions followed by the draws of the three popular sampling algorithms presented above are almost invariably (and implicitly) considered in the literature as perfect normal distributions. We will then make the same assumption to derive our DP analysis.

4.1.4 The problem of the unbounded Poisson distribution is not a problem

A drawback of Poisson quantization is that it is not bounded, as the Poisson distribution is not. Therefore, even if the Gaussian noises are bounded by the practical
limitations of their sampling algorithms, the quantized noised updates will not be bounded. However, we show in this section that this is actually not a problem for our method.

A first argument to address the issue of the theoretically infinite range of the Poisson distribution would be to study the sampling algorithms for the Poisson distribution and find a practical bound, similarly to what we did for the normal distribution (Section 4.1.3). Nevertheless, this does no seem that easy for the case of the Poisson distribution.

Rather, let us see what happens if the Poisson sample falls out of the bounds imposed by the cryptosystem (the plaintext modulus). At the time of encryption, a modulo operation would automatically be applied to the exceeding value. The same modulo operation will be performed on the aggregated updates on the server side. Observation 1 shows that these two modulo operations amount to a single modulo operation on the sum of the updates, which constitutes a post-processing on this sum and, as such, does not have any impact on the DP analysis.

Observation 1 Let \((x_i)_{i \in [1,K]} \in \mathbb{Z}^K, N \in \mathbb{N}^*\).

\[
\sum_{i=1}^{K} (x_i \mod N) \mod N = \sum_{i=1}^{K} x_i \mod N.
\]

Recall that only discrete values can be manipulated in the encrypted domain. This means that the quantized noised updates are actually multiplied by the inverse quantization scale \(\frac{1}{s}\) before being encrypted, and that the participants rescale the averaged updates by \(s\) once received from the server at the next round.

Let us now consider the influence of this modulo operations on the accuracy of the model. First of all, the result of the Poisson quantization may be non-positive due to the negative offset \(\mu\) and thus may fall out of \([0, \ldots, N-1]\), where \(N\) denotes the plaintext modulus. To avoid this situation, we make the participants send the quantized updates without adding the (potentially non-positive) offset \(\mu\). When they receive the averaged updates from the server, they just have to add \(\mu\) to them to get the actual averaged updates.

The second case is encountered when a sample exceeds the plaintext modulus. Nevertheless, this event is very rare if the modulus is big enough. With the parameters used in our experiments (Section 5), Chebyshev's inequality gives a probability lower than \(10^{-28}\) for this event. In any case, our experiments prove that this has no practical influence on the model accuracy.

4.1.5 DP analysis

According to the discussion above, our fully secure learning mechanism has the same DP guarantees as a learning mechanism where the noise is generated by the server, with a true unbounded and continuous (non-quantized) Gaussian noise. The noise introduced as a side-effect by Poisson quantization may even improve the privacy but, for simplicity reasons, we consider it as banal post-processing.

As explained in 3.2 and pretty much like in [18] for instance, we use the moments accountant technique [21] to compose the privacy costs in an efficient way across the multiple iterations of the learning process.

Formally, given \(\sigma \in \mathbb{R}_+^*\) the standard deviation for the aggregated noise and \(S \in \mathbb{R}_+^*\) the clipping bound in L2-norm, let us consider the two density functions corresponding
to two adjacent databases respectively not containing and containing the adversary’s target client:

\[ f_1 : x \in \mathbb{R} \mapsto \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \]

and

\[ f_2 : x \in \mathbb{R} \mapsto \frac{1 - q}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} + \frac{q}{\sigma \sqrt{2\pi}} e^{-\frac{(x - 2S)^2}{2\sigma^2}} \]

where \( q = \frac{K}{M} \) is the fraction of participants by round i.e. the probability of a given client being chosen to participate in a given round.

Without loss of generality, \( f_1 \) is defined with a null mean, since the integral for the moments accountants computation is on whole \( \mathbb{R} \). Note that the part corresponding to the event whereby the target client is chosen as participant has an offset of \( 2S \) rather than \( S \) because, if the absolute values are constrained by the clipping bound \( S \), the actual span of the values is \( 2S \). As a result, changing the participant may modify the updates by \( 2S \).

The moments accountant of order \( l \in \mathbb{R}^+ \) corresponding to a single query to the private database, i.e. a single learning round, is:

\[ \alpha(l) = \max \left[ \int_{\mathbb{R}} (f_1(x))^l f_2(x) dx, \int_{\mathbb{R}} (f_2(x))^l f_1(x) dx \right]. \]

The total moments accountant of the learning process is \( \alpha_{\text{total}} = T \max_{l \in \mathbb{R}^+} \alpha(l) \) where \( T \) is the number of learning rounds. In practice, we compute the max for \( l \) being an integer varying in \( [1, \ldots, 20] \).

Finally, we apply Theorem 2 to derive the classical DP guarantees \((\epsilon, \delta)\) from \( \alpha_{\text{total}} \).

**Privacy cost from the point of view of a participant:** For a comprehensive analysis, one must not forget that, from the point of view of an adversarial participant, the noise it generated does not participate in the privatisation process. Indeed, if one knows the noise that was added on a value, one just has to remove this noise from the noised value to get the initial value. Hence, we must take into account only the noises of the other participants. The individual noises added by the participants are calibrated such that their sum has a certain standard deviation \( \sigma \) i.e. these individual noises are Gaussian noises with standard deviation \( \frac{\sigma}{\sqrt{K}} \), \( K \) being the number of participants in each round. As a result, the DP guarantees from the point of view of a single participant must be computed by substituting \( \sigma \) by \( \frac{\sqrt{K} - 1}{\sqrt{K}} \sigma \), which does not have a great influence if \( K \) is large. Note that this is still quite conservative as it assumes that the considered participant may participate to all the training rounds.

### 4.2 Homomorphic encryption protects the data (and the model) against the server

While considering our learning framework protected by a distributed noising, it may not be clear why the framework ever needs to make use of cryptography. Indeed, the server receives the updates from the participants after they have been noised. However, the individual noise that seems to blur the updates has been calibrated such that the aggregated noise will obfuscate the sensitive information from a specific participant. If we call \( \sigma \) the standard deviation of the aggregated noise i.e.
standard deviation necessary to hide the data from one participant, the standard.deviation of the individual noise generated by a participant is $\frac{\sigma}{K}$. However, it should be equal to $\sigma$ if it were to protect the updates from the server before aggregation. Such a setting is referred as local differential privacy in the literature. Yet, in our case, this would result in an aggregated standard deviation of $K\sigma$ which would completely destroy the utility of the averaged updates, yielding a totally useless model.

In terms of concrete homomorphic encryption, the fact that we are considering the simple FederatedAveraging operator allows us to use additive-only schemes such as in [13] where the Paillier cryptosystem is used with batching. In the experimental results reported in the next section we have used the BFV cryptosystem which allows for more massive batching and, as such, results in much lower (amortized) overheads. Additionally, one key contribution of [13] was to associate Paillier-based homomorphic calculations to Verifiable Computing (VC) techniques (e.g. [44]) to further extend the server threat model beyond the honest-but-curious one and bring execution integrity. However, that latter work lacked Differential Privacy. Indeed, adding the DP noise on the server requires a tag that can only be generated with knowledge of the VC scheme secret key, meaning, in the Federated Learning context, that at least one of the clients would have knowledge of the noise added (resulting in a collapse of the DP guarantees, even when that knowledge is only probabilistic). As such, associating DP with VC for server-side computation integrity requires a distributed noise generation as provided in this paper. Additionally, the scheme of Fiore et al. [44] has been adapted to work with the BFV scheme in [48]. As such, the noise generation technique provided in this work is also directly applicable to setups were homomorphic calculations are paired with VC techniques.

As a very interesting side-effect, the homomorphic encryption layer also hides the model parameters from the server throughout the training. This may be valuable when the clients want to keep their model private, or give only a black-box access to it, either for privacy or economic reasons (cf. machine learning as a service).

5 Experimental results

In order to prove the practicality of our learning framework, we performed experiments that enable us to evaluate the performance of our method in terms of accuracy, confidentiality and computation time. We chose the Federated Extended MNIST (FEM-NIST) dataset\(^4\) to run the experiments. The extended version of MNIST contains 62 classes (digits, upper and lower letters) of hand-written characters coming from 3,596 writers and comes with the writer id. FEMNIST, the federated version, was built by partitioning the data based on the writer [49]. Each selected user’s dataset has a train/test data split for a total of 165,050 train and 27,433 test images.

The network architecture is the same as in [13]: a standard CNN composed of two convolution layers (respectively with 5 × 5 kernel size and 128 channels, and with 3 × 3 kernel size and 64 channels, each followed with 2 × 2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (486,654 total parameters).

Table 1 shows the influence of the adaptations necessary to ensure DP guarantees on the accuracy of the global model. Starting from a non-secure baseline from the state of the art [13], we successively modified parameters of the framework, each of

\(^4\)Dataset available at https://www.nist.gov/itl/products-and-services/emnist-dataset
these modifications being required by the DP analysis\textsuperscript{4} The successive steps are the following:

- reduce the number of learning rounds from 200 to 100, hence reducing the amount of queries to the clients’ sensitive datasets
- increase the total number $M$ of clients and the number $K$ of participants per round. This has two advantages. Firstly, the ratio $\frac{K}{M}$ is smaller, decreasing the probability of a target client participating at a given round and thus the probability of this target releasing any information during this round. Secondly, the absolute value $K$ is greater, so that the information of the target participant is more diluted in the averaged updates. The impact on the accuracy is due to the larger number of writers, inducing a higher variety in the training samples (non-IDD across the different writers) which makes the classification task more complex.

- assign the same coefficient $\frac{1}{K}$ to all the participants in the weighted average (rather than the proportion $\frac{n_k}{n}$ of training samples owned by the participant $k$) so that the sensitivity of the average for every participant is $\frac{S}{K}$ rather than $\max_{k \in [1:K]} \frac{n_k}{n} S$, where $\max_{k \in [1:K]} \frac{n_k}{n}$ may be much larger than $\frac{S}{K}$
- on the participant side, quantify the noised updates via Poisson quantization in order to be able to represent them with a limited number of bits and fix the plaintext modulus of the encryption scheme

- clip the updates with clipping bound $S$ to ensure a finite sensitivity. In addition, this guarantees that the updates are bounded and, together with the quantization, this ensures they are representable with a limited number of bits
- add the Gaussian noise necessary to make the learning process differentially private

For Poisson quantization, we used as common lower bound $\mu$ of the noises the lower bound from the ziggurat algorithm with 255 rectangles, i.e. 15.81 (see Section 4.1.3), of course multiplied by the standard deviation of the distribution. This lower bound is greater (in absolute value) and then more conservative than the lower bounds of the two other sampling algorithms we considered. Moreover and quite importantly, ziggurat algorithm with 255 rectangles is actually the one chosen by the numpy library we used.

Table 2 summarises the modifications of the parameters between the baseline and our fully private framework.

The whole training process is $(\epsilon, \delta)$-differentially private, with $\epsilon = 5.31$ and $\delta = 10^{-5}$. Actually, for $\delta$ being fixed to $10^{-5}$, $\epsilon = 5.306$ for an end-user which is not a participant and to $\epsilon = 5.313$ for a participant (see discussion at the end of Section 4.1.5).

The experimental results for the homomorphic encryption were realised with BFV in batched mode and Palisade library (version 1.1.6) on an Intel Core i7 with 4 cores at 3 GHz with 32 GB Ram on Ubuntu 18.04. The security level was set to 128 bits and the batch size used was of 16384. Following this the overall 486654 updates can be packed in only 30 ciphertexts (where each of the 16384 slots contains one gradient update). Table 3 provides the overall homomorphic computation time for the full model for a 57 bits modulus resulting in a (fairly practical) maximum of 2.5 secs of homomorphic calculations (per FL round) in the larger setup we considered with 50

\textsuperscript{4}Note that the order in which we made these successive adaptations does not correspond to the order in which they are executed in the learning workflow
| State of the art                          | 84.6% |
|-----------------------------------------|-------|
| Decrease the number of learning rounds $T$ | 83.58%|
| Increase $M$ and $K$                     | 81.04%|
| Assign same coefficients                | 80.09%|
| Quantization                            | 79.67%|
| Clipping of the updates                 | 78.97%|
| Adding random noise                     | 76.78%|

Table 1: Influence of adaptations on accuracy.

|                          | baseline | DP framework |
|--------------------------|----------|--------------|
| Number of rounds $T$     | 200      | 100          |
| Number of clients $M$    | 500      | 3596         |
| Number of participants $K$ | 10      | 1000         |
| Coefficients of the updates in the average | $\frac{n_k}{n}$ | $\frac{1}{K}$ |
| Quantization scale $s$   | 0 (no quantization) | $10^{-4}$ |
| Clipping bound $S$       | infinite (no clipping) | 1          |
| Standard deviation of the total added noise $\sigma$ | 0 (no noise) | 6           |
| Accuracy                 | 84.6%    | 76.78%       |

Table 2: Comparison between the baseline and our framework by parameter

clients. Scaling up to 1000 clients leads to a computation time of around 50 secs per FL round. Performing the full FL cycle (without communications) on a GPU-based HPC cluster takes around 20 hours (i.e., 12 minutes per FL round), resulting in a 7% computation overhead imputable to FHE calculations.
6 Conclusion and perspectives

In this work, we addressed the problem of collaborative learning where all the actors of the training stage -clients and server - and the end-users are potential honest-but-curious adversaries. We chose the popular FL framework and adapted it in two ways. Firstly, by having the clients add random noise on the information they send to the server, we made the learning mechanism differentially private (with specified DP guarantees) from the point of view of any end-user of the model - including the server - and that of the clients themselves. Secondly, following [13], we added a homomorphic encryption layer on the server side so that the server cannot see the updates coming from the clients, that may release sensitive information about the training data.

The homomorphic encryption layer has a major impact on the random noise added to ensure DP, essentially because of the limited number of bits available for a reasonable computation time. However, we proved that this interference can be dealt with in terms of privacy thanks to some adaptations among which a new carefully crafted stochastic quantization operator.

We ran experiments on the popular FEMNIST dataset that prove the practicality of our framework in terms of accuracy, privacy cost and computation time, thoroughly analysing the cost of DP in accuracy compared to an insecure baseline.

The present work could easily be extended to a setting whereby the assumption of the honest-but-curious server is relaxed, making the learning process robust to a server who would, willingly or not, make mistakes in its computations. As argued in Sect. [42], this could be done using verifiable computing techniques, as in [13], in a quite straightforward further work thanks to the fact that the server is not in charge of adding the random noise necessary to DP.

Testing our framework on a larger, more cross-device-oriented dataset would be quite interesting to estimate its scalability. Moreover, this could be advantageous on the privacy point of view since this would allow to increase the number of clients $K$ and thus having simultaneously a big number of participants $m$ but a low ratio $m/K$, conditions that will both improve the DP guarantees of the learning mechanism.

Finding another quantization function that would not need to lower bound the
random noise added to the updates would allow us to get rid of the argument of the imperfect sampling algorithms and to use our framework with other noise distributions, not necessarily bounded, even in practice.

7 Ethical principles

The legal landscape in Artificial Intelligence is evolving quickly, following growing public concerns about weaknesses, risks, and misuse of artificial intelligence (AI). In Europe, United States and China, laws and regulations are evolving to address these concerns. In 2016, the European Union provided an answer to the protection of individual’s privacy against information leakage from a legal standpoint by publishing the General Data Protection Regulation – GDPR. It formally provides with the following key principles:

- Transparency. Personal data must be processed with transparency, and data owners can ask to be informed at any time on how their data are being used.
- Limited purposes and retention. Personal data must be collected for clear, understandable and legitimate purposes, and should not be processed for any other purpose than the initial ones. Moreover, data must be kept for a period that does not exceed the processing of the selected purposes.
- Data integrity and privacy. Personal data must be processed in such a way as to ensure appropriate security of such data, including protection against unauthorised or unlawful processing and against accidental loss.
- Access, rectification, erasing. Data owners should be granted easy access to their personal data as well as the possibility to rectify any inaccuracy or to erase any personal data.
- Privacy by design. Each new technology or application processing personal data, or making it possible to process it, must ensure, from its design and each time it is used, that it incorporates all the protection principles of the GDPR.

Our paper aims at reconciling artificial intelligence practices with existing and future regulations on privacy requirements. It empowers citizens with innovative tools to ensure personal data privacy. Therefore we argue that this work is not only compliant with current ethical expectations, but also contributes to improve future legal and ethical requirements related to artificial intelligence.

References

[1] “GDPR 2018 reform of eu data protection rules.” https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf 2018.

[2] Centers for Medicare & Medicaid Services, “The Health Insurance Portability and Accountability Act of 1996 (HIPAA).” Online at http://www.cms.hhs.gov/hipaa/, 1996.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in IEEE SP, pp. 3-18, IEEE, 2017.
[4] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou, “White-box vs black-box: Bayes optimal strategies for membership inference,” in *ICML*, pp. 5558–5567, 2019.

[5] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *ACM SIGSAC*, pp. 1322–1333, 2015.

[6] M. Ribeiro, K. Grolinger, and M. A. Capretz, “Mlaas: Machine learning as a service,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 896–902, IEEE, 2015.

[7] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation.” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 486–503, Springer, 2006.

[8] Z. Ji, Z. C. Lipton, and C. Elkan, “Differential privacy and machine learning: a survey and review,” *arXiv preprint arXiv:1412.7584*, 2014.

[9] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, “Privacy-preserving deep learning via additively homomorphic encryption.” Cryptology ePrint Archive, Report 2017/715, 2017. [https://eprint.iacr.org/2017/715](https://eprint.iacr.org/2017/715).

[10] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM TIST*, vol. 10, no. 2, pp. 1–19, 2019.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in *ACM SIGSAC*, p. 1175–1191, 2017.

[12] V. Mugunthan and A. Polychroniadou, “SMPAI: Secure multi-party computation for federated learning,” 2019.

[13] A. Madi, O. Stan, A. Mayoue, A. Grivet-Sébert, C. Gouy-Pailler, and R. Sirdey, “A secure federated learning framework using homomorphic encryption and verifiable computing,” in *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, pp. 1–8, IEEE, 2021.

[14] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, “Federated learning of deep networks using model averaging,” *arXiv preprint arXiv:1602.05629*, 2016.

[15] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the gan: information leakage from collaborative deep learning,” in *Proceedings of the 2017 ACM SIGSAC conference on Computer and Communications security*, pp. 603–618, 2017.

[16] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, IEEE, 2019.

[17] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private language models without losing accuracy,” *arXiv preprint arXiv:1710.06963*, 2017.

[18] R. C. Geyer, T. Klein, and M. Nabi, “Differentially private federated learning: A client level perspective,” *arXiv preprint arXiv:1712.07557*, 2017.

[19] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *ACM SIGSAC*, pp. 1310–1321, 2015.
[20] C. Sabater, A. Bellet, and J. Ramon, “Distributed differentially private averaging with improved utility and robustness to malicious parties,” *arXiv preprint arXiv:2006.07218*, 2020.

[21] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *ACM SIGSAC*, pp. 308–318, 2016.

[22] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy,” in *ICML*, pp. 201–210, 2016.

[23] A. Sanyal, M. Kusner, A. Gascón, and V. Kanade, “TAPAS: Tricks to accelerate (encrypted) prediction as a service,” in *ICML*, 06 2018.

[24] E. Hesamifard, H. Takabi, and M. Ghasemi, “Deep neural networks classification over encrypted data,” in *ACM CODASPY*, p. 97–108, 2019.

[25] F. Bergamaschi, S. Halevi, T. T. Halevi, and H. Hunt, “Homomorphic training of 30,000 logistic regression models,” in *International Conference on Applied Cryptography and Network Security*, pp. 592–611, Springer, 2019.

[26] S. Carpov, N. Gama, M. Georgieva, and J. R. Troncoso-Pastoriza, “Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption.” Cryptology ePrint Archive, Report 2019/101, 2019. [https://eprint.iacr.org/2019/101](https://eprint.iacr.org/2019/101).

[27] Q. Lou, B. Feng, G. C. Fox, and L. Jiang, “Glyph: Fast and accurately training deep neural networks on encrypted data,” *arXiv preprint arXiv:1911.07101*, 2019.

[28] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, “Helen: Maliciously secure coopetitive learning for linear models,” in *2019 IEEE SP*, pp. 724–738, IEEE, 2019.

[29] J. Li and H. Huang, “Faster secure data mining via distributed homomorphic encryption,” in *ACM SIGKDD*, pp. 2706–2714, 2020.

[30] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, “POSEIDON: Privacy-preserving federated neural network learning,” *arXiv preprint arXiv:2009.00349*, 2020.

[31] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, “Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning,” in *2020 {USENIX} Annual Technical Conference (ATC 20)*, pp. 493–506, 2020.

[32] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, “Distributed learning without distress: Privacy-preserving empirical risk minimization,” *NeurIPS*, vol. 31, pp. 6343–6354, 2018.

[33] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, “cpsgd: Communication-efficient and differentially-private distributed sgd,” *NeurIPS*, vol. 31, pp. 7564–7575, 2018.

[34] A. Koskela, J. Jälkö, L. Prediger, and A. Honkela, “Tight differential privacy for discrete-valued mechanisms and for the subsampled gaussian mechanism using lft,” in *International Conference on Artificial Intelligence and Statistics*, pp. 3358–3366, PMLR, 2021.

[35] C. Canonne, G. Kamath, and T. Steinke, “The discrete gaussian for differential privacy,” *arXiv preprint arXiv:2004.00010*, 2020.
[36] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, pp. 1273–1282, 2017.

[37] C. Dwork, A. Roth, et al., “The algorithmic foundations of differential privacy,” Foundations and Trends® in Theoretical Computer Science, vol. 9, no. 3–4, pp. 211–407, 2014.

[38] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) Fully Homomorphic Encryption Without Bootstrapping,” in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS ’12, pp. 309–325, 2012.

[39] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” IACR Cryptology ePrint Archive, vol. 2012, p. 144, 2012.

[40] P. Aubry, S. Carpov, and R. Sirdey, “Faster homomorphic encryption is not enough: improved heuristic for multiplicative depth minimization of boolean circuits,” in CT-RSA, pp. 345–363, 2019.

[41] I. Chilotti, N. Gama, M. Georgieva, and M. Izabachène, “Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds,” in ASIACRYPT, pp. 3–33, 2016.

[42] M. Zuber, S. Carpov, and R. Sirdey, “Towards real-time hidden speaker recognition by means of fully homomorphic encryption.” Cryptology ePrint Archive, Report 2019/976, 2019.

[43] A. Grivet Sèbert, R. Pinot, M. Zuber, C. Gouy-Pailler, and R. Sirdey, “Speed: secure, private, and efficient deep learning,” Machine Learning, vol. 110, no. 4, pp. 675–694, 2021.

[44] D. Fiore, R. Gennaro, and V. Pastro, “Efficiently verifiable computation on encrypted data,” in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 844–855, 2014.

[45] G. E. P. Box and M. E. Muller, “A note on the generation of random normal deviates,” The Annals of Mathematical Statistics, vol. 29, pp. 610–611, 1958.

[46] D. E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley, third ed., 1997.

[47] G. Marsaglia and W. W. Tsang, “The ziggurat method for generating random variables,” Journal of statistical software, vol. 5, pp. 1–7, 2000.

[48] A. Madi, O. Stan, and R. Sirdey, “Computing neural networks with homomorphic encryption and verifiable computing,” in Proceedings of the 2nd Workshop on Cloud Security and Privacy, no. 12418 in LNCS, pp. 295–317, 2020.

[49] S. Caldas, S. K. Duddu, P. Wu, T. Li, J. Konečný, H. McMahan, V. Smith, and A. Talwalkar, “Leaf: A benchmark for federated settings.” arXiv preprint 1812.01097, 2019.