Towards Learning on Vertically Partitioned Data with Distributed Differential Privacy

Ergute Bao*, Fei Wei†, Yin Yang‡, Xiaokui Xiao*, Tianyu Pang§, Chao Du§
*National University of Singapore, †Alibaba Group, ‡Hamad Bin Khalifa University, §SEA AI Lab,
ergute@comp.nus.edu.sg, feiwei@alibaba-inc.com, yyang@hbku.edu.qa, xkxiao@nus.edu.sg, {tianyupang, duchao}@sea.com

***Abstract*—Analysis of distributed data typically requires the collaboration of the data owners, as well as privacy protection. This paper focuses on the scenario where the database is *vertically* partitioned onto the data owners (referred to as VFL), e.g., an e-commerce platform and an online payment service collaborate to build a model to predict user behavior. To avoid revealing their private data during model fitting, the data owners commonly participate in a cryptographic protocol such as secure multiparty computation. However, the resulting model may still leak sensitive information under sophisticated data extraction attacks. A rigorous solution to this issue is to compute the model with differential privacy (DP), which provides strong and well-accepted privacy guarantees. Enforcing DP on VFL turns out to be highly challenging and there does not yet exist an effective solution that does not rely on any trusted party. Consequently, practitioners are left with rather basic approaches for ensuring DP, e.g., each data owner perturbs her local data with additive noises, leading to suboptimal model utility. Can we achieve privacy-utility trade-offs for VFL with DP comparable to the centralized setting, without trusting any party?**

In this paper, we take a significant step towards providing a positive answer to this question. We focus on a subset of the data analysis and machine learning tasks–the class of tasks where the sensitive information to release can be expressed as a polynomial function of the input. Following the distributed DP framework that does not require any trusted party, we propose a generic mechanism to solve this class of problems, called the *Skellam Quantization Mechanism* (*SQM*). We formally prove the privacy guarantee of our solution, and show that it is able to match the privacy-utility trade-offs in the centralized setting. We then instantiate *SQM* on two classical tasks, principal component analysis and logistic regression. Extensive experiments on real-world datasets confirm the strong performance of *SQM*.

I. Introduction

Privacy-preserving data analysis and machine learning over partitioned databases have received considerable attention in the database community in recent years [1]–[12], where a dataset containing sensitive information is scattered among multiple database owners, who must keep their private information confidential, and at the same time would like to collaborate to analyze the sensitive data and build machine learning models with strong predictive power. This motivates federated learning (FL) [13], in which data can be horizontally partitioned (i.e., each party possesses a subset of the records) or vertically partitioned (each party possesses a subset of the attributes). This paper focuses on the latter, commonly referred to as vertical FL (VFL), which has promising practical applications (see [14] for a survey). For instance, a search engine and an online payment service may possess users' search history and financial transactions, respectively, and their user bases often have large overlaps. VFL enables these data owners to jointly train a machine learning model to predict user behavior, without sharing data with each other. We refer to Figure 1 for an illustration of the data partition in VFL.

Sensitive data

Record ID	Att 1	…	Att j	…	Att n
Alice	0.3	…	-0.1	…	1
Bob	0.22	…	0.61	…	0
…	…	…	…	…	…

$\mathbf{X}$

Party j's partition

Record ID	Att j
Alice	-0.1
Bob	0.61
…	…

$\mathbf{X}[:, j]$

Fig. 1: An example of data partitioning in VFL–each party possesses one attribute of the dataset.

Earlier FL protocols involve a centralized coordinator who interacts with the data owners to build a model collaboratively [15], [16]. A concern with this approach is that the coordinator (who could be adversarial) might infer sensitive information on the participants' data through the FL training process. This can be addressed through a cryptographic solution such as secure multiparty computation (MPC) and homomorphic encryption (HE) which preserves the confidentiality of each data owner [3], [6], [9], [11], [17]. However, the analysis results (e.g., the trained model) may still leak private information of the underlying database, especially in the presence of increasingly powerful data extraction attacks [18], [19], which might lead to violations of privacy regulations such as GDPR[1] and CCPA[2].

To address the above privacy issue, a rigorous framework is to compute the model with differential privacy (DP) [20], [21], a strong privacy standard that has been accepted in the database community [22]–[29] and also successfully adopted in practice [30]–[32]. The common approach to achieve DP is through noise injection–a random noise is injected into the analysis results (e.g., parameters of the trained model), where the scale of the noise is calibrated to the *sensitivity* of the analysis function. Here, the sensitivity corresponds to the maximum influence of an individual record on the analysis

[1]https://gdpr-info.eu
[2]https://oag.ca.gov/privacy/ccpa

result. This ensures that the adversary cannot infer with high confidence whether or not a particular record is present in the underlying data, thus providing plausible deniability to the individuals involved in the data.

Gap. It is not straightforward to adopt the idea of noise injection on vertically partitioned databases (or VFL) to achieve DP while maintaining high utility. In the literature, DP has been extensively studied in the centralized (e.g., see [28], [33], [34]) and local setting (e.g., see [35]–[38]). The former, referred to as central DP, assumes a centralized party who has access to all records and their attributes in the database, performs the analysis, and injects noises into the result. Central DP, however, is *inherently incompatible* to the distributed setting due to the lack of a trusted party. The latter, referred to as local DP, is preferred in the distributed setting, especially under the strong threat model where no party can be trusted. Instead of perturbing the analysis results, local DP perturbs the individual data records with locally injected noises, which often accumulate to a much larger one, leading to *low utility*.

A more effective alternative to local DP is the distributed DP framework, which combines MPC protocols with locally injected noises [39]–[42]). Distributed DP injects a far lower amount of noise (meaning higher result utility) than local DP, while providing a strong privacy guarantee. The overall idea of distributed DP is to run some cryptography protocol (e.g., [43]) to simulate the role of a trusted party, who could perform the analysis and noise injection over the partitioned database in a centralized fashion. The problem, however, is that existing distributed DP solutions are limited to horizontally partitioned FL (or HFL) where the analysis function is linear (e.g., the sum of gradients computed from different parties [42], [44], [45]); to our knowledge, there does not yet exist an effective distributed DP mechanism for VFL.

We also emphasize that naively injecting random noises into the analysis results on vertically partitioned databases (or VFL) does not preserve DP automatically. First, the process of computing the result, which requires collaboration among parties, may not be private in the first place (if the parties share their sensitive data in plain text). In this case, adding noise to the result does not fix the privacy issue. Second, even when the computation process is protected (e.g., using MPC), letting an arbitrary party perform the noise injection also leads to privacy violation, as the party himself could launch data extraction attacks such as [18], [19] on the noise-less result.

Consequently, practitioners are left with the very basic local DP approaches to enforce DP in a VFL setup. Hence, the main research question in this paper is: can we satisfy differential privacy in a VFL setup, and achieve high model utility comparable to the centralized setting, without relying on any trusted party?

Challenges. Designing an effective distributed DP mechanism for VFL is a complex task, which involves overcoming several major challenges. The approach should align with the principles of distributed DP in HFL, utilizing an MPC protocol to emulate a trusted party for analysis and noise injection, thereby preventing the accumulation of excessive locally injected noise.

The first challenge is the precision issue. In particular, existing DP mechanisms (e.g., the classic Laplace and Gaussian mechanisms [21], [46]) are mostly designed under the continuous domain that requires infinite precision to represent, whereas MPC protocols only support finite precisions over discrete fields [47] or floating-point numbers [48]. Consequently, naive implementations of (approximate) continuous DP mechanisms on discrete systems may lead to privacy violations, as pointed out in [44], [49], [50].

Indeed, discrete counterparts of continuous DP mechanisms (e.g., the discrete Gaussian mechanism [51]) are proposed to resolve the precision issue. Extending this to the VFL setting, however, is highly non-trivial. First, securely sampling discrete noises using MPC is a complicated task, and existing solutions often incur privacy overhead or utility degradation, according to [52], [53]. Second, sampling noises using MPC are also vulnerable to timing attacks, leading to privacy violations–as the outcome of the sampled noise is highly correlated to the running time of the sampling protocol [49].

The above precision issue also complicates the sensitivity analysis, which is critical for establishing rigorous DP guarantees. As per the IEEE standard [54], results of numerical computation are required to be exactly rounded, which may lead to underestimation of the sensitivity and, thus, privacy violations. In the centralized DP setting, this sensitivity issue can be resolved by letting the trusted data curator enforce an artificial clipping on the computation result to ensure that the sensitivity is bounded by some target constant (e.g., gradient clipping in [55]). In VFL, however, there does not exist a trusted data curator who can perform such operations. Since the database is vertically partitioned, computing the gradient of an individual record requires the collaboration of multiple parties.

In this paper, we make a solid step towards secure, high-utility VFL with distributed differential privacy[3], by proposing an effective mechanism that is applicable when the function of interest (e.g., the model fitting process) can be expressed as a polynomial with respect to the data. In particular, we demonstrate that our solution applies to classical data analytics tasks such as principal component analysis (PCA) and logistic regression (LR) and achieves result utility comparable to a centralized DP setting.

Solution. Overall, in our solution, the participants utilize a general-purpose MPC protocol to evaluate the target polynomial function with integer-valued DP noise injected, as illustrated in Figure 2. In what follows, we outline the main ideas for addressing the aforementioned challenges.

First, to avoid the aforementioned precision issue, both the function evaluation and noise injection are performed over integers. This is done by requiring each party to pre-process its local data (in private) into integers; then, subsequent computations are performed over the discrete integer domain.

[3]The original framework of distributed DP [41] is to securely shuffle locally perturbed data to amplify privacy. Here, we extend its meaning to achieving centralized-DP-like privacy-utility trade-off in distributed settings without assuming trusted parties.

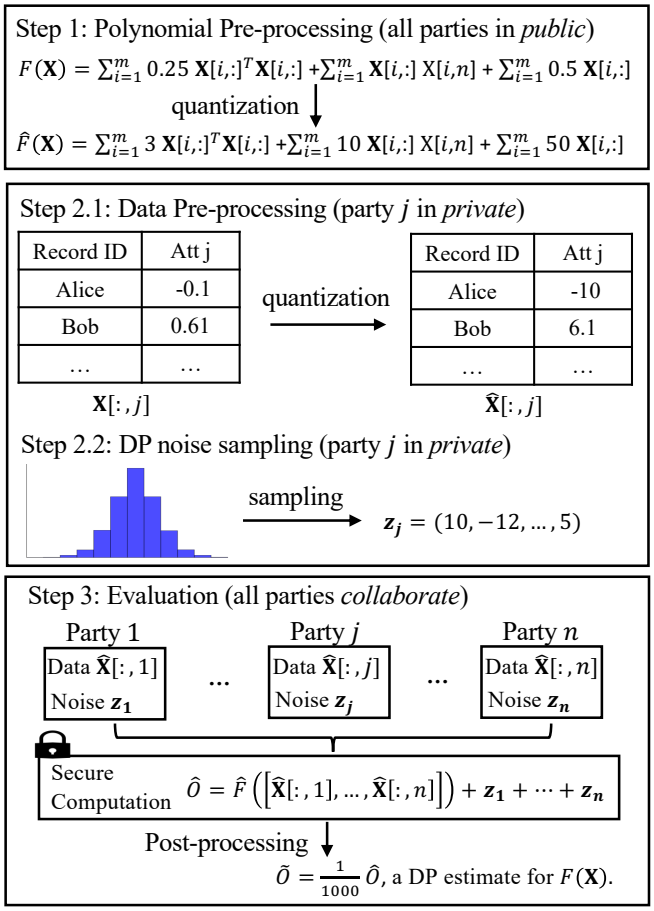

Fig. 2: The overview of our solution for evaluating polynomial functions over vertically partitioned datasets with differential privacy (numbers are for illustration purposes).

However, pre-processing the original data to integers could increase the sensitivity of the analysis function as the input domain expands. For instance, a value of 0.01 may be rounded to 1, leading a 100x sensitivity increase and, thus, asignificantly higher amount of random noise to satisfy DP. Note that although the above problem is also present for distributed DP for FL with horizontally partitioned databases (i.e., HFL), existing solutions for such a setting (e.g., [42], [44], [45]) do not apply to our problem, as they are mostly limited to estimating the linear sum of private data items where each item is possessed by a single party, who, in turn, can control the sensitivity of the rounded data item through clipping. On the contrary, as we need to evaluate polynomials of the inputs in the VFL setting, allowing a single party to process the sensitive (intermediate) outcome lead to privacy violations in the first place.

We address this problem through local quantization, in which each party up-scales the original inputs by a large factor before rounding (see Step 2.1 in Figure 2). The intuition is that the impact due to rounding the scaled inputs is negligible compared to the scaling itself. Note that scaling the inputs does not affect the relative signal-noise ratio as one can simply multiply the scale of the noise to be injected accordingly, followed by a post-processing step to down-scale the final outcome (see the bottom of Step 3 in Figure 2). We apply this idea to evaluating one-dimensional polynomials over a vertically partitioned dataset, and show that the sensitivity and error overhead due to quantization approach 0 as we increase the scaling factor.

However, the above idea does not perform well when computing a general multi-dimensional polynomial, where each dimension is a polynomial containing monomials of different degrees, since scaling the inputs would lead to high overall sensitivity due to different scaling amplitudes for the monomials of different degrees. We tackle this situation by ensuring that each monomial is scaled by (roughly) the same factor regardless of its original degree. To do that, we accurately account for the overall amplification factors of every monomial and then compensate for their differences by multiplying them by different factors. This multiplication is done as a pre-processing step on the constant coefficients of the polynomial to be evaluated, which does not incur additional privacy cost, since the description of the polynomial is public (see Step 1 in Figure 2). Overall, the above local quantization allows us to obtain a tight sensitivity analysis over the quantized data, and the sensitivity overhead it introduced approaches 0 as we increase the quantization granularity.

Next, we need to design an effective noise generation algorithm to satisfy DP. Specifically, we require that the overall DP noise follow the Skellam distribution (with values from $\{0, \pm 1, \pm 2, \ldots\}$), which is known to have comparable privacy-utility trade-off as the continuous Gaussian noise [42], [45]. To do so, we let each party to privately sample a "share" of Skellam noise locally (see Step 2.2 in Figure 2). After performing local noise generation and data quantization, the parties collaboratively use MPC to evaluate the function of interest on the quantized data and, at the same time, inject the locally sampled noises into the outcome (see Step 3 in Figure 2). Since Skellam is closed under summation the overall noise injected into the outcome still follows a Skellam distribution. As the overall DP noise is the aggregation of n local noises contributed by different parties, no single party is able to break the privacy guarantees even though she/he knows the outcome of one local noise. Furthermore, the generation of DP noise is also robust against the timing attack, as each party can sample the local noise prior to participating in the MPC protocol.

Contribution. We call our solution the Skellam Quantization Mechanism (SQM), a generic mechanism for evaluating polynomials over a vertically partitioned dataset. The main techniques of SQM, as mentioned above, are the local quantization and the Skellam noises described above. In this work, our main technical contribution is to prove that SQM matches the asymptotic privacy-utility trade-off of the centralized setting without relying on any trusted party–the sensitivity and error overhead due to quantization converge to zero as the quantization granularity gets finer. This is the first of its kind in the literature. We then instantiate SQM on two classical tasks, principal component analysis and logistic regression. Our theoretical analysis and numerical experiments

on various real-world datasets show that SQM indeed achieves comparable performance as the central-DP baseline, provided with some level of quantization granularity (in our experiments, only fewer than 16 bits are used to represent each dimension of the data).

We also note our SQM is an MPC-friendly framework in the sense that the clients only need to utilize MPC as a black box (we are not creating any new MPC protocol specific to VFL) after careful quantization of the target function and the partitioned data and the local generation of DP noises. This overhead incurred during the data quantization and noise generation on the client side is negligible, compared with the native cost of running cryptographically secure protocols that are needed in the first place to keep the local data unseen during the collaboration among parties (e.g., see [3]).

II. Preliminaries

Notations. For a positive integer $n \in \mathbb{Z}^+$, we use $[n]$ to represent the set of integers $\{1, \ldots, n\}$. We denote an n-dimensional row vector as $\mathbf{v} = (v[1], \ldots, v[n])$, where $v[i]$ is the i-th element of $\mathbf{v}$ for each $i \in [n]$. Similarly, we denote an m-dimensional column vector as $\mathbf{u} = (u[1], \ldots, u[m])^T$. For an $m \times n$ matrix $\mathbf{X}$, we use $X[i\,, j]$ to represent the j-th element on the i-th row of $\mathbf{X}$. We use $\mathbf{X}[i\,, :]$ and $\mathbf{X}[:\,, j]$ to represent the i-th row and j-th column of $\mathbf{X}$, respectively. Accordingly, a matrix X can be viewed as a row vector $\mathbf{X} = (\mathbf{X}[:\,, 1], \ldots, \mathbf{X}[:\,, n])$.

Polynomials and Monomials. A polynomial is a mathematical expression consisting of variables and coefficients, which involves only the operations of addition, subtraction, multiplication, and positive-integer powers of variables. An example polynomial on $\mathbf{x} = (x[1], x[2], x[3])$ is $f(\mathbf{x}) = (x[1])^3 + 1.5 \cdot x[2]x[3] + 2$. A monomial (also called a power product) is a polynomial with only one term. The degree of a monomial is defined as the number of multiplications between variables, and the degree of a polynomial is defined as the highest degree of its monomials. In the above example, the degree of $f(\mathbf{x})$ is 3.

A. Differential Privacy

Differential Privacy (DP) [21] is a rigorous framework for quantifying data privacy, which measures indistinguishability of the output distributions for a randomized mechanism on neighboring databases. $\mathbf{X}$ and $\mathbf{X}'$ are called neighboring databases if they differ by one record (written as $\mathbf{X} \sim \mathbf{X}'$). If we see $\mathbf{X}$ and $\mathbf{X}'$ as matrices (namely, rows of attribute vectors), then neighboring $\mathbf{X}$ and $\mathbf{X}'$ differ by one row. A classic DP definition is (ϵ, δ)-DP.

Definition 1 ((ϵ, δ)-DP [21])**.** *Mechanism $\mathcal{M}$ satisfies (ϵ, δ)-DP if for any neighboring databases $\mathbf{X} \sim \mathbf{X}'$ from the input domain and any set of outputs $\mathcal{O}$, it holds that*

$$\Pr[\mathcal{M}(\mathbf{X}) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(\mathbf{X}') \in \mathcal{O}] + \delta. \quad (1)$$

Here a smaller ϵ or δ implies that it is more difficult to distinguish the distributions with neighboring inputs $\mathbf{X}$ and $\mathbf{X}'$, which provides stronger individual privacy. A common alternative DP definition is Rényi DP (RDP) [56], which quantifies indistinguishability under Rényi divergence.

Definition 2 (Rényi DP [56])**.** *Mechanism $\mathcal{M}$ satisfies (α, τ)-RDP for some $\alpha \in (0, 1) \cup (1, \infty)$ if for any neighboring databases $\mathbf{X} \sim \mathbf{X}'$ it holds that*

$$D_\alpha(\mathcal{M}(\mathbf{X}) \,\|\, \mathcal{M}(\mathbf{X}')) \leq \tau,$$

where $D_\alpha(P \,\|\, Q)$ denotes the Rényi divergence of P from Q.

Given a function of interest F, a canonical approach to make it differentially private is to inject random noise (perturbation) into its output. The scale of the noise is calibrated to the sensitivity of the mechanism [21], denoted as $S(F)$.

$$S(F) = \max_{\mathbf{X} \sim \mathbf{X}'} \|F(\mathbf{X}) - F(\mathbf{X}')\|, \quad (2)$$

where the maximum is taken over all pairs of neighboring databases, and $\|\cdot\|$ represents a norm measure, e.g. $\mathcal{L}_2$-norm.

Lemma 1 (Gaussian noise satisfies RDP [56])**.** *Injecting Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$ to any d-dimensional function with bounded $\mathcal{L}_2$ sensitivity Δ_2 achieves $(\alpha, \frac{\alpha \Delta_2^2}{2\sigma^2})$-RDP.*

B. Secure Multiparty Computation

Secure multiparty computation (MPC) [57] coordinates multiple clients to jointly compute a function without revealing any information on their private inputs except for the result of the function. As an effective method to avoid relying on a trusted party, MPC has been widely applied in federated learning (e.g., see [43], [58]) and interested readers are referred to [59] for more background.

Although MPC ensures the confidentiality of the inputs, the output of the computed function may still leak private information [18], [19]. This motivates the combination of DP and MPC in FL, to ensure that both the computation process and the outcome preserve individuals' privacy. Realizing this idea, as we have mentioned in Section I, is non-trivial, due to the fact that classic DP mechanisms involve sampling noise from a continuous domain, whose distribution is not preserved in the MPC process in which participants communicate through discrete channels, leading to privacy violations [49], [50].

In our solution SQM, we adopt the classic BGW protocol [60] to securely compute any polynomial function, rather than a specific one. BGW has mature and efficient implementations, as reported in [61]. Our SQM invokes BGW as a black box to perform the computations securely (as we will see, we can enforce DP during this process). In the actual implementation, one can replace BGW with any other MPC protocol as long as it allows the parties to compute a polynomial without revealing their inputs, e.g., using the SPDZ framework [47], without affecting the DP guarantees.

C. Integer-Valued DP Noise

Integer-valued noises for enforcing DP are compatible with MPC. The state-of-the-art is the Skellam noise [42], [45], which is obtained as the difference between two independent

Poisson variables of the same parameters. We write $\mathbf{Z} \sim Sk^d(\mu)$ if $\mathbf{Z}$ is obtained as $\mathbf{U} - \mathbf{V}$, where both $\mathbf{U}$ and $\mathbf{V}$ are independently sampled from $Pois^d(\mu)$, which is the ensemble of d independent $Pois(\mu)$.

Lemma 2 (Skellam noise preserves RDP [42], [45])**.** *Injecting* $\mathrm{Sk}^d(\mu)$ *to the outcome of any* d*-dimensional integer-valued function* F *with bounded* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *sensitivities* Δ_1 *and* Δ_2 *satisfies RDP. For any integer* $\alpha > 1$*, we have*

$$\sup_{\mathbf{X}\sim\mathbf{X}'} D_\alpha \left(F(\mathbf{X}) + \mathrm{Sk}^d(\mu) \,\|\, F(\mathbf{X}') + \mathrm{Sk}^d(\mu)\right) \leq \frac{\alpha}{2}\cdot\frac{\Delta_2^2}{2\mu} + \min\left(\frac{(2\alpha-1)\Delta_2^2 + 6\Delta_1}{16\mu^2}, \frac{3\Delta_1}{4\mu}\right). \quad (3)$$

Comparing Lemma 2 (ignoring the min term which is often dominated by the preceding one) with Lemma 1, we see that Skellam noise almost achieves the same level of privacy-utility trade-off as the continuous Gaussian noise [42].

III. Problem Formulation

We consider federated learning over a vertically partitioned database where the input database $\mathbf{X}$ is a collection of $m = |\mathbf{X}|$ records from the domain $\mathbb{R}^n$. For each record $\mathbf{x} \in \mathbf{X}$, we assume $\|\mathbf{x}\|_2 \leq c$ for some constant c. Database $\mathbf{X}$ is partitioned among multiple data owners (referred to as *clients* in the following) by attributes, i.e., columns. Without loss of generality, we assume there are n clients in total, and client j possesses the j-th column of $\mathbf{X}$, denoted as $\mathbf{X}[:, j]$. An untrusted coordinator (called *server* in the following) aims to evaluate a function of interest, denoted as F, on $\mathbf{X}$. Here F is an aggregate function for the individual records of the input, i.e., for any input private database $\mathbf{X}$, we have

$$F(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}), \quad (4)$$

for some function $f : \mathbb{R}^n \to \mathbb{R}^d$, where d is the dimensionality of the function's output. In this work, we consider the case when f is a polynomial function. For example, the covariance matrix of the input can be expressed as $\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$ with $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$.

We aim to design a general solution $\mathcal{M}$ for evaluating any given polynomial function f with low error, while preserving differential privacy with respect to the clients' data (formalized next). We restrict the problem domain to traditional tasks. However, we would like to argue that our problem is general in the sense that polynomials can be used to approximate various functions (including the activation functions in deep learning models). One recent example can be found in [62], where polynomial approximations for GELU (Gaussian Error Linear Unit) and Tanh (hyperbolic tangent function) are used for efficient and private inference on transformers. In this work, we focus on presenting positive results on the traditional task of PCA and LR (matching the privacy-utility trade-offs in central DP) and leave the more advanced tasks as future work.

A. Privacy Requirements

We consider the semi-honest threat model. Specifically, the adversary follows the specification of mechanism $\mathcal{M}$ while trying to infer the private inputs of other parties. Here, the adversary could be either the server or a client j ($j \in [n]$), and its only adversarial behavior is to conduct a computation based on the observation it has received and its own input. These assumptions have been commonly adopted in the distributed DP literature [44], [45]. In what follows, we formalize the privacy requirements under RDP; the classical (ϵ, δ)-DP can be obtained with a standard conversion [51] from RDP. We consider two possible adversaries, curious servers and curious clients.

We say $\mathcal{M}$ is DP against both adversaries if for any neighboring databases $\mathbf{X}$ and $\mathbf{X}'$ and the corresponding observations by the server, denoted as $O \sim \mathcal{M}_{\text{server}}(\mathbf{X})$ and $O' \sim \mathcal{M}_{\text{server}}(\mathbf{X}')$ respectively, and the corresponding observations by the any client j, denoted as $\mathcal{M}_{\text{client}_j}(\mathbf{X})$ and $\mathcal{M}_{\text{client}_j}(\mathbf{X}')$ respectively,

$$\sup_{\mathbf{X}\sim\mathbf{X}'} D_\alpha(\mathcal{M}_{\text{server}}(\mathbf{X}) \,\|\, \mathcal{M}_{\text{server}}(\mathbf{X}')) \leq \tau_{\text{server}}, \text{ and} \quad (5)$$

$$\sup_{\mathbf{X}\sim\mathbf{X}'} D_\alpha(\mathcal{M}_{\text{client}_j}(\mathbf{X}) \,\|\, \mathcal{M}_{\text{client}_j}(\mathbf{X}')) \leq \tau_{\text{client}} \quad (6)$$

hold for some $\alpha > 1$ and some finite positive τ_{client} and τ_{server}. Accordingly, we say $\mathcal{M}$ satisfies $(\alpha, \tau_{\text{server}})$ server-observed RDP and $(\alpha, \tau_{\text{client}})$ client-observed RDP.

Since client j naturally knows more information about the input than the server, τ_{client} may be larger than τ_{server}. For example, the server may not know the overall number of records in the inputs, whereas such information is not considered private for a client–every client j has full access to the corresponding private portion $\mathbf{X}[:, j]$, including the information for identifying an individual record (e.g., name, ID, and address).

We emphasize that a mechanism that is DP against a curious server is not necessarily DP against a curious client, and vice versa, motivating us to take both Eq. (5) and Eq. (6) into consideration for the privacy requirements.

Proposition 1. *A mechanism that satisfies Eq. (5) can violate Eq. (6) (i.e., non-private for a curious client). Similarly, a mechanism that satisfies Eq. (6) can violate Eq. (5) (i.e., non-private for a curious server).*

We sketch the proof by construction. For the first statement, we let client j^* collect the private data from all clients, compute the target function, perturb the outcome with DP noise, and then release the perturbed outcome to the server. This mechanism satisfies DP against the server, but not against client j^*. For the latter statement, we let the server collect the data from all clients, perform the computation and noise injection, and then broadcast the outcome to all clients. This mechanism satisfies DP against all clients, but not against the server.

We assume there is no trusted third party; otherwise, there is little motivation for protecting data privacy on VFL in the first place and we can easily construct a mechanism that achieves a central-DP-like privacy-utility trade-off as in the above proof (let the trusted party collect all partitions and then run any

central-DP mechanism on the whole dataset). Due to this reason, solutions such as [63], [64] do not apply to our problem.

B. A Baseline Solution

Regardless of the function of interest, there is a local DP solution that achieves our privacy requirements. The idea is to let each client j first independently perturb her local data portion with random Gaussian noise, and then share the perturbed outcome with the server, who then constructs the whole perturbed database $\tilde{\mathbf{X}}$ and evaluates the function on $\tilde{\mathbf{X}}$. The privacy guarantee follows from the additive Gaussian noise and that post-processing preserves DP.

Unfortunately, local DP solutions often requires a large amount of noise to satisfy DP, leading to poor result utility [40], [44]. Intuitively, with local DP, each element in $\tilde{\mathbf{X}}$ is perturbed with $O(1)$ error, and the error accumulates when evaluating an aggregate function on individual records of $\tilde{\mathbf{X}}$. For instance, consider the covariance matrix estimation with $F(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}^T\mathbf{x}$. Since each element in $\mathbf{x}$ is independently perturbed with a Gaussian noise, each element of the outer product $\mathbf{x}^T\mathbf{x}$ incurs an error of $O(1)$. Since there are m such outer products, the overall error becomes $O(m)$, far higher than the centralized DP setting where the injected noise is of the same scale as the sensitivity of the covariance matrix, which is constant.

IV. Proposed Solution

In this section, we present the general Skellam Quantization Mechanism, a generic distributed DP framework for evaluating any polynomial function on VFL [4].

A. SQM for One-Dimensional Monomials

We first consider any given one-dimensional monomial function $f : \mathbb{R}^n \to \mathbb{R}$ of degree $\lambda \geq 1$, expressed as $f(\mathbf{x}) = \Pi_{j=1}^n (x[j])^{\lambda[j]}$ with $\sum_{j=1}^n \lambda[j] = \lambda$. The goal is to compute $F(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$ while satisfying Eq. (5) and (6). Without loss of generality, we assume that the coefficient of f is 1, since otherwise, the server could post-process the outcome to obtain the result corresponding to any given coefficient, without affecting DP. We outline the proposed Skellam quantization mechanism (SQM) as in Algorithm 1.

SQM consists of three major steps: **(1)** data quantization, which processes the private data to integers, setting up the stage for enforcing DP over the integer domain; **(2)** Skellam noise sampling, in which each client samples a Skellam noise privately; and **(3)** function evaluation and perturbation in which all clients collaboratively compute the target function on the processed data with the aggregate of the locally-generated Skellam noises injected. We go through each step in more detail in the following.

Data quantization. Each client j independently processes her/his local data $\mathbf{X}[:, j]$ using Algorithm 2 (Lines 1-2 in Algorithm 1). In particular, each real-valued vector $\mathbf{v} \in \mathbb{R}^m$ (i.e., a column of $\mathbf{X}$ that is possessed by a single client) is first scaled to $\gamma\mathbf{v}$ and then each dimension of the scaled vector is randomly rounded to one of its nearest integers by flipping a coin. The outcome, denoted as $\hat{\mathbf{v}}$ is in expectation equal to $\gamma\mathbf{v}$. The processed data portions of all clients constitute database $\hat{\mathbf{X}} = (\hat{\mathbf{X}}[:, 1], \ldots, \hat{\mathbf{X}}[:, n])$.

Noise sampling. Next, each client j independently samples $\mathbf{Z}_j \sim Sk(\frac{\mu}{n})$ (Lines 3-4 in Algorithm 1). Both $\hat{\mathbf{X}}[:, j]$ and $\mathbf{Z}_j$ are kept private by client j.

Function evaluation. The clients then collectively run the BGW protocol (Line 5 in Algorithm 1) to compute

$$\hat{y}(\mathbf{X}) = \sum_{\hat{\mathbf{x}}\in\hat{\mathbf{X}}} \hat{f}(\hat{\mathbf{x}}) + \sum_{j=1}^n \mathbf{Z}_j. \tag{7}$$

The result $\hat{y}(\mathbf{X})$ is then sent to the server (Line 6). After that, the server computes $\tilde{y} = \frac{1}{\gamma^\lambda}\hat{y}(\mathbf{X})$ as the estimate for $\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$ (Line 7 in Algorithm 1). This post-processing step does not affect the privacy guarantee of computing $\hat{y}$.

Note that both the processed data portion $\hat{\mathbf{X}}[:, j]$ and the sampled noise Z_j are the private inputs of client j to the BGW protocol. Essentially, the BGW protocol evaluates a surrogate function defined over $\mathbb{R}^{(m+1)\times n}$, where the extra row represents the locally generated DP noise. From the server's perspective, $\hat{y}$ is perturbed with $Z \sim Sk(\mu)$, the aggregate of the local Skellam noises, thus avoiding the large noise overhead of local DP. For a client j, the outcome $\hat{y}$ is perturbed with $Z^* \sim$

Algorithm 1: SQM for One-dimensional Monomials

Input: One-dimensional monomial function f of degree λ; private input database $\mathbf{X}$ that is partitioned among n clients; scaling parameter γ; noise parameter μ.

1 **for** *each client* $j \in [n]$ **do**
2 $\quad \hat{\mathbf{X}}[:, j] \leftarrow$ Algorithm 2 $(\mathbf{X}[:, j], \gamma)$. // the processed data portions constitute database $\hat{\mathbf{X}}$.
3 **for** *each client* $j \in [n]$ **do**
4 $\quad$ Sample $Z_j \sim Sk\left(\frac{\mu}{n}\right)$. // noise sampling
5 The clients collectively run the BGW protocol to evaluate

$$\hat{y}(\mathbf{X}) = \sum_{\hat{\mathbf{x}}\in\hat{\mathbf{X}}} \hat{f}(\hat{\mathbf{x}}) + \sum_{j=1}^n Z_j.$$

6 The clients share the outcome $\hat{y}(\mathbf{X})$ to the server.
7 The server compute $\tilde{y} \leftarrow \frac{1}{\gamma^\lambda} \cdot \hat{y}(\mathbf{X})$.

Output: $\tilde{y}$ as the estimate for $\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$.

Algorithm 2: Data Pre-processing/Quantization

Input: Real-valued vector $\mathbf{v} \in \mathbb{R}^m$; scaling factor γ.

1 $\hat{\mathbf{v}} \leftarrow \gamma \cdot \mathbf{v}$.
2 **for** *Dimension* $j \in [m]$ **do**
3 $\quad$ Flip a coin with heads probability $\hat{\mathbf{v}}[j] - \lfloor\hat{\mathbf{v}}[j]\rfloor$.
4 $\quad$ **if** *Heads* **then**
5 $\quad\quad \hat{\mathbf{v}}[j] \leftarrow \lfloor\hat{\mathbf{v}}[j]\rfloor + 1$.
6 $\quad$ **else**
7 $\quad\quad \hat{\mathbf{v}}[j] \leftarrow \lfloor\hat{\mathbf{v}}[j]\rfloor$.

Output: Integer-valued vector $\hat{\mathbf{v}} \in \mathbb{Z}^m$.

[4]The detailed proofs of our claims can be found in the full version [65].

$Sk\left(\frac{n-1}{n}\mu\right)$ (since she knows the outcome of her local noise) which is close to $Sk(\mu)$ when n is large. Hence, intuitively our privacy requirements are satisfied with the injected aggregate noise.

Next, we show that by making γ large enough, the approximation error, as well as the overhead in DP noise due to rounding, becomes negligible. In other words, SQM achieves asymptotically comparable privacy-utility trade-off in VFL as in the centralized setting, for evaluating one-dimensional monomial functions. We will use the asymptotic notation to illustrate the idea in the rest of this subsection; constant factors will appear in the instantiation of SQM on PCA and logistic regression in Section V. We first analyze the error due to rounding and neglect the requirement of privacy for the moment (i.e., setting μ to 0).

Lemma 3. *Given any monomial function of degree λ, for any single-record input $\{x\}$ ($\|x\|_2 \le c$) to Algorithm 1 with scaling parameter $\gamma \ge 100\lambda$ and noise parameter $\mu = 0$, Algorithm 1 satisfies $\hat{y}(\{\mathbf{x}\}) - \gamma^\lambda f(\mathbf{x}) = O(\gamma^{\lambda-1})$, and hence, $\frac{1}{\gamma^\lambda}\hat{y}(\{\mathbf{x}\}) - f(\mathbf{x}) = o(1)$.*

The key argument to prove Lemma 3 is that since λ multiplicands are scaled by γ first, the additive error of 1 (which causes a constant difference) on each multiplicand introduced by rounding becomes negligible compared with the scaling factor of γ^λ, when γ is large enough (i.e., when quantization is fine-grained enough).

Lemma 3 implies the following error guarantee.

Corollary 1 (Error guarantee of quantization)**.** *Consider input database $\mathbf{X}$ such that each $\mathbf{x} \in \mathbf{X}$ satisfies $\|\mathbf{x}\|_2 \le c$, one-dimensional monomial f with degree $\lambda \ge 1$, scaling parameter $\gamma \ge 100\lambda$, and noise parameter $\mu = 0$. Algorithm 1 satisfies $\tilde{y} - \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}) = o(1)$, meaning that the approximation error approaches 0 as we increase the scaling parameter γ.*

We can then use the triangle inequality to show that the sensitivity overhead of evaluating f on the quantized data becomes negligible when γ is large, leading to the following privacy guarantees.

Lemma 4 (Privacy guarantees)**.** *Algorithm 1 for monomial function f with degree $\lambda \ge 1$, scale parameter γ, and noise parameter μ satisfies (α, τ_{server}) server-observed RDP and (α, τ_{client}) client-observed RDP with*

$$\tau_{server} = \frac{\alpha\Delta^2}{4\mu} + \frac{3\Delta}{4\mu}, \text{ and } \tau_{client} = \frac{\alpha n\Delta^2}{(n-1)\mu} + \frac{3n\Delta}{2(n-1)\mu},$$

where $\Delta = \gamma^\lambda \max_{|x|_2 \le c} |f(x)| + O(\gamma^{\lambda-1})$.

We sketch the computation for τ_{server}. The computation for τ_{client} that corresponds to client-observed DP can be proved similarly while noting two differences. The first is that the sensitivity is doubled. This is because for a client, the number of records is regarded as public information, and neighboring databases are obtained by replacing a record rather than adding/removing one. In addition, the scale of DP noise μ is replaced by $\frac{n-1}{n}\mu$, since a client knows the outcome of the local noise she has generated. Considering $\mathbf{X}' = \mathbf{X} \cup \{\mathbf{x}\}$, without loss of generality, we have that the maximum difference in computing $\hat{y}(\mathbf{X})$ and $\hat{y}(\mathbf{X} \cup \{\mathbf{x}\})$ is at most $\gamma^\lambda |f(x)| + O(\gamma^{\lambda-1})$, due to the triangle inequality. The result of τ_{server} then follows from Lemma 2.

Privacy-utility trade-off. How good is our SQM? We follow prior work on HFL [42], [44] and compare the classic central-DP Gaussian mechanism with ours under server-observed DP; it is unfair to consider the client-observed DP here since in the central-DP setting the adversarial often has no information of m or the DP noise. To answer this question, it suffices to compare the errors of the two mechanisms. In Lemma 4, the term $\frac{3\Delta}{4\mu}$ is often dominated by $\frac{\alpha\Delta^2}{4\mu}$ when γ is large (i.e., under fine-grained quantization). Hence, for SQM to achieve (α, τ)-RDP against the server, it suffices to set μ to roughly $\frac{\alpha}{2} \cdot \frac{\gamma^{2\lambda} \max |f(\mathbf{x})|^2 + O(\gamma^{2\lambda-1})}{2\tau}$. Here, this $O(\gamma^{2\lambda})$ increase is reduced to 1 after the post-processing by the server (Line 7 in Algorithm 1). Besides, the approximation error over the quantized data is $o(\gamma^\lambda)$, which becomes negligible after the server's post-processing. As a result, the overall noise variance in $\tilde{y}$ becomes $\frac{\alpha}{2} \cdot \frac{\max |f(\mathbf{x})|^2 + o(1)}{2\tau}$, which is comparable to the Gaussian mechanism's variance $\frac{\alpha}{2} \cdot \frac{\max |f(\mathbf{x})|^2}{2\tau}$ (see Lemma 1); and the $o(1)$ error can be made arbitrarily small by increasing γ. In conclusion, our SQM achieves a comparable privacy-utility trade-off on VFL as the centralized DP mechanism.

B. SQM for Multi-dimensional Polynomials

Next, we generalize SQM to the more general case where the function of interest f is a multi-dimensional polynomial. Given input $\mathbf{x}$, we can write $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_d(\mathbf{x}))$, where each $f_t(\mathbf{x})$ is a one-dimensional polynomial of order λ_t. We can write $f_t(\mathbf{x})$ as

$$f_t(\mathbf{x}) = \sum_{l=1}^{v_t} a_t[l] \cdot \Pi_{j=1}^n x[j]^{B_t[l,j]}, \tag{8}$$

for some constant $v_t \ge 1$. Here $\mathbf{a}_t = (a_t[1], \ldots, a_t[v]) \in \mathbb{R}^v$ represents the coefficient vector for all v monomials, and $a_t[l] \in \mathbb{R}$ represents the coefficient for the l-th monomial component, for the t-th dimension. Accordingly, matrix $\mathbf{B}_t \in \mathbb{N}^{v\times n}$ represents the exponents, and $B_t[l, j]$ is the exponent for $x[j]$ in the l-th monomial, for the t-th dimension.

Challenge. It is tempting to run Algorithm 1 independently for each monomial (indexed by l) in each dimension (indexed by t) independently (i.e., evaluating $a_t[l] \cdot \Pi_{j=1}^n x[j]^{B_t[l,j]}$ independently). However, since the monomials may be of different degrees (i.e., $\sum_{j=1}^n B_t[l, j]$ could be different for different l's and t's). Applying the same scaling factor γ to each term would result in different scaling factors for different monomials (i.e., different powers of γ), making the overall sensitivity difficult to analyze. For example, the monomial $\frac{1}{2}x^2$ would be amplified by a different factor than the monomial x. One might suggest computing and perturbing components of different degrees separately instead of the whole function. For example, the term $\frac{1}{2}x^2$ (resp. x) is computed and perturbed by the clients using BGW and then sent to the server, which

down-scales the outcome by $\frac{1}{\gamma^3}$ (resp. $\frac{1}{\gamma}$). This approach, however, leads to increased sensitivity, as the sensitivities for different components are analyzed separately, and their worst-case sensitivities may correspond to different inputs; hence, they constitute a pessimistic upper bound for the overall sensitivity of the original function.

Main Idea. The proposed solution aims to ensure that each monomial in each dimension is scaled by the same factor regardless of its original degree, so that we can utilize our established analysis (Lemma 3 and Corollary 1) to quantify the overall sensitivity for computing the d-dimensional polynomial as a whole. Specifically, our solution first identifies the monomial of the largest degree among all dimensions. Without loss of generality, we denote this degree as λ, and refer to it as the degree of the d-dimensional polynomial f. For each private input $x[i, j]$, it is scaled by some factor $\gamma > 1$ and then randomly rounded to its nearest integers, using Algorithm 2. This operation would result in different scaling factors for monomials of different degrees.

To compensate for such differences, for each coefficient $a_t[l]$ for the l-th monomial in the t-th dimension of f, we scale it by $\gamma^{1+\lambda-\sum_{j=1}^{n} B_t[l\,,j]}$ and then round it to its nearest integer, using Algorithm 2. As a result, the processed coefficient for any monomial of degree λ is roughly γ times larger than its original value; the processed coefficient for any monomial of degree $\lambda - 1$ is roughly γ^2 times larger than its original value; and so on. Overall, the result for evaluating each monomial component is roughly $\gamma^{\lambda+1}$ times larger than its original value, regardless of its original degree. Accordingly, during post-processing, the server multiplies the result by $\frac{1}{\gamma^{\lambda+1}}$.

With slight modifications on Lemma 3 and Corollary 1, we can show that the approximation error for each monomial term is still in $o(1)$ when γ is large enough. By linearity, the overall error and sensitivity overhead of SQM for evaluating any d-dimensional polynomial is negligible compared with the continuous Gaussian mechanism. We outline the complete procedure in Algorithm 3. Compared with Algorithm 1, the main difference is in Lines 1-3 of Algorithm 3, where the coefficients are processed using Algorithm 2 with different scaling parameters for monomials of different degrees (the result is publicly known to all clients), and in Line 11, where the server down-scales the outcome by $\frac{1}{\gamma^{\lambda+1}}$ instead of $\frac{1}{\gamma^{\lambda}}$. Next, we provide the privacy analysis for Algorithm 3.

Lemma 5. *Algorithm 3 for any d-dimensional polynomial function f with degree $\lambda \geq 1$, scale parameter γ, and noise parameter μ satisfies (α, τ_{server}) server-observed RDP and (α, τ_{client}) client-observed RDP with*

$$\tau_{server} = \frac{\alpha\Delta_2^2}{4\mu} + \frac{3\Delta_1}{4\mu}, \text{ and } \tau_{client} = \frac{\alpha n\Delta_2^2}{(n-1)\mu} + \frac{3n\Delta_1}{2(n-1)\mu},$$

where $\Delta_2 = \gamma^{\lambda+1} \max \|f(x)\|_2 + o(\gamma^{\lambda+1})$, and $\Delta_1 = \min(\Delta_2^2, \sqrt{d}\Delta_2)$.

Here we use $\Delta_1 = \min(\Delta_2^2, \sqrt{d}\Delta_2)$ since the absolute value of any integer is bounded by its square and that in general, the $\mathcal{L}_1$ norm of any vector is bounded by $\sqrt{d}$ times its $\mathcal{L}_2$ norm (by Jensen's inequality). We note a slight difference between proving Lemmas 5 and 4. The sensitivity overhead for evaluating a d-dimensional polynomial function, and each dimension of it is a polynomial containing v_t monomials. This means that the overhead in terms of both error and sensitivity (recall Lemma 3 and Corollary 1) is at most $d \times \max_t v_t$ times larger than the overhead for evaluating a one-dimensional monomial, by the triangle inequality. However, such overheads are still negligible compared with the overall scaling factor $\gamma^{\lambda+1}$ when γ is large enough. Hence, for evaluating general multi-dimensional polynomials, SQM can achieve comparable utility-privacy trade-offs as the multi-dimensional continuous Gaussian in the centralized setting.

Algorithm 3: SQM for Multi-dimensional Polynomials

Input: Multi-dimensional polynomial function f of degree λ and dimension d; private input database $\mathbf{X}$ that is partitioned among n clients; scaling parameter γ; noise parameter μ.

1 **for** *each coefficient* $a_t[l]$ **do**
2 Compute $\lambda_t[l] = \sum_{j=1}^{n} B_t[l\,,j]$.
3 $\hat{a}_t[l] \leftarrow$ Algorithm 2 $(\hat{a}_t[l], \gamma^{1+\lambda-\lambda_t[l]})$. // coefficient pre-processing
4 **for** *each client* $j \in [n]$ **do**
5 $\hat{\mathbf{X}}[:\,, j] \leftarrow$ Algorithm 2 $(\mathbf{X}[:\,, j], \gamma)$. // the processed data portions constitute database $\hat{\mathbf{X}}$.
6 **for** *each dimension* $t \in [d]$ **do**
7 **for** *each client* $j \in [n]$ **do**
8 Sample $Z_j \sim Sk\left(\frac{\mu}{n}\right)$. // noise sampling
9 The clients collectively run the BGW protocol to evaluate

$$\hat{y}_t = \sum_{i=1}^{m} \sum_{l=1}^{v_t} \hat{a}_t[l] \cdot \Pi_{j=1}^{n} \hat{X}[i, j]^{B_t[l\,,j]} + \sum_{j=1}^{n} Z_j.$$

10 The clients share the outcome $\hat{y}_t$ to the server.
11 The server compute $\tilde{\mathbf{y}} \leftarrow \frac{1}{\gamma^{\lambda+1}} \cdot (\hat{y}_1, \ldots, \hat{y}_d)$.

Output: $\tilde{\mathbf{y}}$ as the estimate for $\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$.

V. Applications

In this section, we instantiate SQM on the tasks of principal component analysis and logistic regression. The detailed proofs of our claims appear in the full version [65].

A. Principal Component Analysis

Problem definition. Given an integer k (usually $k \ll n$), the server aims to (approximately) learn a rank-k subspace $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$ that preserves most of the variance in the original database $\mathbf{X}$, i.e., maximizing $\|\mathbf{X}\tilde{\mathbf{V}}\|_F^2$. We follow the classic algorithm for differentially private PCA under the centralized setting, where the principal subspace $\tilde{\mathbf{V}}$ is obtained as the top k eigenvectors from the perturbed version of the covariance matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$. Here, the polynomial of interest is $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$, which computes the outer product of record $\mathbf{x}$. The dimension of f is n^2 and the degree of f is 2.

VFL Algorithm. The clients call Algorithm 3 with scaling parameter γ and noise parameter μ to compute the noisy

covariance matrix $\tilde{C}$, and shares the outcome with the server. Here since the coefficient in each dimension of f is 1, we choose not to pre-process the coefficients. Namely, all clients independently discretize the their data with Algorithm 2 with scaling parameter γ, and then collaboratively compute the covariance matrix $\tilde{\mathbf{C}}$ of the discretized data using BGW, and share the outcome with the server. The server then computes the k-dimensional principal singular subspace (i.e., top k eigenvectors) of $\frac{1}{\gamma^2} \cdot \tilde{\mathbf{C}}$ as the estimated principal components.

Analysis. When each record $\mathbf{X}[i,:]$ has $\mathcal{L}_2$ norm bounded by a constant c, the sensitivity for computing matrix $\mathbf{C}$ is bounded by c^2. To show this, one can consider X' that is obtained by removing the m-th row from X. Then we have

$$\|\mathbf{X}^T\mathbf{X} - \mathbf{X}'^T\mathbf{X}'\|_F \leq \sum_j (\mathbf{X}[m,j])^2 \leq c^2.$$

Hence, the quantized version satisfies $\|\hat{\mathbf{X}}[i,:]\|_2 \leq \sqrt{\gamma^2 c^2 + n}$. For the sensitivity for for computing $\tilde{\mathbf{C}}$, we have

$$\|\hat{\mathbf{X}}^T\hat{\mathbf{X}} - \hat{\mathbf{X}}'^T\hat{\mathbf{X}}'\|_F \leq \sum_j \left(\hat{\mathbf{X}}[m,j]\right)^2 \leq \gamma^2 c^2 + n. \quad (9)$$

Here the additional n corresponds to the $O(\gamma^{\lambda-1})$ overhead for the sensitivity computation. Indeed, we see that n becomes negligible compared with $\gamma^2 c^2$ when γ is large enough. Combining Eq. (9) with Lemma 2, we have the DP guarantees.

Lemma 6 (Privacy analysis)**.** *With scaling parameter γ and noise parameter μ, computing the noisy covariance matrix over using Algorithm 3 satisfies (α, τ_{server}) server-observed RDP and (α, τ_{client}) client-observed RDP with*

$$\tau_{server} = \frac{\alpha\Delta_2^2}{4\mu} + \frac{3\Delta_1}{4\mu}, \text{ and } \tau_{client} = \frac{\alpha n\Delta_2^2}{(n-1)\mu} + \frac{3n\Delta_1}{2(n-1)\mu},$$

where $\Delta_2 = \gamma^2 c^2 + n$, and $\Delta_1 = \min(\Delta_2^2, \sqrt{d}\Delta_2)$.

The corresponding (ϵ, δ)-DP guarantees can be obtained using the conversion rule [51]. Finally, we present the utility result for the obtained principal components and compare it with the strong central DP baseline [66].

Lemma 7 (Utility analysis)**.** *Let $\mathbf{V}$ be the rank-k subspace of the original matrix $\mathbf{X}$ and let $\tilde{\mathbf{V}}$ be the principal rank-k subspace obtained from Algorithm 3 with scaling parameter $\gamma \gg n$. Then Algorithm 3 satisfies (ϵ, δ) server-observed DP, and with high probability, we have*

$$\|\mathbf{X}\tilde{\mathbf{V}}\|_F^2 \geq \|\mathbf{X}\mathbf{V}\|_F^2 - O(k\sqrt{n}\sqrt{\mu_{\epsilon,\delta}}), \quad (10)$$

where $\sqrt{\mu_{\epsilon,\delta}} = c_0\sqrt{\log(1/\delta)}/\epsilon$ for some constant c_0.

The optimal error rate in the centralized setting [66] is also $O(k\sqrt{n}\sqrt{\log(1/\delta)}/\epsilon)$. For (α, τ)-RDP, we substitute $\sqrt{\mu_{\epsilon,\delta}}$ with $\sqrt{\mu} \approx \sqrt{\alpha/\tau} \cdot \Delta_2/2$ (Lemma 6) and the error is still in $O(k\sqrt{n})$. Both results indicate the optimality of our solution.

B. Logistic Regression

Problem definition. Consider an input database (matrix) $\mathbf{X}$, where each record (row) consists of a feature vector $\mathbf{x} \in \mathbb{R}^{n-1}$ and a label indicator $y \in \{0, 1\}$. We assume that $\|\mathbf{x}\|_2 \leq 1$, as is done in previous work [67]. We denote $d = n - 1$ for convenience. The server is interested in finding a weight vector $\mathbf{w} \in \mathbb{R}^d$ that minimizes the cross-entropy loss over all records in $\mathbf{X}$, written as $\arg\min_{\mathbf{w}} \frac{1}{|\mathbf{X}|} \sum_{(\mathbf{x},y)\in\mathbf{X}} L(\sigma(\langle \mathbf{w}, \mathbf{x}\rangle), y)$, where $L(\sigma(\langle \mathbf{w}, \mathbf{x}\rangle), y) = -y\log(\sigma(\langle \mathbf{w}, \mathbf{x}\rangle)) - (1-y)\log(1 - \sigma(\langle \mathbf{w}, \mathbf{x}\rangle))$. Here $\langle \mathbf{u}, \mathbf{v}\rangle$ represents the inner product between $\mathbf{u}$ and $\mathbf{v}$, and $\sigma(u) = \frac{1}{1+\exp(-u)}$ is the sigmoid function. To find the optimal $\mathbf{w}$, we perform the gradient descent algorithm [68] that is widely used in FL. The idea is to repeatedly update $\mathbf{w}$ towards the opposite direction of the loss function's gradient $\sum_{(\mathbf{x},y)\in\mathbf{X}} g((\mathbf{x}, y))$, where

$$g((\mathbf{x}, y)) = \sigma(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x} - y \cdot \mathbf{x}. \quad (11)$$

Reduction to polynomial evaluation. Due to the sigmoid function, the computation of Eq. (11) cannot be written as a polynomial of $(\mathbf{x}, y)$. Here we follow [67] to approximate the sigmoid function as polynomials using Taylor series: $\sigma(u) = \sum_{h=0}^{\infty}\left(\frac{\sigma^{(h)}(u)|_{u=0}}{h!} \cdot u^h\right)$, where $\sigma^{(h)}(u)|_{u=0}$ represents the h-th order derivative of $\sigma(u)$ evaluated at $u = 0$. Here, we consider $H = 1$ (we will see soon that it is enough to obtain high utility). Then we can write $\sigma(u) \approx \frac{1}{2} + \frac{1}{4} \cdot u$, and approximate the gradient with $g((\mathbf{x}, y)) \approx \frac{1}{2} \cdot \mathbf{x} + \frac{1}{4} \cdot (\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x} - y \cdot \mathbf{x}$, which is a polynomial of the input $(\mathbf{x}, y)$.

VFL Algorithm. Now we are ready to instantiate SQM on the task of logistic regression. Given weight parameter $\mathbf{w}$, we specify the function of interest for record $(\mathbf{x}, y)$ as

$$f(\mathbf{w}, (\mathbf{x}, y)) = \frac{1}{2} \cdot \mathbf{x} + \langle \frac{\mathbf{w}}{4}, \mathbf{x}\rangle \cdot \mathbf{x} - y \cdot \mathbf{x}. \quad (12)$$

First, the server randomly initializes the model weight $\mathbf{w}$, and clips $\|\mathbf{w}\|_2$ to 1. In each subsequent iteration, the clients randomly sample a batch of records, denoted as $\mathbf{B}$, using shared randomness. The membership of $\mathbf{B}$ is not revealed to the server. Next, the clients call SQM to evaluate the sum of the function $f(\mathbf{w}, \cdot)$ on records from $\mathbf{B}$ using the current weight parameter $\mathbf{w}$ by running Algorithm 3 with scaling parameter γ and noise parameter μ.

The main difference from PCA is in the processing of the coefficients for the polynomial, $\frac{1}{2}, \frac{1}{4}\mathbf{w}$, and 1. The server repeatedly runs Algorithm 2 in each iteration to process these coefficients and then share the results with the clients. Next, the clients evaluate the target polynomial on the processed data and coefficients using BGW, and share the outcome (i.e., perturbed gradient sum) with the server, which then updates $\mathbf{w}$ accordingly, followed by clipping.

Lemma 8 (Privacy analysis)**.** *Given scaling parameter γ, noise parameter μ, running Algorithm 3 over a subset of the input*

data with sampling ratio $q < 1$ for R rounds satisfies (α, τ_{server}) server-observed RDP (α, τ_{client}) client-observed RDP with

$$\tau_{server} = \frac{R}{\alpha - 1} \cdot \log \left((1-q)^{\alpha-1}(\alpha q - q + 1) + \sum_{l=2}^{\alpha} \binom{\alpha}{l} (1-q)^{\alpha-l} q^l e^{(l-1)\tau_l} \right),$$

$$\tau_{client} = R \left(\frac{\alpha n \Delta_2^2}{(n-1)\mu} + \frac{3n\Delta_1}{2(n-1)\mu} \right),$$

where $\Delta_2 = \sqrt{\left(\frac{3}{4}\gamma^3\right)^2 + 9\gamma^5 d + 36\gamma^4}$ *and* $\Delta_1 = \min(\Delta_2^2, \sqrt{d}\Delta_2)$, *and* $\tau_l = \frac{l\Delta_2^2}{4\mu} + \frac{3\Delta_1}{4\mu}$ *for* $l = 2, \ldots, \alpha$.

In the computation for sensitivity Δ_2, $\frac{3}{4}$ corresponds to the original upper bound for the $\mathcal{L}_2$ norm of the 2-degree polynomial $f(\mathbf{w}, (\mathbf{x}, y))$ evaluated on the original input (recall Eq. (12)). With the scaling parameter γ, this sensitivity becomes $\frac{3}{4}\gamma^3 + o(\gamma^3)$, where the small-oh overhead corresponds to the term $\sqrt{9\gamma^5 d + 36\gamma^4}$ in Δ_2. Again, as we increase γ, the relative sensitivity overhead $\sqrt{36\gamma^4 + 9\gamma^5 d}/\gamma^3$ approaches 0. In other words, with a large enough γ, we can make the sensitivity overhead arbitrarily small.

Plugging in the above sensitivities to Lemma 2 and then applying the results on privacy amplification by subsampling and composition theorems [56], [69], [70] give us the result of τ_{server}. Here the subsampling is performed on the record level. A more advanced analysis on the user level privacy (see [22] for a reference) is a promising future work direction. We also note that τ_{client} does not benefit from record-level subsampling, since each client already knows which record is placed in the sampled batch. Further enhancing the protection against clients is a promising future work direction.

C. Discussions

On discretization. In our SQM and its instantiations, we have used an explicit discretization procedure (Algorithm 2) for pre-processing the continuous data instead of using a black box (such as the 64-bit representation of double-precision floating numbers). The reason is that we want to accurately account for the sensitivity that is crucial for DP analysis, since otherwise, it may cause sensitivity underestimation and privacy violation [71], as we have mentioned in Section I. We emphasize that unlike the gradient sum estimation in horizontal FL [42], [44], [45] where each client can independently clip their local gradient to enforce a sensitivity upper bound, in the setting of vertical FL, we cannot trust a party to perform such clipping on an individual gradient, which contains private information regarding all clients' data. Hence, we have enforced explicit local pre-processing for the data to obtain tractable sensitivity analysis.

Computational overhead. In SQM, the quantization procedures for the input data and coefficients and the generation of Skellam noises can be done in parallel efficiently. These two procedures, together with the operation for combining the local Skellam noises injection (performing n additions in BGW), constitute the computational overhead for enforcing DP on top of performing MPC, which protects the clients' data from being observed during FL. Considering that the cost for n additions is much smaller than the cost for computing the covariance matrix of $m \times m$ for PCA or approximating the gradients for a batch of samples in logistic regression, the overhead due to DP is small, compared with MPC itself.

Enhancing DP against clients. Recent work [52] proposes an MPC algorithm for untrusted clients to collectively sample a discrete Gaussian noise [51], a substitute of Skellam for performing integer-valued DP noise injection. Combining it with our SQM could further enhance the client-observed level of privacy, as each client will have no information about the noise outcome (rather than knowing a $\frac{1}{n}$ part of the aggregate noise). There are several caveats. First, generating DP noises using MPC could be prone to side-channel attacks, as the outcome of the random variable is strongly correlated to the overall runtime of the MPC protocol [49]. Another issue is the computational overhead. In our solutions, the clients can generate local noises efficiently in an offline manner while avoiding side-channel attacks. Furthermore, the current implementation [52] focuses on the traditional (ϵ, δ)-DP and also incurs a non-negligible overhead on the privacy parameter δ, which could lead to privacy overheads in compositional mechanisms where the private data is repeatedly queried.

Outsourcing the computation. Besides participating in MPC, which requires synchronicity and incurs high computation and communication costs, the clients can also opt to outsource the computation [72] to non-colluding servers (e.g., 2 servers), and let the servers perform the computation (e.g., using the two-party MPC protocol ABY [73]). This approach is adopted in the federated analytics (FA) framework [74], [75]. Note that their original framework only considers linear functions, and our solution for polynomial evaluation can be seen as a generalization.

Data partitioning. The privacy guarantees in Lemmas 6 and 8) depend on the quantization level, the scale of the local noises, and the number of clients that participated in the protocol, and is independent of how clients partition the data. In our experiments, we assume that each client partitions one column of the overall sensitive dataset, without loss of generality.

VI. Experiments

In this section, we validate the performance for our SQM on the task of principal component analysis and logistic regression. Following previous works in distributed DP algorithms for federated learning [42], [44], [45], we focus on measuring the performance for the fitted models in terms of privacy-utility trade-offs. We use a single machine to *simulate* the distributed environment where each party is assumed to have a secure and noiseless channel for communication. **Baselines.** We consider the local DP solution (outlined in Section III-B) as the VFL baseline, which perturbs the dataset $\mathbf{X}$ on the client side and then trains the PCA/LR model on the perturbed dataset. We are aware of other VFL solutions that try to enforce DP (e.g., [3], [63], [64]). However, their threat models and privacy

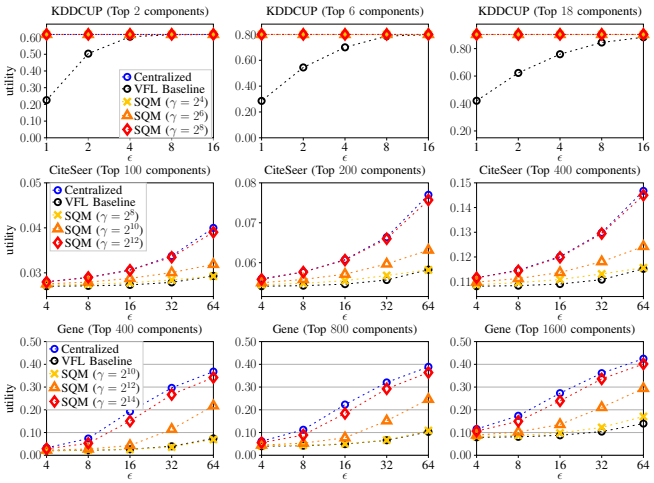

Fig. 3: Performance for PCA on multiple datasets for varying numbers of top components and DP constraints.

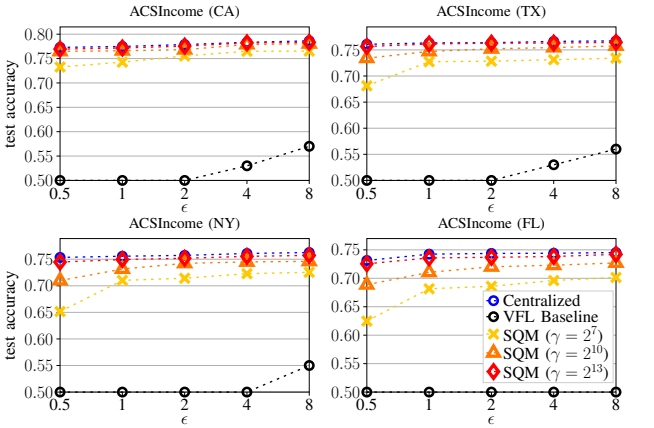

Fig. 4: Performance for logistic regression on multiple datasets under different DP constraints.

requirements largely differ from ours and hence, they are not directly comparable with ours. Overall, we want to demonstrate that SQM achieves comparable utility-privacy trade-off as the central-DP baseline, as we increase the quantization granularity. Other local DP solutions (e.g., [5], [76]), no matter how advanced, exhibit notable performance gaps between the central-DP baseline and are omitted.

We also use a central DP mechanism to establish the performance upper limit for VFL (the goal for our mechanism to achieve). For PCA, we use the classic [66]; and for LR, we use the popular centralized-DP mechanism, DPSGD [55]. We omit other central-DP mechanisms since they perform similarly. For SQM, we also vary the scaling parameter γ (larger γ means finer quantization granularity).

Principal Component Analysis. For PCA, we evaluate the KDDCUP dataset [77] with $m = 195666$ and $n = 117$, and high dimensional datasets, CiteSeer [78] with $m = 2110$ and $n = 3703$ and Gene [79] with $m = 801$ and $n = 20531$. All datasets are partitioned to n clients. We report the utility

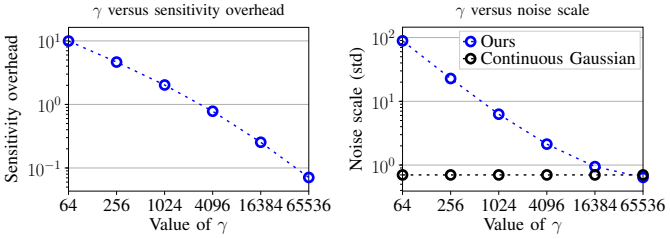

Fig. 5: Effect of parameter γ on the sensitivity and noise overheads of SQM on LR compared with central-DP baseline.

of the obtained subspace $\tilde{\mathbf{V}}$ computed on the input dataset $\mathbf{X}$, defined as $\|\mathbf{X}\tilde{\mathbf{V}}\|_F^2$, for different numbers of top principal components. In terms of privacy, we focus on the server-observed DP under the standard (ϵ, δ)-DP framework, since it is also commonly considered in the centralized and horizontal FL settings. We fix δ to 10^{-5} and vary ϵ. For our mechanism, we also vary the scaling parameter γ. For the high-dimensional CiteSeer and Gene, we choose a relatively larger γ to maintain the signal-to-noise ratio.

The average utility over 20 independent runs is in Figure 3. It is clear that our solution consistently outperforms the local DP baseline. The performance of SQM is close to that of centralized DP under various parameter settings, especially when γ is large. This confirms the optimality of SQM. In particular, the performance of SQM improves with γ. This is because as γ increases, the relative sensitivity overhead becomes smaller, hence achieving higher utility while maintaining the same level of privacy. In particular, for KDDCUP, SQM performs almost the same as the centralized-DP solution for a wide range of γ values. Similar conclusions can be drawn for the other two high-dimensional datasets, Gene and CiteSeer. The performance gap between SQM and the centralized solution is negligible when γ is large enough (when γ reaches 2^{14} and 2^{12}, respectively).

Logistic Regression. For LR, we evaluate the ACSIncome datasets that is collected from the US Census in the year 2018 regarding four states of the US: California, Texas, New York, and Florida [80]. The task is to predict whether a person has an annual income over $50K$. Each dataset contains about $n = 800$ dimensions (including features and the label) and 100,000 records. We randomly sample 10% of the datasets as the training data, resulting in $m \approx 10,000$. All datasets are vertically partitioned onto n clients.

For all experiments, we fix the privacy parameter $\delta = 10^{-5}$ and vary ϵ. We do not tune the hyperparameters in favor of any algorithm. We fix the subsampling rate for records to 0.001 for all experiments. For $\epsilon = 0.5, 1, 2, 4, 8$ we run our SQM and DPSGD over the subsampled batches for $2, 5, 8, 10, 10$ epochs, respectively. For the VFL baseline where the model is fitted on perturbed DP data, we train the model until convergence. Similar to PCA, the privacy level is calculated based on server-observed privacy. The average test accuracy over 20 independent runs is in Figure 4.

In Figure 4, SQM significantly outperforms the VFL baseline under all parameter settings for all datasets. In addition, even

when ϵ is small, our solution can still maintain a decent utility. In addition, given a sufficient amount of privacy budget, SQM achieves comparable performance as the centralized-DP approach. Specifically, the accuracy drop of our solution with $\gamma = 2^{13}$ compared with the centralized solution is negligible for $\gamma \geq 1$. When $\gamma = 2^{10}$, our solution still maintains decent utilities across different privacy parameter settings.

Effect of scaling parameter. We study how γ influences the sensitivity overhead and hence the scale of DP noise for the analysis in Lemma 8. We compute the $\mathcal{L}_2$ sensitivity overhead for different choice of γ ranging from $\{64, 256, 1024, 4096, 16394, 65536\}$, which is $\sqrt{(\frac{3}{4})^2 + \frac{9d}{\gamma} + \frac{36}{\gamma^2}} - \frac{3}{4}$, with $d = 800$ (recall Lemma 8 and that $\frac{3}{4}$ is the original sensitivity).

Next, we fix privacy parameters $\epsilon = 1$ and $\delta = 10^{-5}$, and vary γ to find the minimum scale of noise such that the target privacy level is satisfied (for running SQM with a subsampling rate of 0.001 and epoch number of 5). Similarly, we show the normalized scale of the Skellam noise in SQM (namely, the standard deviation of the noise injected into the processed gradient sum on the server side). As a reference, we have included the scale of the Gaussian noise of centralized DPSGD.

From Figure 5, it is clear that as we increase the scaling paraemter γ, both the sensitivity overhead and noise overhead compared with the centralized Gaussian quickly decreases to 0 (note that the y-axis is in log scale). This result explains why the performance of our SQM on LR approaches the centralized competitor in Figure 4, as we increase γ. Similar explanations also apply to PCA, and are omitted due to space constraints.

VII. Related Work

Federated learning over vertically partitioned databases has been extensively studied in the database community, e.g., see [1]–[10], [12], [81]. Various algorithms, frameworks, and systems have been proposed for privacy protection in VFL. For example, [3], [6], [7] train tree-based models, where models are trained using clients' partitioned data without explicitly sharing it; Fu et al. [8] propose a communication-efficient framework for selecting features; Li et al. [5] study K-means clustering with local DP constraints; [7], [9], [10] propose and implement efficient and practical VFL systems. We refer readers to [14] for a comprehensive survey.

Recent works that try to enforce DP for VFL adopt different privacy requirements and threat models than ours. Specifically, in works such as [63], [64], the clients first collectively compute the function of interest over their partitioned data using MPC protocols. After that, a client (or some third party) injects DP noises into the outcome produced by MPC. This approach may lead to privacy violations since the party who performs the noise injection could infer information about the clients' inputs from the exact result before the noise injection. Wu et al. [3] let the clients use shared randomness to jointly sample a Laplace noise using MPC for DP protection. This mechanism is also non-private in the presence of a curious client, who has observed the noise outcome and the exact result.

Preserving data privacy for analyzing databases jointly owned by several clients predates DP and FL [82]–[84]. Due to the differences in privacy definitions and problem formalizations, our work is not directly comparable with theirs, and we refer interested readers to the original papers for more details.

Amplifcation of DP in FL. The idea of utilizing cryptographic protocols to enhance DP originates from horizontal FL [40], [41], which, in turn, drives the development of integer-valued DP mechanisms (e.g., see [42], [45], [51]) and inspires other applications such as [11], [85]–[87]. The most relevant works with ours are [42], [45], the function of interest is the sum of private data items possessed by multiple clients. We note that their approaches do not apply to our problem of evaluating polynomial functions over vertically partitioned databases, due to *non-linearity*–we cannot convert our function of interest, a polynomial of inputs, to a linear sum of individual components so that each of which can be computed and perturbed by a single client without accessing other clients' data. Adopting other techniques for distributed DP (e.g., [88], [89]) into our VFL setting is a promising future work direction.

Numerical issues in DP. We highlight the importance of DP mechanisms that allow accurate privacy accounting and easy implementation in practice. Mironov [50] first exploited the discrepancy between theoretical analysis and practical implementation of the continuous Laplace noise, and designed an attack that reconstructs an entire database consisting of $18,000$ records under a restrictive privacy budget smaller than $\epsilon = 10^{-6}$. Integer-valued DP mechanisms are a solution to the numerical issue–recent papers [42], [44], [45], [51], [90], [91] propose discrete DP algorithms that are easily implementable on computers and distributed systems with finite computation and communication capabilities. The OpenDP library (https://opendp.org) also forbids the users from accessing floating-point mechanisms that do not have verified proofs by default.

VIII. Conclusion

In this work, we study the problem of evaluating polynomial functions over vertically partitioned datasets in federated learning, with differential privacy guarantees. We present SQM that provably achieves comparable performance as the centralized approach, without assuming any trusted party. We apply our mechanism to two classic machine learning problems, PCA and logistic regression, and validate their empirical performance.

Regarding future work directions, we plan to further enhance the privacy protection of our mechanism against adversarial clients when performing record-level subsampling, by preventing the clients from knowing exactly which records are sampled. A trivial solution is to let the clients compute the outcome for all possibilities of subsampling, and then randomly choose one of them using shared randomness, which would slow down the FL procedure dramatically. We also plan to extend our mechanism to other application scenarios, such as analyzing distributed social networks, mining association rules, and recommendation algorithms over distributed user data.

REFERENCES

[1] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *ICDE*, 2021, pp. 181–192.

[2] J. Wang, L. Zhang, A. Li, X. You, and H. Cheng, "Efficient participant contribution evaluation for horizontal and vertical federated learning," in *ICDE*, 2022, pp. 911–923.

[3] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *PVLDB*, vol. 13, no. 11, pp. 2090–2103, 2020.

[4] F. Fu, Y. Shao, L. Yu, J. Jiang, H. Xue, Y. Tao, and B. Cui, "Vf2boost: Very fast vertical federated gradient boosting for cross-enterprise learning," in *SIGMOD*, 2021, p. 563–576.

[5] Z. Li, T. Wang, and N. Li, "Differentially private vertical federated clustering," in *PVLDB*, 2023, p. 1277–1290.

[6] F. Fu, H. Xue, Y. Cheng, Y. Tao, and B. Cui, "Blindfl: Vertical federated machine learning without peeking into your data," in *SIGMOD*, 2022, pp. 1316–1330.

[7] X. Li, Y. Hu, W. Liu, H. Feng, L. Peng, Y. Hong, K. Ren, and Z. Qin, "Opboost: a vertical federated tree boosting framework based on order-preserving desensitization," in *PVLDB*, 2022, p. 202–215.

[8] R. Fu, Y. Wu, Q. Xu, and M. Zhang, "Feast: A communication-efficient federated feature selection framework for relational data," in *PACMMOD*, 2023.

[9] Y. Wu, N. Xing, G. Chen, T. T. A. Dinh, Z. Luo, B. C. Ooi, X. Xiao, and M. Zhang, "Falcon: A privacy-preserving and interpretable vertical federated learning system," *PVLDB*, p. 2471–2484, 2023.

[10] F. Fu, X. Miao, J. Jiang, H. Xue, and B. Cui, "Towards communication-efficient vertical federated learning training via cache-enabled local updates," in *PVLDB*, 2022, p. 2111–2120.

[11] Z. Huang, Y. Qiu, K. Yi, and G. Cormode, "Frequency estimation under multiparty differential privacy: one-shot and streaming," *PVLDB*, p. 2058–2070, 2022.

[12] Y. Cheng, L. Zhang, J. Wang, X. Chu, H. Dongbo, and L. Xu, "Fedmix: Boosting with data mixture for vertical federated learning," in *ICDE*, 2024, pp. 3379–3392.

[13] B. 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.

[14] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," in *TKDE*, 2024, p. 1–20.

[15] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015, pp. 1310–1321.

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

[17] S. Maddock, G. Cormode, T. Wang, C. Maple, and S. Jha, "Federated boosted decision trees with differential privacy," in *CCS*, 2022, p. 2249–2263.

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

[19] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *S&P*, 2019, pp. 691–706.

[20] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *PODS*, 2003, p. 202–210.

[21] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.

[22] J. Fang and K. Yi, "Privacy amplification by sampling under user-level differential privacy," *PACMMOD*, vol. 2, no. 1, 2024.

[23] S. Takagi, L. Xiong, F. Kato, Y. Cao, and M. Yoshikawa, "Hrnet: Differentially private hierarchical and multi-resolution network for human mobility data synthesization," *PVLDB*, vol. 17, no. 11, p. 3058–3071, 2024.

[24] Y. Hu, Y. Du, Z. Zhang, Z. Fang, L. Chen, K. Zheng, and Y. Gao, " Real-Time Trajectory Synthesis with Local Differential Privacy," in *ICDE*, 2024, pp. 1685–1698.

[25] X. Ran, Q. Ye, H. Hu, X. Huang, J. Xu, and J. Fu, "Differentially Private Graph Neural Networks for Link Prediction," in *ICDE*, 2024, pp. 1632–1644.

[26] Y. Mao, Q. Ye, H. Hu, Q. Wang, and K. Huang, "PrivShape: Extracting Shapes in Time Series Under User-Level Local Differential Privacy," in *ICDE*, 2024, pp. 1739–1751.

[27] X. Sun, Q. Ye, H. Hu, J. Duan, T. Wo, J. Xu, and R. Yang, "LDPRecover: Recovering Frequencies from Poisoning Attacks Against Local Differential Privacy," in *ICDE*, 2024, pp. 1619–1631.

[28] W. Dong, D. Sun, and K. Yi, "Better than composition: How to answer multiple relational queries under differential privacy," in *PACMMOD*, 2023, p. 155–163.

[29] Y. Jiang, X. Luo, Y. Yang, and X. Xiao, "Calibrating noise for group privacy in subsampled mechanisms," *PVLDB*, p. to appear, 2025.

[30] F. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *SIGMOD*, 2009, pp. 19–30.

[31] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *CCS*, 2014, pp. 1054–1067.

[32] K. Nissim, "Privacy: From database reconstruction to legal theorems," in *PODS*, 2021, p. 33–41.

[33] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *PODS*, 2005, p. 128–138.

[34] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: private data release via bayesian networks," in *SIGMOD*, 2014, p. 1423–1434.

[35] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *SIGMOD*, 2018, p. 131–146.

[36] M. Xu, B. Ding, T. Wang, and J. Zhou, "Collecting and analyzing data jointly from multiple services under local differential privacy," in *PVLDB*, 2020, pp. 2760–2772.

[37] T. Wang, B. Ding, M. Xu, Z. Huang, C. Hong, J. Zhou, N. Li, and S. Jha, "Improving utility and security of the shuffler-based differential privacy," in *PVLDB*, 2020, pp. 3545 – 3558.

[38] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE*, 2019, pp. 638–649.

[39] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld, "Prochlo: Strong privacy for analytics in the crowd," in *SOSP*, 2017, p. 441–459.

[40] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *SODA*, 2019, p. 2468–2479.

[41] A. Cheu, A. D. Smith, J. R. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *EUROCRYPT*, 2019, pp. 375–403.

[42] E. Bao, Y. Zhu, X. Xiao, Y. Yang, B. C. Ooi, B. H. M. Tan, and K. M. M. Aung, "Skellam mixture mechanism: a novel approach to federated learning with differential privacy," *PVLDB*, pp. 2348–2360, 2022.

[43] 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 *CCS*, 2017, p. 1175–1191.

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

[45] N. Agarwal, P. Kairouz, and Z. Liu, "The skellam mechanism for differentially private federated learning," in *NeurIPS*, 2021, pp. 5052–5064.

[46] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *FTTCS*, vol. 9, no. 3-4, pp. 211–407, 2014.

[47] M. Keller, "Mp-spdz: A versatile framework for multi-party computation," in *CCS*, 2020, p. 1575–1590.

[48] D. Rathee, A. Bhattacharya, R. Sharma, D. Gupta, N. Chandran, and A. Rastogi, "Secfloat: Accurate floating-point meets secure 2-party computation," in *S&P*, 2022, pp. 576–595.

[49] J. Jin, E. McMurtry, B. I. P. Rubinstein, and O. Ohrimenko, "Are we there yet? timing and floating-point attacks on differential privacy systems," in *S&P*, 2021, pp. 473–488.

[50] I. Mironov, "On significance of the least significant bits for differential privacy," in *CCS*, 2012, p. 650–661.

[51] C. L. Canonne, G. Kamath, and T. Steinke, "The discrete gaussian for differential privacy," in *NeurIPS*, 2020.

[52] C. Wei, R. Yu, Y. Fan, W. Chen, and T. Wang, "Securely sampling discrete gaussian noise for multi-party differential privacy," in *CCS*, 2023, pp. 2262–2276.

[53] H. Keller, H. Möllering, T. Schneider, O. Tkachenko, and L. Zhao, "Secure noise sampling for DP in MPC with finite precision," in *ARES*, 2024.

[54] "Ieee standard for floating-point arithmetic - redline," *IEEE Std 754-2008 (Revision of IEEE Std 754-1985) - Redline*, pp. 1–82, 2008.
[55] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016, pp. 308–318.
[56] I. Mironov, "Rényi differential privacy," in *CSF*, 2017, pp. 263–275.
[57] A. C. Yao, "How to generate and exchange secrets," in *FOCS*, 1986, pp. 162–167.
[58] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *S&P*, 2017, pp. 19–38.
[59] G. Oded, *Foundations of Cryptography*, USA, 2009.
[60] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *STOC*, 1988, p. 1–10.
[61] A. Ben-Efraim, Y. Lindell, and E. Omri, "Optimizing semi-honest secure multiparty computation for the internet," Cryptology ePrint Archive, 2016.
[62] Q. Pang, J. Zhu, H. Mollering, W. Zheng, and T. Schneider, "Bolt: Privacy-preserving, accurate and efficient inference for transformers," in *S&P*, 2024, pp. 4753–4771.
[63] D. Xu, S. Yuan, and X. Wu, "Achieving differential privacy in vertically partitioned multiparty learning," in *Big Data*, 2021, pp. 5474–5483.
[64] T. Ranbaduge and M. Ding, "Differentially private vertical federated learning," *CoRR*, vol. abs/2211.06782, 2022.
[65] E. Bao, F. Wei, Y. Yang, X. Xiao, T. Pang, and C. Du. (2024) Towards learning on vertically partitioned data with distributed differential privacy (technical report). [Online]. Available: https://github.com/erguteb/sqm_icde2025/blob/main/technical_report.pdf
[66] C. Dwork, K. Talwar, A. Thakurta, and L. Zhang, "Analyze gauss: optimal bounds for privacy-preserving principal component analysis," in *STOC*, 2014, pp. 11–20.
[67] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *PVLDB*, p. 1364–1375, 2012.
[68] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462 – 466, 1952.
[69] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled gaussian mechanism," *CoRR*, vol. abs/1908.10530, 2019.
[70] Y. Zhu and Y.-X. Wang, "Poission subsampled Rényi differential privacy," in *ICML*, 2019, pp. 7634–7642.
[71] S. Casacuberta, M. Shoemate, S. Vadhan, and C. Wagaman, "Widespread underestimation of sensitivity in differentially private libraries and how to fix it," in *CCS*, 2022, p. 471–484.
[72] Z. Shan, K. Ren, M. Blanton, and C. Wang, "Practical secure computation outsourcing: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, 2018.
[73] D. Demmler, T. Schneider, and M. Zohner, "Aby - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.
[74] K. Talwar, S. Wang, A. McMillan, V. Jina, V. Feldman, B. Basile, A. Cahill, Y. S. Chan, M. Chatzidakis, J. Chen *et al.*, "Samplable anonymous aggregation for private federated data analysis," *arXiv preprint arXiv:2307.15017*, 2023.
[75] Google Research, "Federated analytics: Collaborative data science without data collection," 2020. [Online]. Available: https://research.google/pubs/pub48576/
[76] D. Wang and J. Xu, "Principal component analysis in the local differential privacy model," *TCS*, vol. 809, pp. 296–312, 2020.
[77] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *CISDA*, 2009, pp. 1–6.
[78] C. L. Giles, K. D. Bollacker, and S. Lawrence, "CiteSeer: An automatic citation indexing system," in *JCDL*, New York, NY, USA, 1998, pp. 89–98.
[79] S. Fiorini, "Gene expression cancer RNA-Seq," UCI Machine Learning Repository, 2016.
[80] F. Ding, M. Hardt, J. Miller, and L. Schmidt, "Retiring adult: New datasets for fair machine learning," in *NeurIPS*, 2021, pp. 6478–6490.
[81] W. Dai, X. Jiang, L. Bonomi, Y. Li, H. Xiong, and L. Ohno-Machado, "VERTICOX: vertically distributed cox proportional hazards model using the alternating direction method of multipliers," *TKDE*, vol. 34, no. 2, pp. 996–1010, 2022.
[82] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD*, 2002, p. 639–644.
[83] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *CRYPTO*, 2004, pp. 528–544.
[84] A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining," in *PODS*, 2003, p. 211–222.
[85] A. R. Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha, "Crypt?: Crypto-assisted differential privacy on untrusted servers," in *SIGMOD*, 2020, pp. 603–619.
[86] S. Wagh, X. He, A. Machanavajjhala, and P. Mittal, "Dp-cryptography: marrying differential privacy and cryptography in emerging applications," *Commun. ACM*, vol. 64, no. 2, pp. 84–93, 2021.
[87] S. Zhang, X. He, A. Kundu, S. Mehrotra, and S. Sharma, "Secure normal form: Mediation among cross cryptographic leakages in encrypted databases," in *ICDE*, pp. 5560–5573.
[88] S. Liu, Y. Cao, T. Murakami, J. Liu, and M. Yoshikawa, " CARGO: Crypto-Assisted Differentially Private Triangle Counting Without Trusted Servers ," in *ICDE*, 2024.
[89] S. P. Liew, T. Takahashi, S. Takagi, F. Kato, Y. Cao, and M. Yoshikawa, "Network shuffling: Privacy amplification via random walks," in *SIGMOD*, 2022, p. 773–787.
[90] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. B. McMahan, "Cpsgd: Communication-efficient and differentially-private distributed sgd," in *NeurIPS*, 2018, p. 7575–7586.
[91] C. Ilvento, "Implementing the exponential mechanism with base-2 differential privacy," in *CCS*, 2020, p. 717–742.
[92] B. Balle and Y.-X. Wang, "Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising," in *ICML*, 2018.
[93] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[94] R. Vershynin, "Spectral norm of products of random and deterministic matrices," *Probability Theory and Related Fields*, vol. 150, pp. 471–509, 2011.
[95] G. Dai, Z. Su, and H. Wang, "Tail bounds on the spectral norm of sub-exponential random matrices," in *Random Matrices: Theory and Applications*, vol. 13, no. 01, 2023, p. 2350013.
[96] F. Valovich and F. Aldà, "Computational differential privacy from lattice-based cryptography," in *NuTMiC*, vol. 10737, 2017, pp. 121–141.

APPENDIX

A. Differential Privacy

Injecting Gaussian noise to a function F satisfies (ϵ, δ)-DP.

Lemma 9 (Analytic Gaussian Mechanism [92])**.** *Injecting Gaussian noise $\mathcal{N}\left(\mathbf{0}, \sigma^2 \cdot \mathbf{I}\right)$ into the output of F satisfies (ϵ, δ)-differential privacy, if*

$$\frac{S(F)}{\sigma} \leq \sqrt{2}\left(\sqrt{\chi^2+\epsilon}-\chi\right),$$

where $\mathbf{0}$ and $\mathbf{I}$ are a zero vector and a $d \times d$ identity matrix, respectively, and χ is the solution to

$$\operatorname{erfc}(\chi)-\exp(\epsilon) \cdot \operatorname{erfc}\left(\sqrt{\chi^2+\epsilon}\right)=2\delta,$$

and $\operatorname{erfc}()$ *denotes the complementary error function. Namely,*

$$\operatorname{erfc}(x)=1-\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t.$$

RDP can be converted to the classic (ϵ, δ) DP using the following lemma.

Lemma 10 (Converting (α, τ)-RDP to (ϵ, δ)-DP [51])**.** *Given a mechanism $\mathcal{M}$ satisfies (α, τ)-RDP for any $\alpha \in (1, \infty)$, it satisfies (ϵ, δ)-DP for $\delta > 0$ and*

$$\epsilon=\tau+\frac{\log(1/\delta)+(\alpha-1)\log(1-1/\alpha)-\log(\alpha)}{\alpha-1}.$$

The composition of multiple RDP mechanisms also satisfies RDP.

Algorithm 4: Baseline Solution for VFL with DP

Input: Dataset $\mathbf{X} = (\mathbf{X}[:,1], \ldots, \mathbf{X}[:,n])$ partitioned among n clients; noise parameter σ.

for *each client* $j \in [n]$ **do**
  $\tilde{\mathbf{X}}[:,j] \leftarrow \mathbf{X}[:,j] + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_m)$.
  Client j sends the perturbed $\tilde{\mathbf{X}}[:,j]$ to the server.

The server reconstructs $\tilde{\mathbf{X}} = (\tilde{\mathbf{X}}[:,1], \ldots, \tilde{\mathbf{X}}[:,n])$.

Lemma 11 (Composition Lemma for RDP [56]). *If mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_T$ satisfies $(\alpha, \tau_1), \ldots, (\alpha, \tau_T)$-RDP, respectively, then $\mathcal{M}_1 \circ \ldots \circ \mathcal{M}_T$ satisfies $(\alpha, \sum_{t=1}^{T} \tau_t)$-RDP.*

If a mechanism is Rényi-DP, then the same mechanism that runs on a random subset of the input dataset also satisfies Rényi-DP.

Lemma 12 (Subsampling for RDP [69]). *Let $\mathcal{M}$ be a mechanism that satisfies (l, τ_l)-RDP for $l = 2, \ldots, \alpha$ ($\alpha \in \mathbb{Z}, \alpha > 2$), and S_q be a procedure that uniformly samples each record of the input data with probability q. Then $\mathcal{M} \circ S_q$ satisfies (α, τ)-RDP with*

$$\tau = \frac{1}{\alpha - 1} \cdot \log \left((1-q)^{\alpha-1}(\alpha q - q + 1) + \sum_{l=2}^{\alpha} \binom{\alpha}{l} (1-q)^{\alpha-l} q^l e^{(l-1)\tau_l} \right).$$

B. The BGW Protocol

For completeness, we briefly review the idea of BGW [60] in the following. A building block of BGW is Shamir's secret sharing algorithm [93], which enables a secret holder to distribute a secret among a group of parties in a way such that no single party can learn any non-trivial information about the secret, and when a sufficiently large number of parties combine their information, the secret is reconstructed. Built upon secret sharing [93], BGW [60] implements a three-phase execution.

1) First, each party distributes her private input as secret shares to other clients using Shamir's algorithm [93]. In our setting, each party (namely, a client) distributes her private data partition to other clients as secret shares. Note that no party can infer any other party's private data partition.
2) Next, each party simulates the computation of the function using a digital circuit while keeping the value of each computed gate (of the circuit) as a secret shared by all parties. Similar to the first step, the value of each computed gate can not be inferred by any party.
3) Finally, all of the parties reconstruct the true outcome of the function, using their secret shares.

BGW achieves a strong notion of information-theoretic security. Since all intermediate outcomes in Steps (1) and (2) are observed by the parties as secret shares, they could infer anything non-trivial about the private inputs. Only after step (3), will they obtain non-trivial information about the private inputs–the reconstructed outcome. Besides the outcome itself, no information regarding the private inputs can be learned from any party throughout this process (regardless of the computation power). We refer interested readers to the original paper for more details.

The baseline solution of VFL with DP is outlined as in Algorithm 4, where the clients perturb the private dataset $\mathbf{X}$ directly and share the perturbed dataset with the server. As we have mentioned in Section III-B, this method applies to arbitrary tasks but incurs high errors.

We present the privacy guarantees for Algorithm 4.

Lemma 13. *For any noise parameter σ, and integer $\alpha > 1$, Algorithm 4 satisfies (α, τ_{server}) server-observed RDP and (α, τ_{client}) server-observed RDP with $\tau_{server} = \frac{\alpha c^2}{2\mu}$, and $\tau_{client} = \frac{\alpha 2c^2}{\mu}$.*

The proof follows from Lemma 1. Note that τ_{server} and τ_{client} defer by a factor of 2^2. This is because, in client-observed DP, the sensitivity for releasing the input data is two times that in server-observed DP. The corresponding (ϵ, δ)-DP guarantees can be obtained using Lemma 9.

Proof of Lemma 3. We first prove the case when the input record x one dimensional with $|x| \leq c$. We denote its outcome of Algorithm 2 (with scaling parameter γ) as $\hat{x}$, and show that $(\hat{x})^\lambda = \gamma^\lambda x^\lambda + O(\gamma^{\lambda-1})$. The case for multi-dimensional inputs is omitted as it easily follows from mathematical induction on the number of dimensions–when both u and v are bounded, we have that the multiplication of $\gamma^{\lambda_1} u^{\lambda_1} + O(\gamma^{\lambda_1 - 1})$ and $\gamma^{\lambda_2} v^{\lambda_2} + O(\gamma^{\lambda_2 - 1})$ is $\gamma^{\lambda_1+\lambda_2} u^{\lambda_1} v^{\lambda_2} + O(\gamma^{\lambda_1+\lambda_2-1})$.

Without loss of generality, we consider $x \geq 0$. We first compute the upper bound for $(\hat{x})^\lambda$. We have

$$(\hat{x})^\lambda \leq (\gamma x + 1)^\lambda = \gamma^\lambda x^\lambda + \binom{\lambda}{1} \gamma^{\lambda-1} x^{\lambda-1} + \ldots + \binom{\lambda}{\lambda} \gamma^0 x^0.$$

When $0 \leq x < 1$, we have

$$\begin{aligned}(\hat{x})^\lambda &\leq \gamma^\lambda x^\lambda + \binom{\lambda}{1} \gamma^{\lambda-1} + \binom{\lambda}{2} \gamma^{\lambda-2} + \ldots + \binom{\lambda}{\lambda} \gamma^0 \\ &\leq \gamma^\lambda x^\lambda + \gamma^\lambda \left(\frac{\lambda}{\gamma} + \frac{\lambda^2}{2!\gamma^2} + \ldots + \frac{\lambda^\lambda}{\lambda! \gamma^\lambda} \right) \\ &\leq \gamma^\lambda x^\lambda + \gamma^\lambda \left(\exp\left(\frac{\lambda}{\gamma}\right) - 1 \right).\end{aligned}$$

Since $\exp(v) \leq 1 + 2v$ when $v \leq 0.01$ (recall that $\gamma \geq 100\lambda$), we have

$$(\hat{x})^\lambda \leq \gamma^\lambda x^\lambda + 2\lambda \gamma^{\lambda-1}. \tag{13}$$

When $x \geq 1$, we can also derive

$$(\hat{x})^\lambda \leq \gamma^\lambda x^\lambda + 2\lambda x^{\lambda-1} \gamma^{\lambda-1}. \tag{14}$$

Similarly, we have the following lower bounds for $(\hat{x})^\lambda$

$$(\hat{x})^\lambda \geq \gamma^\lambda x^\lambda - 2\lambda \gamma^{\lambda-1}, \tag{15}$$

when $0 \leq x < 1$, and

$$(\hat{x})^\lambda \geq \gamma^\lambda x^\lambda - 2\lambda x^{\lambda-1} \gamma^{\lambda-1}. \tag{16}$$

Here we note that the term $\lambda x^{\lambda-1}$ in Eq. (14) and (16) is bounded by the constant $\lambda c^{\lambda-1}$, which becomes negligible with respect to $\gamma^{\lambda-1}$ when γ is large. Combining Eq. (13), (14), (15), (16), we conclude that

$$(\hat{x})^\lambda = \gamma^\lambda x^\lambda + O(\gamma^{\lambda-1}). \tag{17}$$

□

Before we present the proof of Lemma 7, we first prove the following.

Lemma 14. *Let $\mathbf{V}_k$ be the rank-k subspace of the original matrix $\mathbf{X}$ and let $\tilde{\mathbf{V}}_k$ be the principal rank-k subspace of matrix $\tilde{\mathbf{C}}$ obtained from Algorithm 3 with scaling parameter $\gamma \gg n$ and noise parameter μ. Then with high probability, we have that*

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - O(k\sqrt{n}\sqrt{\mu}), \tag{18}$$

Proof of Lemma 14. Recall that to obtain $\hat{\mathbf{X}}$, we first obtain $\gamma\mathbf{X}$ (Line 1 in Algorithm 2) and then randomly round the entries in γD to the nearest integers (Lines 2-6 in Algorithm 2). Hence, we can decompose $\hat{\mathbf{X}}$ as $\gamma\mathbf{X} + \mathbf{E}$, where $\mathbf{E}$ is the random matrix due to stochastic rounding. Every entry of $\mathbf{E}$ is independent and is of mean 0 and is from the region $[-1, 1]$. Hence, we can rewrite $\tilde{\mathbf{C}}$ as follows.

$$\tilde{\mathbf{C}} = \gamma^2\mathbf{X}^T\mathbf{X} + \underbrace{2\gamma\mathbf{E}^T\mathbf{X} + \mathbf{E}^T\mathbf{E} + \mathbf{N}}_{\text{denoted as } \mathbf{H}}, \tag{19}$$

where $\mathbf{N}$ is the symmetric random matrix, where each entry on the upper diagonal is independently sampled from $\mathrm{Sk}(\mu)$. We define $\mathbf{H} := 2\gamma\mathbf{E}^T\mathbf{X} + \mathbf{E}^T\mathbf{E} + \mathbf{N}$, the random matrix due to scaling, rounding, and noise injection.

Since $\tilde{\mathbf{V}}_k$ is the principal subspace of matrix $\tilde{\mathbf{C}}$, we have

$$\begin{aligned}
&Tr(\tilde{\mathbf{V}}_k^T(\gamma^2\mathbf{X}^T\mathbf{X} + \mathbf{H})\tilde{\mathbf{V}}_k) \\
&\geq Tr(\mathbf{V}_k^T(\gamma^2\mathbf{X}^T\mathbf{X} + \mathbf{H})\mathbf{V}_k) \\
&\geq Tr(\gamma^2\mathbf{V}_k^T\mathbf{X}^T\mathbf{X}\mathbf{V}_k) + Tr(\mathbf{V}_k\mathbf{H}\mathbf{V}_k) \\
&\geq Tr(\gamma^2\mathbf{V}_k^T\mathbf{X}^T\mathbf{X}\mathbf{V}_k) - k\|\mathbf{H}\|_2.
\end{aligned}$$

Hence,

$$Tr(\tilde{\mathbf{V}}_k^T(\gamma^2\mathbf{X}^T\mathbf{X})\tilde{\mathbf{V}}_k) \geq Tr(\gamma^2\mathbf{V}_k^T\mathbf{X}^T\mathbf{X}\mathbf{V}_k) - 2k\|\mathbf{H}\|_2,$$

which is equivalent to

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - 2k\|\frac{1}{\gamma^2}\cdot\mathbf{H}\|_2. \tag{20}$$

It suffices to bound the spectral norm of $\frac{1}{\gamma^2}\cdot\mathbf{H}$, which is the sum of $\frac{2}{\gamma}\cdot\mathbf{E}^T\mathbf{X}$, $\frac{1}{\gamma^2}\cdot\mathbf{E}^T\mathbf{E}$, and $\frac{1}{\gamma^2}\cdot\mathbf{N}$.

First, note that each entry of $\mathbf{E}$ is independent and has zero mean and $\mathbf{X}$ is a deterministic matrix. [94] showed that the spectral norm of matrix $\mathbf{E}^T\mathbf{X}$ has mean $O(\sqrt{n})$ and variance smaller than $3n$. Applying Chebyshev's inequality, we can see that the spectral norm of matrix $\mathbf{E}^T\mathbf{X}$ is $O(n)$ with high probability. When $\gamma \gg n$, this $O(n)$ becomes negligible after scaled by $\frac{2}{\gamma}$

Next, for $\mathbf{E}^T\mathbf{E}$, we have that $\|\mathbf{E}^T\mathbf{E}\|_2 \leq \|\mathbf{E}^T\|_2\|\mathbf{E}\|_2$, where both $\mathbf{E} = \mathbf{E}\mathbf{I}$ and $\mathbf{E}^T = \mathbf{E}^T\mathbf{I}$ have spectral norms of expectation $\sqrt{n}$. Using the same argument, we have that the spectral norm of matrix $\mathbf{E}^T\mathbf{E}$ is also of $O(n^2)$ with high probability. When $\gamma \gg n$, this $O(n)$ becomes negligible after scaled by $\frac{1}{\gamma^2}$

To bound the spectral norm of matrix $\frac{1}{\gamma^2}\mathbf{N}$, we first review some basic properties of the symmetric Skellam distribution $\mathrm{Sk}(\mu)$. The distribution of $\mathrm{Sk}(\mu)$ is obtained by taking the difference of two independent Poisson random variates sampled from $\mathrm{Pois}(\mu)$. The moment generating function of $Z \sim \mathrm{Sk}(\mu)$ is written as follows

$$\mathbb{E}[e^{\lambda Z}] = e^{\mu(e^\lambda + e^{-\lambda} - 2)}. \tag{21}$$

In particular, for $\frac{1}{\mu} \in [0, 1)$, we have that $e^{\frac{1}{\mu}} + e^{-\frac{1}{\mu}} - 2 \leq 1.09\frac{1}{\mu^2}$. Hence, for $\mu > 1$, we can bound the sub-exponential norm of Z as μ. [95] show that the spectral norm of the n-by-n symmetric random matrix whose entries are distributed as a sub-exponential random variable with norm μ is bounded by $O(\sqrt{n}\mu)$ with high probability. Now that $\mu = O(\gamma^4)$ (see Lemma 6), the factor of $\sqrt{\mu}$ gets canceled when dividing it by γ^2 in Eq. 20, leaving $O(\sqrt{n}\sqrt{\mu})$ in the end. □

The proof of Lemma 7 then follows from Lemma 14 and the fact that to achieve (ϵ, δ)-cite server-observed DP, it suffices to set $\mu_{\epsilon,\delta} = b^2\log(1/\delta)/\epsilon^2$, where b is some constant [96].

Similarly, we can obtain the privacy-utility trade-off for cite SPCA under the RDP framework by replacing $\sqrt{\mu_{\epsilon,\delta}}$ with $\sqrt{(1.09\alpha + 0.91)/\tau}$, formalized as follows.

Lemma 15. *Let $\mathbf{V}_k$ be the rank-k subspace of the original matrix $\mathbf{X}$ and let $\tilde{\mathbf{V}}_k$ be the principal rank-k subspace of matrix $\tilde{\mathbf{C}}$ obtained from Algorithm 3 with discretization parameter $\gamma \gg n$. Then Algorithm 3 satisfies (α, τ)- server-observed RDP and with high probability, we have that*

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - O(k\sqrt{n\alpha/\tau}). \tag{22}$$

Proof to Lemma 8. We sketch the proof as follows. We first obtain the upper bound for the $\mathcal{L}_2$ norm of Eq. (12) (a 2nd-degree polynomial of dimension $d = n - 1$) on the processed inputs. For the first term $\frac{1}{2}\cdot\mathbf{x}$ (monomial of degree 1), the processed outcome is $\gamma^3\frac{1}{2}\cdot\mathbf{x}$ plus an error of at most $2|x_j|\gamma$ in each dimension j (recall from Eq. (13), (14), (15), (16)). Similarly, for the second term $\langle\frac{\mathbf{w}}{4}, \mathbf{x}\rangle\cdot\mathbf{x}$ (monomial of degree 2), the processed outcome is $\gamma^3\langle\frac{\mathbf{w}}{4}, \mathbf{x}\rangle\cdot\mathbf{x}$ plus an error of at most $2\lambda|x_j|\gamma^2$ in each dimension j. For the term $-y\cdot\mathbf{x}$, the processed outcome is $-\gamma^3 y\cdot\mathbf{x}$ with an error of at most $2|x_j|\gamma^2$. Overall, we can show that due to scaling and rounding, the maximum norm for evaluating Eq. (12) is bounded by $\sqrt{\gamma^6(\frac{3}{4})^2 12\gamma^5\sqrt{d}\frac{3}{4}\sqrt{d} + 36\gamma^4\|\mathbf{x}\|_2^2} + \sqrt{d}\frac{3}{4}$ and $\sqrt{d}$ corresponds to the maximum $\mathcal{L}_1$ norms of $f(\mathbf{w}, (\mathbf{x}, y))$ and $\mathbf{x}$, respectively. The rest of proof then follows from applying Lemma 2 to the computed $\mathcal{L}_2$ and $\mathcal{L}_1$ sensitivities and then use the results on privacy amplification by subsampling and composition theorems [56], [69], [70]. We note that τ_{client} does not benefit from subsampling, since each client already knows which record is placed in the sampled batch. □