Combining homomorphic encryption and differential privacy in federated learning

Arnaud Grivet Sebert, Marina Checri, Oana Stan, Renaud Sirdey, Cedric Gouy-Pailler

Combining homomorphic encryption and differential privacy in federated learning

Arnaud Grivet Sébert, Marina Checri, Oana Stan, Renaud Sirdey and Cédric Gouy-Pailler
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
name.surname@cea.fr

***Abstract*—Recent works have investigated the relevance and practicality of using techniques such as Differential Privacy (DP) or Homomorphic Encryption (HE) to strengthen training data privacy in the context of Federated Learning protocols. As these two techniques cover different sources of confidentiality threats (other clients/end-users for the former, aggregation server for the latter), there is a need to consistently combine them in order to bridge the gap towards more realistic deployment scenarios. In this paper, we achieve that goal by means of a novel stochastic quantization operator which allows us to establish DP guarantees when the noise is both quantized and bounded due to the use of HE. The paper is concluded by experiments on the FEMNIST dataset which show that the precision required to get state-of-the art privacy/utility trade-off (which directly impacts HE parameters and, hence, HE operations performances) results in a computation time overhead between 0.2% and 1.1% imputable to HE (depending on the key setup, either single key or threshold), for the whole training of a 500k parameters model and state-of-the-art privacy/utility trade-off.**

***Index Terms*—Federated learning, differential privacy, homomorphic encryption, quantization**

I. Introduction

Federated Learning (FL) [1] is a decentralized 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 localized on the client devices thus helping to protect data privacy and reducing communication costs. After a common (server-side) arbitrary initialization of the global model, the FL process consists of successive rounds of communication between the server and the clients. The most common approach to optimization for FL is the Federated Averaging algorithm [2]. 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 with several epochs of stochastic gradient descent (SGD) using its own data. The participants then communicate only the updated parameters or the updates[1] themselves (depending on the setting) back to the server. In our work, the participants send the updates because their norm is easier to constrain (necessary for privacy reasons). Finally, the server computes the weighted average of these updates before accumulating them into the global model, thereby concluding the round. The weight associated to a participant in the average is generally the fraction of training samples owned by the participant. Throughout the paper, M is the total number of clients; K is the number of participants per round; the participants are indexed by k with n_k the number of training samples of k.

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 (imperfect) protection to the data with regards 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 [3], [4].

The contribution of this paper is an approach to consistently combine countermeasures of different natures, namely Differential Privacy (DP) and Homomorphic Encryption (HE), with the aim to enable the integration of both in more secure FL frameworks. Indeed, the above-mentioned attacks on the training data can be mitigated via DP, either if they come from the other participants of the training process or from the end-users of the model. Other potential threats come from the central aggregation server which, without any mechanism to secure the aggregation, has access to the model updates. HE can then allow to mitigate these latter threats: 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. As a key contribution, we introduce a novel stochastic quantization operator based on the Poisson distribution, to consistently articulate DP and HE. This operator behaves as if it were applied as post-processing of a Gaussian mechanism. As a result, it keeps the DP guarantees of this standard mechanism unchanged and does not require any supplementary analysis. We can then seamlessly get rid of the quantization issue due to the use of HE. Note that this harmless quantization technique is of independent interest in a context of communication constraints and DP requirements, even without use of cryptographic techniques.

The paper is organized as follows. A review of the literature on the issues of data privacy in an FL context follows this introduction in Section II. Then, Section III provides the technical prerequisites necessary to understand our solution, that we thoroughly explain in Section IV. The results of the experiments that we ran to illustrate the feasibility of our solution are presented in Section V, before some perspectives for further work (Section VI).

II. Related work

The principal focus and key contribution of our paper deals with the interference between DP and HE. The main issue induced by this interference is that the range of the messages to be encrypted (which are the noised updates in our work) has to be discrete and bounded (and encoded with as few bits as possible for better efficiency of the HE computations). For various reasons (not necessarily related to encryption), some authors have studied the possibility of using discrete noises for differential privacy.

[1] Difference between the updated parameters and the old ones.

For instance, the authors of [5] propose a secure and communication efficient distributed learning framework. They 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 if the noise is generated in a distributed way. The analysis is quite involved and only provides DP bounds for the multidimensional binomial mechanism for one round of learning. Indeed, the moments accountant method is not easily applicable to the binomial distribution. Moreover, the presented DP guarantee is worse than the Gaussian mechanism's one and needs the quantization scale to tend to zero (and hence the communication cost to infinity) to approach it.

In [6], 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 experiments. Cannone et al. [7] introduce 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. More critically, the discrete Gaussian distribution is not stable by addition, thus precluding its direct use in a context of distributively generated noise that a collaborative learning task with untrusted server requires (see Section IV-A).

In [8], [9], the authors propose FL protocols protected by DP and secure aggregation (which requires discrete and bounded values, as HE, but needs communication before learning). These works respectively use the discrete Gaussian mechanism and the Skellam mechanism to ensure DP. At the cost of a careful DP analysis, they show that, for fine enough quantization scale, their DP guarantees approach the Gaussian mechanism's ones. These two works have to make use of conditional randomized rounding to ensure that the rounding of the unnoised values does not increase their norm too much. Since we perform quantization after noising with a quantization that can be viewed (from the DP perspective) as a post-processing (see Section IV-B), we do not have such an issue.

The closest work to ours is [10] that uses a binomial law to ensure DP not by adding noise but by encoding the sensitive values into a parameter of the binomial law, as we do with the Poisson law. Thanks to an involved analysis, that makes use of Kashin's representation [11], they get a privacy-utility trade-off from this mechanism that asymptotically reaches the same order of magnitude as the Gaussian mechanism's one. The key difference with our work is that we do not use the discrete noise to ensure DP but only as a quantization operator that commutes with the aggregation. By actually adding Gaussian noise and only treating the Poisson quantization as a post-processing, and at a negligible cost in accuracy, our work thus proposes a much simpler way to obtain the very same guarantees as the Gaussian mechanism, without needing to constrain the quantization scale and with a straightforward analysis.

III. Preliminaries

A. Differential privacy

Differential Privacy (DP) [12] is a gold standard concept in privacy-preserving data analysis. It provides a guarantee that, under a reasonable privacy cost (ϵ, δ), two *adjacent* databases produce statistically indistinguishable results. In our FL framework, the term database denotes the concatenation of all the clients' datasets and two databases are adjacent if they have the same number of clients and differ on a single client, all the others remaining unchanged. Yet, the differing clients may have totally different data, making our notion of adjacency quite conservative (this is called *user-level privacy*). We define below the notions of DP and sensitivity and state a fundamental property of DP, namely immunity to post-processing.

Definition 1. *Given* $(\epsilon,\delta) \in \left(\mathbb{R}_+^*\right)^2$*, a randomized mechanism* $\mathcal{M}$ *with output range* $\mathcal{R}$ *satisfies* (ϵ,δ)-DP *if, for any two adjacent databases* d,d' *and for any subset* $S \subset \mathcal{R}$*, one has* $\mathbb{P}[\mathcal{M}(d) \in S] \leq e^{\epsilon}\mathbb{P}[\mathcal{M}(d') \in S] + \delta$.

Definition 2. *Let* $\mathcal{M}$ *be a randomized 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.*

Proposition 1 ([13]). *Let* $\mathcal{M}$ *be a randomized algorithm, with output range* $\mathcal{R}$*, that is* (ϵ,δ)*-differentially private, with* $(\epsilon,\delta) \in \left(\mathbb{R}_+^*\right)^2$*. Let* $f : \mathcal{R} \to \mathcal{R}'$ *be an arbitrary randomized mapping. Then* $f \circ \mathcal{M}$ *is* (ϵ,δ)*-differentially private.*

To determine the privacy cost (ϵ, δ) of our protocol, we determine the privacy cost per query and then we compose the privacy costs of all queries to get the overall cost. To keep track of the privacy cost along the training we use the *moments accountant* [14] and its key properties of composition and tail bound (Theorems 1 and 2).

Definition 3. *The* moments accountant *is defined for any* $l \in \mathbb{N}^*$ *as*

$$\alpha_{\mathcal{M}}(l) := \max_{aux,d,d'} \log\left(\mathbb{E}_{o\sim\mathcal{M}(aux,d)}\left[\left(\frac{\mathbb{P}[\mathcal{M}(aux,d)=o]}{\mathbb{P}[\mathcal{M}(aux,d')=o]}\right)^l\right]\right)$$

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

Theorem 1 ([14]). *Let* $p \in \mathbb{N}^*$*. Let us consider a mechanism* $\mathcal{M}$ *defined on a set* $\mathcal{D}$ *that consists of a sequence of adaptive mechanisms* $\mathcal{M}_1,\ldots,\mathcal{M}_p$ *where, for any* $i \in \{1,\ldots,p\}$*,* $\mathcal{M}_i : \prod_{j=1}^{i-1}\mathcal{R}_j \times \mathcal{D} \mapsto \mathcal{R}_i$*. Then, for any* $l \in \mathbb{N}^*$*,* $\alpha_{\mathcal{M}}(l) \leq \sum_{i=1}^{p}\alpha_{\mathcal{M}_i}(l)$.

Theorem 2 ([14]). *For any* $\epsilon \in \mathbb{R}_+^*$*, the mechanism* $\mathcal{M}$ *is* (ϵ,δ)*-differentially private for* $\delta = \min_{l\in\mathbb{N}^*}\exp(\alpha_{\mathcal{M}}(l) - l\epsilon)$.

B. Homomorphic Encryption

For some of the most popular HE schemes (BGV [15], BFV [16]) the plaintext domain is defined over the ring $R_t = R/tR$ with $R = \mathbb{Z}[x]/f(x)$ the polynomial ring modulo the function f and the integer $t \geq 2$ with, usually, $f(x) = X^n + 1$ and n a power of 2. As such, before encryption, each message has to be encoded as a plaintext consisting in a polynomial of degree smaller than n with integer coefficients from the range (0, t-1), and all operations over individual elements are performed modulo $(X^n + 1)$, and modulo t. The ciphertext space for these schemes is $R_q = R/qR$.

Moreover, a lot of HE schemes, like BFV that we use in our experiments (Section V), offer a *batching* capability by which multiple cleartexts can be packed in one ciphertext resulting in SIMD (Single Instructions Multiple Data) homomorphic operations. i.e., $\mathsf{Enc}(m_1, ..., m_\kappa) \oplus \mathsf{Enc}(m'_1, ..., m'_\kappa) = \mathsf{Enc}(m_1+m'_1,...,m_\kappa+m'_\kappa)$ (and similarly so for $\otimes$). Typically, several hundreds such slots are available (for BFV, the maximal number of slots coincides with n), which often allows to significantly speed up encrypted domain calculations. In order to apply batching for BFV, t has to be prime and $t=1 \ mod \ [2n]$.

In all the mainstream FHEs with decent efficiency, data (on which calculation is carried out) are encrypted under a unique public key. In other words, users share the same key pair and can therefore decrypt other users' private data. That is a notable limitation for protocols involving sensitive or confidential data that different parties cannot or do not want to share.

The improved performance of the FHE has paved the way for other approaches, more sophisticated than single-user approaches. Several cryptographic schemes investigate using multi-key techniques to adapt HE to multi-user settings [17], such as threshold homomorphic encryption (ThHE) [18]–[21], multi-key homomorphic encryption (MKHE) [22]–[25] or more recently, hybrid approaches proposed by [26].

However, these techniques yield much higher computational overhead than merely FHE. Specifically, the growth of ciphertext size (at least) linear in the number of users for MKHE leads to a notable increase in the computational cost of homomorphic operators. And, to the best of the authors' knowledge, hybrid approaches have only been built on BGV [27], limiting the set of features one can implement. As for ThHE, this technique requires deleting and re-encrypting all data each time a user leaves.

Although ThHE has a static configuration, the clients are expected to stay mostly the same in our FL use case. Once the group is composed, it should remain the same so the server can train its model correctly. Thus there is no need to re-run the setup.

ThHE schemes allow a subset of the users to collectively decrypt data encrypted under a joint public key (computed from all the individual ones in a public way), where the subset size T is fixed and specified at setup. These schemes ensure that no single entity holds the decryption key (i.e. the private key). Each user can only access partial knowledge about this key (except if the threshold is willingly set at $T=1$) and needs other users to perform the decryption. In a configuration where a semi-honest adversary may control both the server and a number of the clients (i.e. where the server effectively colludes with a number of clients), the threshold can be set to a number strictly greater than that of expected colluding users. That way, such an adversary does not possess a decryption ability on its own and, as a result, remains contained by the protocol in terms of the information it can access in clear form.

IV. Noise generation and quantization

A. Distributed noise generation

When willing to protect the training data by DP in a FL process, having the participants generate the noise in a distributed way ([28], [29]) rather than to rely on the server to do so is desirable to mitigate a server that would communicate the noise to some clients or end-users and then break the DP guarantees. Of course this would not be *stricto sensu* needed for the most basic threat models dealing only with honest-but-curious non-colluding adversaries, and a server that has no access to the trained model. In that case, the central noise would be generated in the clear domain and homomorphically added to the aggregated updates, and quantization would not cause any difficulty.

The distributed noise generation, that we here adopt, is especially practical when one wants the resulting noise to follow a Gaussian distribution, since this distribution is stable by addition. The participants simply need to generate Gaussian noises with well-chosen variances. However, a FL process with DP still requires adaptations:

1) clipping the updates in L2-norm with the clipping bound S (*i.e.* 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 (i.e. the impact of changing from one dataset to an adjacent one) since unbounded sensitivity is incompatible with any DP guarantee;
2) adding noise to the gradients (*e.g.* Gaussian noise);
3) fixing all the coefficients of the mean to $\frac{1}{K}$, independently of the size of the participant's dataset, to bound the sensitivity more easily.

B. Poisson quantization

We here propose a new probabilistic quantization operator that commutes with the sum[2], and is therefore harmless for the DP guarantee of the mechanism. In the following, $\mathcal{P}(\lambda)$ 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 4. *Let $\mu \in s\mathbb{Z}$. We define the probabilistic function*

$$Q_{s,\mu}\colon x \in]\mu;+\infty[\mapsto sY+\mu$$

where $Y \sim \mathcal{P}\left(\frac{x-\mu}{s}\right)$. We call it the Poisson quantization *of scale s and offset μ. Similarly, we define $Q_{s,\mu}\colon x = \left(x^{(i)}\right)_{i\in[\![1;d]\!]} \in]\mu;+\infty[^d \mapsto \left(Q_{s,\mu}\left(x^{(i)}\right)\right)_{i\in[\![1;d]\!]}$.*

Given $\mu \in s\mathbb{Z}$, for all $x \in]\mu;+\infty[^d$, $Q_{s,\mu}(x)$'s support is included in $(s\mathbb{Z})^d$ and its mean is equal to x so we can actually consider the Poisson quantization as an unbiased quantization operator. 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 $m \in \mathbb{N}^*$, $x_1,...,x_m \in \mathbb{R}$. Let $\mu \in s\mathbb{Z}$ such that $\mu < \min\{x_i | i \in [\![1;m]\!]\}$. $\sum_i^m Q_{s,\mu}(x_i)$ has the same distribution as $Q_{s,m\mu}(\sum_i^m x_i)$.*

Proof. $\sum_i^m Q_{s,\mu}(x_i) \sim \sum_i^m (sY_i + \mu) = s\sum_i^m Y_i + m\mu$ where, for all $i \in [\![1;m]\!]$, $Y_i \sim \mathcal{P}(\frac{x_i-\mu}{s})$. By stability of the Poisson law by addition, we know that $\sum_i^m Y_i \sim \mathcal{P}\left(\sum_i^m \frac{x_i-\mu}{s}\right) = \mathcal{P}\left(\frac{\sum_i^m x_i - m\mu}{s}\right)$. □

Proposition 2 together with Proposition 1 enable us to conclude that Poisson quantization has no influence on the DP guarantee. Indeed, the output distribution is the same as if we had applied the Poisson quantization after the aggregation of the *continuously* noised updates. Since besides adding continuous

[2] Commutativity must be understood in a large sense, as the offset parameter of the quantization changes depending on the order of the operators.

Gaussian noises distributively on the updates and sum the noised updates up afterwards amounts to add a Gaussian noise to the sum of the unnoised updates, the Poisson quantization acts as if it was applied on top of the Gaussian mechanism. *Hence, the key advantage of our Poisson quantization operator is that it allows to reduce the DP analysis back to the vanilla analysis of the Gaussian mechanism* (Section IV-E).

Note that, since Poisson quantization is probabilistic, it might harm the accuracy of the model. Given $\mu \in s\mathbb{Z}$ and $x \in]\mu;+\infty[$, the variance of $Q_{s,\mu}(x)$ is $s^2 \frac{x-\mu}{s} = s(x-\mu)$. For a small enough s, this variance is very small since x is bounded, and there is actually no impact on accuracy in our experiments (Section V).

An important point to notice is that Poisson quantization implies that the values to quantize have an *a priori* common lower bound (otherwise the sum of the quantized values may depend on these values and not only on their sum). In our case, these values are the noised updates. The updates are already bounded by the clipping. As for the noises, the following section shows we can consider that the noises have a common lower bound in practice.

C. Bounded Gaussian noises

The most common algorithms to sample from a Gaussian distribution are Box-Muller transform in its Cartesian and polar forms ([30], [31]) and the ziggurat algorithm [32]. Knowing that they rely on a source of uniform randomness, it can be easily shown that all these algorithms actually generate values whose range have bounds which are way smaller than the range of double-precision floats. For 64 bits, the Box-Muller transform in Cartesian form, the Box-Muller transform in polar form and the ziggurat algorithm respectively generates samples of the standard normal distribution whose absolute value is bounded by 9.42, 13.27 and 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^{-20} 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. This discussion allows us to exhibit a lower bound for the Gaussian noises. As a consequence we can apply Poisson quantization.

D. The problem of the unbounded Poisson distribution is not a problem

A drawback of Poisson quantization is that it is not bounded, while the cryptosystem only works on a finite set of values. Indeed, even if the noised updates are bounded in practice, the quantized noised updates are not. However, we show in this section that this is actually not a problem for our method. Let us see what happens if the Poisson sample falls out of the bounds imposed by the plaintext domain. A modulo operation will automatically be applied to the individual updates at encryption and to 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 affect 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 integer values can be manipulated in the encrypted domain. This implies that the quantized noised updates are 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 μ. To avoid this situation, we make the participants send the quantized updates without adding the (potentially non-positive) offset μ. When they receive the averaged updates from the server, they just have to add μ to them after decryption 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 we use (Section V), we can show, using Chebyshev's inequality, that the probability for a quantized gradient (resp. the sum of the K quantized gradients) to exceed the plaintext modulus is lower than 1.01×10^{-9} (resp. 1.61×10^{-5}), to compare to the number 486,654 of parameters. In any case, our experiments prove that this has no practical influence on the model accuracy.

E. DP analysis of the Gaussian mechanism

According to the discussion above, our mechanism has the same DP guarantees as a mechanism where true unbounded and continuous Gaussian noise is added by the server *after* aggregation, a.k.a. the Gaussian mechanism. The noise introduced as a side-effect by Poisson quantization may even improve the privacy but, for simplicity, we consider it as banal post-processing. Hence, the DP analysis reduces to the vanilla Gaussian mechanism's analysis. As explained in III-A and pretty much like in [33] for instance, we use the moments accountant [14] to compose privacy costs in an efficient way across the multiple learning rounds.

1) 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 a participant k, the noise generated by k does not participate in the privatization process[3]. Hence, we must take into account only the other participants' noises. The individual noises added by the participants are calibrated such that their sum has a certain standard deviation σ i.e. these individual noises have standard deviation $\frac{\sigma}{\sqrt{K}}$, K being the number of participants in each round. As a result, the DP guarantee from the point of view of a single participant must be computed by substituting σ by $\frac{\sqrt{K-1}}{\sqrt{K}}\sigma$, which has an insignificant influence if K is large (1000 in our experiments). Note that this is still quite conservative as it assumes that the considered participant may participate to all training rounds. We can also interestingly extend our threat model in a straightforward way by considering that some clients may collude and share their noises with each other, quite like in [29]. From the point of view of a colluding client, the noise added by all the colluding clients would be known and it would therefore not participate in the DP protection of the data. This would result in a degraded DP guarantee, obtained by substituting σ by $\sqrt{1-\chi}\sigma$, where χ is the ratio of colluding participants. This modification of the DP guarantees

[3] For instance, if one knows the noise that was added to a value, one just has to remove this noise from the noised value to get the initial value.

also applies in the case of some clients dropping out. Figure 2 illustrates the loss of confidentiality due to collusion of clients.

F. 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 even needs to make use of cryptography. Indeed, the server receives the updates from the participants after they have been noised. However, each individual noise has been calibrated such that the *aggregated* noise will obfuscate the sensitive information of a specific participant. If σ is the standard deviation necessary to hide the data of one participant, the standard deviation of each individual noise is $\frac{\sigma}{\sqrt{K}}$. However, since without HE the server would see each individual noise updates *before* aggregation, the individual noise should be equal to σ if it were to protect the updates from the server. Such a setting is referred to as *local DP* in the literature. Yet, in our case, this would result in an aggregated standard deviation of $\sqrt{K}\sigma$ (for the sum, or $\frac{\sigma}{\sqrt{K}}$ for the average) which would heavily harm the utility of the averaged updates and thus the accuracy of the model.

In terms of concrete HE, the fact that we are considering the simple FederatedAveraging operator allows us to spare much computation time by using additive-only schemes such as in [34] 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 [34] was to associate Paillier-based homomorphic calculations to Verifiable Computing (VC) techniques (*e.g.* [35]) to further extend the server threat model beyond the honest-but-curious one and bring execution integrity, as [36] did with BFV scheme. However, these works lacked DP. Indeed, adding the DP noise on the server requires a tag that can only be generated with knowledge of the VC scheme secret key (i.e. by a client), meaning, in the FL context, that at least one of the clients would have knowledge of the total noise added (resulting in a collapse of the DP guarantee regarding this client, even when that knowledge is uncertain). We could actually imagine that the server generates K noises that sum to the noise it added and send each of them to a participant for tag generation but this would double the communication cost. Hence, associating DP with VC for server-side computation integrity maintaining a reasonable communication cost requires a distributed noise generation as provided in this paper. As such, the noise generation technique proposed in this work is directly applicable to setups where homomorphic calculations are paired with VC techniques.

As a very interesting side-effect, the HE 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).

V. Experimental results

To prove the practicality of the combination of our Poisson quantization technique with HE, we performed experiments that enable us to evaluate training performance in terms of accuracy, precision requirements and computation time. We chose the Federated Extended MNIST (FEMNIST) dataset[4] to run the experiments. The extended version of MNIST contains 62 classes (digits, upper and lower letters) of hand-written characters from 3,596 writers and comes with the writer id. FEMNIST, the federated version, was built by partitioning the data based on the writer [37]. The network architecture is the same as in [34]: 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 by $2*2$ max pooling), a fully connected layer with 128 units and ReLu activation, and a final softmax output layer (486,654 parameters).

Table I shows the influence on the model accuracy of the adaptations necessary to ensure DP. Starting from a non-DP baseline from the state of the art [34], we successively modified parameters of the framework, each of these modifications being required by the DP analysis[5]. The successive steps are:

- reduce the number of learning rounds from 200 to 100, hence reducing the amount of queries to the datasets and thus the privacy cost: as shown in Table I, this has a very mild influence on the model accuracy whereas, for smaller number of learning rounds, the accuracy starts decreasing more significantly
- increase the total number M of clients (to 3596, the total number of writers for FEMNIST) and the number K of participants per round to 1000. This has two advantages. Firstly, we can make the ratio $q = \frac{K}{M}$ 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 the information of the target participant is more diluted in the averaged updates. In practice, the experiments show that, with a fixed distortion ratio $\frac{\sigma}{K}$, which gives roughly the same model accuracy, and with M set to 3596 the DP guarantee ϵ decreases when K increases. We then chose $K = 1000$, in our opinion the largest reasonable value so that a substantial ratio of the clients can stay idle at each round. The impact on the accuracy of the increasing of M and K 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 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 $\frac{1}{K}$
- clip updates with clipping bound S to ensure finite sensitivity (we took $S = 1$ which has a mild impact on accuracy and allows for good DP guarantees)
- on the participant side, quantify the noised updates via Poisson quantization
- apply a modulo operation on the noised updates on the participant side, and on their sum on the server side, which is automatically done by encryption
- add the Gaussian noise necessary to make the learning process differentially private. We chose $\sigma = 6$ for the total

[4] Dataset available at https://www.nist.gov/itl/products-and-services/emnist-dataset

[5] 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.

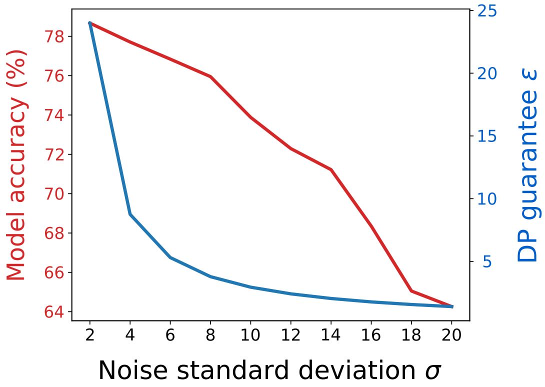

Fig. 1. Model accuracy and DP guarantee vs noise standard deviation ($\delta = 10^{-5}$)

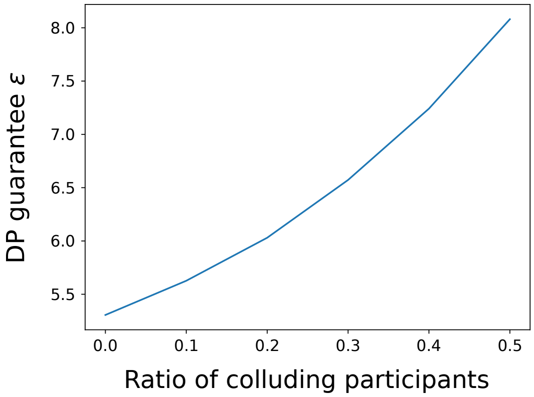

Fig. 2. DP guarantee vs ratio of colluding participants ($\delta = 10^{-5}$)

noise because it gives a good trade-off between privacy and model accuracy as shown in Figure 1.

We fixed the scale for Poisson quantization to 10^{-4}, as in [34], since it does not much affect the accuracy. We used as common lower bound μ of the Gaussian noises the lower bound from the ziggurat algorithm with 255 rectangles, i.e. -15.81 (see Section IV-C), 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 the algorithm chosen by the numpy library we used.

The whole training process is (ϵ, δ)-differentially private, with $\epsilon = 5.31$ and $\delta = 10^{-5}$. Actually, for $\delta = 10^{-5}$, $\epsilon = 5.306$ for an end-user which is not a participant and $\epsilon = 5.309$ for a participant (see discussion at the end of Section IV-E). More widely, Figure 2 represents the privacy cost, from the point of view of a colluding participant i, as a function of the ratio of participants who collude with i. As expected, we see that the privacy cost increases smoothly with the ratio of colluding participants and, up to say 20% of colluding participants (i.e. 200 colluding teachers) the privacy cost remains reasonably close to the one in the non-colluding case.

These DP guarantees together with the model accuracy of 76.84% give us the same privacy/utility trade-off as [8], [9] got with secure aggregation. Nevertheless, if communication is a critical issue, we may use a greater quantization scale, at the expense of accuracy, but this would not harm the DP guarantee, contrary to [8], [9]. Interestingly, we experimentally notice that the quantization and the modulo operation have no influence on the accuracy: the model trained with noise but without quantization or modulo operation still has an accuracy of 76.84%. Therefore, *quantization comes at no cost*, both privacy-wise and accuracy-wise.

TABLE I
INFLUENCE OF SUCCESSIVE ADAPTATIONS ON ACCURACY.

	Accuracy
State of the art [34]	84.6%
Decrease the number of learning rounds T	83.58%
Increase M and K	81.04%
Assign same coefficients	80.21%
Clipping of the updates	79.26%
Quantization	79.03%
Modulo operation	79.07%
Adding random Gaussian noise	**76.84%**

TABLE II
COMPUTATION TIME (IN SECONDS) OF HE OPERATIONS WITH A 26-BIT MODULUS FOR THE *full* 486654 WEIGHTS MODEL.

Users (keys)	1	1000	3596
Participants (additions to perform)	1000	1000	1000
Context generation	0,0036	0,0073	0,0073
Key generation	0,0019	1,5545	5,5885
Encoding	0,0062	0,0072	0,0072
Encryption	0,1509	0,2358	0,2355
Evaluation	1,2761	3,0368	3,0544
Total decryption latency	0,0279	1,4376	4,6242

The experimental results for HE were realised with BFV in batched mode and OpenFHE library (version 1.0.3) on an Intel i7-12700H (20) @ 4.600GHz with 64 GB Ram on Ubuntu 22.04.2 and one core activated. The security level was set to 128 bits and the batch size used was of 8,192. Following this the overall 486,654 updates can be packed in only 60 ciphertexts (where each of the 8,192 slots contains one gradient update). Table II provides the overall homomorphic computation time *for the full model* for a 26-bit modulus and 1000 participants per round resulting in a (fairly practical) order of magnitude between 1 and 3 seconds of homomorphic calculations (per FL round) for all three setups (single key, key shared among 1000 participants, key shared among 3596 participants[6]). Overall, performing the full FL cycle (without communications) on a GPU-based HPC cluster takes around 20 hours (i.e., 12 minutes per FL round), to which between 1.5 secs (single key setup) and 8 secs (threshold setup) of latency per round should be added due to the use of FHE (for encryption, evaluation and decryption). This results in a 0.2% to 1.1% computation time overhead imputable to the use of HE on the overall procedure.

The choice of a 26-bit modulus is due to an empirical investigation. For 26 bits or more, the model trains correctly, with almost no impact of the modulo operation on the accuracy (see Table I). Below 26 bits, the model does not learn at all. This sharp change

[6] Due to current OpenFHE limitation to n-out-of-n threshold schemes we give results for the 1000-out-of-1000 and 3596-out-of-3596 configurations which respectively give (close) lower and upper bounds for an 1000-ouf-of-3596 configuration which would be more appropriate in our experimental setup.

of behavior is due to the fact that the modulus exponentially depends on the number of bits and that the distribution of the quantized noised updates is actually very peaked - the ratio standard deviation over expectation is lower than 2.22×10^{-3}.

VI. Perspectives

On the server side, the present work could be extended to cover more advanced threat models, making the learning process robust to a server who would, willingly or not, make mistakes in its computations. As argued in Section IV-F, this could be done using verifiable computing techniques, as in [34], 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. It should also be emphasized that our quantization technique may also prove useful when combined with other cryptographic techniques for computing over encrypted data such as MPC and Functional Encryption which also have applications in FL. Of course, our quantization operator also addresses the issue of communication overload even in a cryptography-free context.

Testing our approach on a larger, more cross-device-oriented dataset would be quite interesting to further estimate its scalability. Moreover, this could be advantageous from the privacy point of view since this would allow to increase M, the number of clients and thus having simultaneously a large number of participants K and a low ratio $\frac{K}{M}$, conditions that will both improve the DP guarantees of the learning mechanism.

Another quantization function or a more involved analysis 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, possibly unbounded, even in practice.

Acknowledgments

This work was partially supported by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute. The research leading to these results has received partial funding from the European Union's Preparatory Action on Defence Research (PADR-FDDT-OPEN-03-2019). This paper reflects only the authors' views and the Commission is not liable for any use that may be made of the information contained therein. The authors would also like to thank Aurélien Mayoue for providing the Federated Learning code that was used as a starting point to perform the experiments.

References

[1] 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.

[2] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.

[3] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *ACM CCS*, 2017, pp. 603–618.

[4] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *SP*. IEEE, 2019, pp. 691–706.

[5] 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.

[6] A. Koskela, J. Jälkö, L. Prediger, and A. Honkela, "Tight differential privacy for discrete-valued mechanisms and for the subsampled gaussian mechanism using fft," in *AISTATS*, 2021, pp. 3358–3366.

[7] C. Canonne, G. Kamath, and T. Steinke, "The discrete gaussian for differential privacy," *arXiv preprint arXiv:2004.00010*, 2020.

[8] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *ICML*, 2021, pp. 5201–5212.

[9] N. Agarwal, P. Kairouz, and Z. Liu, "The skellam mechanism for differentially private federated learning," *NeurIPS*, 2021.

[10] W.-N. Chen, A. Ozgur, and P. Kairouz, "The poisson binomial mechanism for unbiased federated learning with secure aggregation," in *ICML*, 2022, pp. 3490–3506.

[11] B. Kashin, "Section of some finite-dimensional sets and classes of smooth functions (in russian) izv," *Acad. Nauk. SSSR*, vol. 41, pp. 334–351, 1977.

[12] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*, 2006, pp. 486–503.

[13] 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.

[14] M. Abadi *et al.*, "Deep learning with differential privacy," in *ACM SIGSAC*, 2016, pp. 308–318.

[15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS*, 2012, pp. 309–325.

[16] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[17] A. Aloufi, P. Hu, Y. Song, and K. Lauter, "Computing blindfolded on data homomorphically encrypted under multiple keys: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–37, 2021.

[18] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," in *Advances in Cryptology – EUROCRYPT 2012*, 2012, vol. 7237, pp. 483–501.

[19] A. Jain, P. M. R. Rasmussen, and A. Sahai, "Threshold fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, p. 257, 2017.

[20] D. Boneh *et al.*, "Threshold cryptosystems from threshold fully homomorphic encryption," in *Advances in Cryptology – CRYPTO*, 2018, vol. 10991, pp. 565–596.

[21] S. Chowdhury *et al.*, "Efficient threshold FHE with application to real-time systems," *IACR Cryptol. ePrint Arch.*, p. 1625, 2022.

[22] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, p. 94, 2013.

[23] Y. Doröz, Y. Hu, and B. Sunar, "Homomorphic AES evaluation using the modified LTV scheme," *Designs, Codes and Cryptography*, vol. 80, no. 2, pp. 333–358, 2016.

[24] P. Ananth, A. Jain, Z. Jin, and G. Malavolta, "Multi-key fully-homomorphic encryption in the plain model," in *TCC*, 2020, pp. 28–57.

[25] T. Kim, H. Kwak, D. Lee, J. Seo, and Y. Song, "Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition," *IACR Cryptol. ePrint Arch.*, p. 347, 2022.

[26] A. Aloufi and P. Hu, "Collaborative Homomorphic Computation on Data Encrypted under Multiple Keys," *IWPE*, 2019.

[27] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *IACR Cryptol. ePrint Arch.*, p. 277, 2011.

[28] 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.

[29] A. Grivet Sébert, 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.

[30] 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.

[31] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed. Addison-Wesley, 1997.

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

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

[34] 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 *RDAAPS*. IEEE, 2021, pp. 1–8.

[35] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *ACM CCS*, 2014, pp. 844–855.

[36] A. Madi, O. Stan, and R. Sirdey, "Computing neural networks with homomorphic encryption and verifiable computing," in *Cloud S&P*, 2020, pp. 295–317.

[37] S. Caldas *et al.*, "Leaf: A benchmark for federated settings," 2019, arXiv preprint 1812.01097.