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 vertical federated learning or 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 avoids reliance 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 make 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]. Consider a scenario where sensitive data is distributed across multiple database owners, each possessing only a portion of the dataset. While these database owners must keep their private information confidential, they also seek to collaborate in analyzing the sensitive data or training machine learning models. This challenge has led to the development of federated learning (FL) [13], which enables collaborative learning while keeping data decentralized. FL can be categorized based on how data is partitioned: **(i)** in horizontal FL (HFL), each party owns a subset of the records; **(ii)** in vertical FL (VFL), each party owns a subset of the attributes.

This paper focuses on VFL, which has promising practical applications (see [14] for a survey). For example, a search engine and an online payment service may each hold distinct but complementary data about users' search history and financial transactions, respectively, and their user bases often overlap. VFL allows these entities to jointly train a machine learning model to predict user behavior without sharing sensitive data.

Earlier FL protocols relies on a centralized coordinator (or server) that facilitates collaboration between data owners [15], [16]. However, a key concern with this approach is that the coordinator, which could potentially be adversarial, might gain access to sensitive information during the collaborative training process. To mitigate such risks, cryptographic techniques like secure multiparty computation (MPC) and homomorphic encryption (HE) are commonly used in FL [6], [9], [11], [17]. While these techniques encrypt the training process, they do not prevent information leakage from the final output of FL (i.e., the model parameters), which remains vulnerable to data extraction attacks [18], [19]

Differential privacy (DP) [20], [21] is a rigorous framework that provides strong privacy guarantees by injecting random noise into the outcome of a function (e.g., model parameters). The noise scale is calibrated to the *sensitivity*, which measures the maximum influence of any single record on the function's output. DP has been widely adopted in the database community [22]–[29] and successfully implemented in real-world applications [30]–[32].

Challenges in adopting DP in VFL. Enforcing DP in vertical federated learning (VFL) while maintaining utility presents unique challenges:

1. Noise injection without a trusted party: In centralized DP, a trusted entity injects noise after performing the analysis [28], [33], [34], which is infeasible in VFL due to the lack of a trusted party. Local DP, which requires each client to perturb their own data independently, avoids this issue but results in excessive noise accumulation, significantly degrading utility [35]–[38].

2. Sensitivity analysis in VFL: To maintain a strong privacy-utility trade-off, the noise scale must be calibrated to the function's sensitivity. In HFL, where computations involve summing individual contributions (e.g., gradients), sensitivity

Part of this work was done when Ergute Bao was an intern at SEA AI Lab.

can be controlled via clipping mechanisms [39]–[41]. However, in VFL, even computing a function over a single record requires multiple parties to collaborate, making it impossible for any single party to enforce a sensitivity bound. This lack of control over sensitivity further complicates DP enforcement in VFL.

Distributed DP: a potential solution. In distributed settings, a more effective alternative to local DP is the distributed DP framework [41]–[44], which injects a far lower amount of noise (meaning higher result utility) than local DP, while providing a strong privacy guarantee. The overall workflow of distributed DP is as follows. First, each client injects noise locally to their partitioned data or intermediate results. Next, the clients collectively run some cryptography protocol (e.g., [45]) to aggregate the noisy results securely. Finally, the aggregated output is revealed to all clients (or an un-trusted server). By leveraging cryptographic techniques, distributed DP simulates a trusted party who hides the identities of the locally perturbed results, amplifying the privacy protection while preserving utility. This approach is able to achieve privacy-utility trade-offs comaprable to centralized DP (we refer to [39], [43] for more detailed analysis).

Gaps. Despite its potential, existing distributed DP solutions are limited to HFL, where they typically assume that the function of interest is linear, such as aggregating gradients [39]–[41] and that the data is horizontally partitioned, so that each client can independently clip local computation results to control sensitivity. To our knowledge, there are no effective distributed DP mechanisms for VFL. Consequently, we are left with rather basic local DP approaches that lead to poor utility.

Our objective. In this work, we aim to bridge this gap by designing a novel distributed DP protocol for VFL that eliminates the need for a trusted party, and achieves utility comparable to centralized DP while preserving a strong privacy guarantee. In particular, we assume no party is trusted by any party, and all participants honestly follow the protocol. Namely, we assume the semi-honest threat model, consistent with prior HFL research [39], [41]).

Additional challenges in VFL with distributed DP. Designing an effective distributed DP mechanism for VFL requires addressing several technical challenges:

1. Numerical precision issues. Classic DP mechanisms (e.g., Laplace and Gaussian mechanisms [21], [46]) operate in the continuous domain, requiring infinite precision. In contrast, distributed DP relies on MPC protocols that 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 due to rounding errors, as pointed out in [39], [49], [50].

2. Secure noise sampling via MPC. In the literature, discrete counterparts of continuous DP mechanisms (e.g., the discrete Gaussian mechanism [51]) have been proposed to resolve the precision issue. Extending these solution to the VFL setting, however, is non-trivial. First, letting the clients use shared randomness to jointly sample a discrete noise [3] is non-private in the presence of a curious client, who has observed the noise outcome and can infer the exact result. On the other hand, securely sampling discrete noises using MPC is a complicated task, and existing solutions often incur privacy overhead or utility degradation, according to [52], [53], and could also be 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].

3. Sensitivity analysis in discrete domains. In centralized DP and HFL, sensitivity can be controlled via clipping (e.g., gradient clipping in [54]). However, in VFL, as we have mentioned, no single party can enforce a sensitivity bound, making traditional clipping mechanisms inapplicable.

A. Our solution

In this paper, we propose a distributed DP mechanism for VFL that effectively preserves privacy without a trusted entity. Our method applies to scenarios where the function of interest (e.g., model training) can be expressed as a polynomial over the data. In the proposed solution, the participants utilize a general-purpose MPC protocol to evaluate the target polynomial function with integer-valued DP noise injected, explained as follows.

To avoid the aforementioned numerical 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. 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 0.011 may be rounded to 1, leading to roughly 100x sensitivity increase and, thus, asignificantly higher amount of random noise required to satisfy DP.

Data quantization. We address this problem through *local quantization*, in which each party up-scales the original inputs by a factor before rounding. The intuition is that the impact due to rounding on *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. 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 approaches 0 as we increase the scaling factor.

Coefficient/polynomial quantization. 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. The reason is that scaling the original inputs by some factor would lead un-tractable 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. 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. In addition, we do not need any single party to perform some artificial clipping mechanism to control the sensitivity.

Noise injection. Next, we need to design an effective noise injection algorithm to satisfy DP. Specifically, we require that the overall DP noise follow the discrete 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 [40], [41]. To do so, we let each party to privately sample a "share" of Skellam noise locally. 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. 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 independent 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.

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 are quantization and the Skellam noises described above (see Figure 1 for an illustration). We 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. 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.

SQM is also an MPC-friendly framework in the sense that the clients only need to utilize MPC as a black box after careful quantization of the target function and the partitioned data, as well as the local generation of DP noises. The overhead incurred during the data quantization and noise generation on the client side is negligible, compared with the cost of the MPC protocol itself.

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

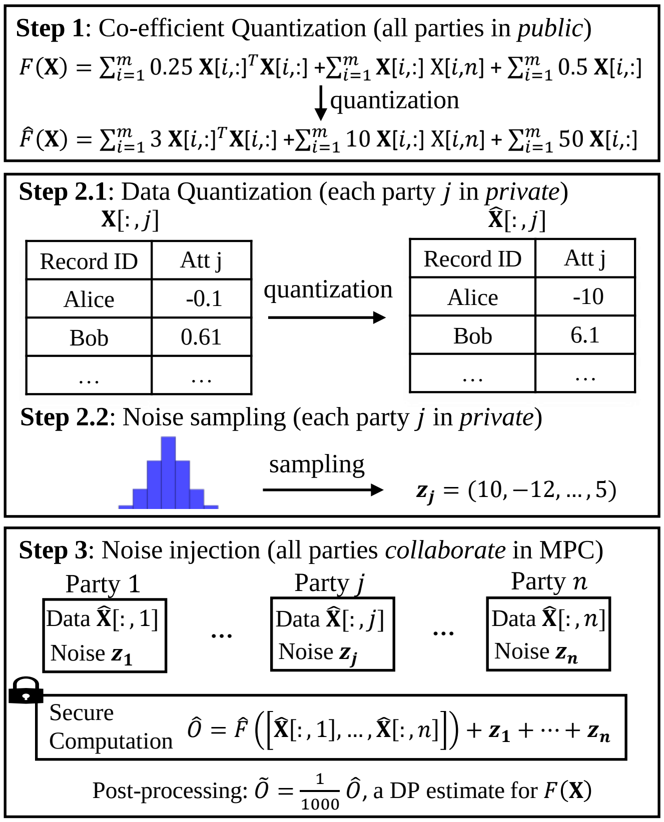
Step 1: Co-efficient Quantization (all parties in *public*)
$F(\mathbf{X}) = \sum_{i=1}^{m} 0.25\, \mathbf{X}[i,:]^T \mathbf{X}[i,:] + \sum_{i=1}^{m} \mathbf{X}[i,:]\, X[i,n] + \sum_{i=1}^{m} 0.5\, \mathbf{X}[i,:]$
quantization
$\hat{F}(\mathbf{X}) = \sum_{i=1}^{m} 3\, \mathbf{X}[i,:]^T \mathbf{X}[i,:] + \sum_{i=1}^{m} 10\, \mathbf{X}[i,:]\, X[i,n] + \sum_{i=1}^{m} 50\, \mathbf{X}[i,:]$
Step 2.1: Data Quantization (each party *j* in *private*)
$\mathbf{X}[:\,,j]$
Record ID | Att j
Alice | -0.1
Bob | 0.61
… | …
quantization
$\hat{\mathbf{X}}[:\,,j]$
Record ID | Att j
Alice | -10
Bob | 6.1
… | …
Step 2.2: Noise sampling (each party *j* in *private*)
sampling
$\boldsymbol{z_j} = (10, -12, \ldots, 5)$
Step 3: Noise injection (all parties *collaborate* in MPC)
Party 1
Data $\hat{\mathbf{X}}[:\,,1]$
Noise $\boldsymbol{z_1}$
…
Party j
Data $\hat{\mathbf{X}}[:\,,j]$
Noise $\boldsymbol{z_j}$
…
Party n
Data $\hat{\mathbf{X}}[:\,,n]$
Noise $\boldsymbol{z_n}$
Secure Computation
$\hat{O} = \hat{F}\left(\left[\hat{\mathbf{X}}[:\,,1], \ldots, \hat{\mathbf{X}}[:\,,n]\right]\right) + \boldsymbol{z_1} + \cdots + \boldsymbol{z_n}$
Post-processing: $\tilde{O} = \frac{1}{1000}\hat{O}$, a DP estimate for $F(\mathbf{X})$

Fig. 1: Example of SQM for evaluating polynomial functions in VFL.

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.

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*

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

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) [55], which quantifies indistinguishability under Rényi divergence.

Definition 2 (Rényi DP [55]). *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 function's *sensitivity* [21], denoted as $S(F) = \max_{\mathbf{X}\sim\mathbf{X}'} \|F(\mathbf{X}) - F(\mathbf{X}')\|$. The maximum is taken over all pairs of neighboring databases, and $\|\cdot\|$ represents a norm measure, e.g. $\mathcal{L}_2$-norm. In particular, 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 [55].

Secure multiparty computation. Secure multiparty computation (MPC) [56] 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 [45], [57]) and interested readers are referred to [58] 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 [59], which has mature and efficient implementations, as reported in [60]. 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 without affecting the DP guarantees, as long as the protocol is privacy preserving under the semi-honest threat model, e.g., Sharemind [61] and ABY3 [62].

Skellam noise for DP. Integer-valued noises for enforcing DP are compatible with MPC. The state-of-the-art is the Skellam noise [40], [41], 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 $Pois^d(\mu)$ (the ensemble of d independent $Pois(\mu)$). The Skellam noise almost achieves the same level of privacy-utility trade-off as the continuous Gaussian noise, formalized as follows.

Lemma 1 (Skellam noise preserves RDP [40], [41]). *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*

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

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 $F(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$, 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 [63], 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, all parties (including the clients and the server) follow the specification of mechanism $\mathcal{M}$ while trying to infer the private inputs of other parties. This threat model have been commonly adopted in the distributed DP literature [39], [40]. 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,

$$\sup_{\mathbf{X}\sim\mathbf{X}'} D_\alpha(\mathcal{M}_{\text{server}}(\mathbf{X}) \,\|\, \mathcal{M}_{\text{server}}(\mathbf{X}')) \leq \tau_{\text{server}}, \text{ and} \quad (3)$$

$$\sup_{\mathbf{X}\sim\mathbf{X}'} D_\alpha(\mathcal{M}_{\text{client}_j}(\mathbf{X}) \,\|\, \mathcal{M}_{\text{client}_j}(\mathbf{X}')) \leq \tau_{\text{client}} \quad (4)$$

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. (3) and Eq. (4) into consideration for the privacy requirements.

Proposition 1. *A mechanism that satisfies Eq.* (3) *can violate Eq.* (4) *(i.e., non-private for a curious client). Similarly, a mechanism that satisfies Eq.* (4) *can violate Eq.* (3) *(i.e., non-private for a curious server).*

Proof sketch. We prove 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, since 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).

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 [39], [43].

Algorithm 1: SQM for One-dimensional Monomials

```
Input: One-dimensional monomial function $f$ of degree $\lambda$;
       private input database $\mathbf{X}$ that is partitioned among $n$
       clients; scaling parameter $\gamma$; noise parameter $\mu$.
1 for each client $j \in [n]$ do
2     $\hat{\mathbf{X}}[:, j] \leftarrow$ Algorithm 2 $(\mathbf{X}[:, j], \gamma)$. // the processed data
      portions constitute database $\hat{\mathbf{X}}$.
3 for each client $j \in [n]$ do
4     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 $\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$.
```

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. The detailed proofs of our claims can be found in the full version [64].

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. (3) and (4). 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 2: Data Pre-processing/Quantization

```
Input: Real-valued vector $\mathbf{v} \in \mathbb{R}^m$; scaling factor $\gamma$.
$\hat{\mathbf{v}} \leftarrow \gamma \cdot \mathbf{v}$.
for Dimension $j \in [m]$ do
    Flip a coin with heads probability $\hat{\mathbf{v}}[j] - \lfloor \hat{\mathbf{v}}[j] \rfloor$.
    if Heads then
        $\hat{\mathbf{v}}[j] \leftarrow \lfloor \hat{\mathbf{v}}[j] \rfloor + 1$.
    else
        $\hat{\mathbf{v}}[j] \leftarrow \lfloor \hat{\mathbf{v}}[j] \rfloor$.
Output: Integer-valued vector $\hat{\mathbf{v}} \in \mathbb{Z}^m$.
```

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

$$\hat{y}(\mathbf{X}) = \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \hat{f}(\hat{\mathbf{x}}) + \sum_{j=1}^{n} \mathbf{Z}_j. \tag{5}$$

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 $Z^* \sim Sk\left(\frac{n-1}{n}\mu\right)$ (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 2. *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 2 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 2 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 3 (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 1.

Privacy-utility trade-off. How good is our SQM? We follow prior work on HFL [39], [41] 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 3, 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 $\frac{\alpha}{2} \cdot \frac{\max |f(\mathbf{x})|^2}{2\tau}$; 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

$$f_t(\mathbf{x}) = \sum_{l=1}^{v_t} a_t[l] \cdot \Pi_{j=1}^{n} x[j]^{B_t[l\,,j]}, \tag{6}$$

for some constant $v_t \geq 1$. 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 2 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.

```
Algorithm 3: SQM for Multi-dimensional Polynomials
Input: Multi-dimensional polynomial function $f$ of degree $\lambda$
       and dimension $d$; private input database $\mathbf{X}$ that is
       partitioned among $n$ clients; scaling parameter $\gamma$;
       noise parameter $\mu$.
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 $\sum_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$.
```

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 2 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 4. *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 4 and 3. 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 2 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.

Finally, we remark that SQM is an MPC-friendly framework in the sense that the clients only need to utilize MPC as a black box after careful quantization of the target function and the partitioned data, as well as the local generation of DP noises. As we can see from Algorithm 3, the clients collectively run a BGW protocol as a black box (which can be replaced by other MPC protocols, e.g., ABY [65] if there are only two clients) to evaluate the noisy outcome of the function. In Section VI, we will show that the overhead due to enforcing differential privacy is negligible, compared with the cost of the MPC protocol itself, which is needed in the first place to avoid revealing the data partitions in plain-text form.

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 [64].

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

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

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

$$\|\hat{\mathbf{X}}^T\hat{\mathbf{X}} - \hat{\mathbf{X}}'^T\hat{\mathbf{X}}'\|_F \le \sum_j \left(\hat{\mathbf{X}}[m,j]\right)^2 \le \gamma^2c^2 + n. \quad (7)$$

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 γ^2c^2 when γ is large enough. Combining Eq. (7) with Lemma 1, we have the DP guarantees.

Lemma 5 (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 $\Delta_2 = \gamma^2c^2 + n$, *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 6 (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*

$$\|\mathbf{X}\tilde{\mathbf{V}}\|_F^2 \ge \|\mathbf{X}\mathbf{V}\|_F^2 - O(k\sqrt{n}\sqrt{\mu_{\epsilon,\delta}}), \quad (8)$$

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 5) and the error is still in $O(k\sqrt{n})$. Both results indicate the optimality of our solution. The key to our proof is to show that the error matrix due to discretization has a bounded spectral norm (in particular, it is $O(n)$ with high probability), which becomes negligible when γ is large enough. We refer interested readers to [67], [68] for useful tools for analyzing the spectral norms of random matrices.

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 [69]. 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 [70] 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 $g((\mathbf{x}, y)) = \sigma(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x} - y \cdot \mathbf{x}$.

Reduction to polynomial evaluation. Due to the sigmoid function, the above gradient is not a polynomial of $(\mathbf{x}, y)$. Here we follow [69] 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 $u = 0$. We consider the first-order approximation (higher order approximations can be done in a similar fashion). Then we can write $\sigma(u) \approx \frac{1}{2} + \frac{1}{4} \cdot u$, and approximate the gradient as $g((\mathbf{x}, y)) \approx \frac{1}{2} \cdot \mathbf{x} + \frac{1}{4} \cdot (\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x} - y \cdot \mathbf{x}$.

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

$$f(\mathbf{w}, (\mathbf{x}, y)) = \frac{1}{2} \cdot \mathbf{x} + \langle \frac{\mathbf{w}}{4}, \mathbf{x}\rangle \cdot \mathbf{x} - y \cdot \mathbf{x}. \tag{9}$$

First, the server randomly initializes the model weight $\mathbf{w}$, and clips $\|\mathbf{w}\|_2$ to 1. Next, in the first and 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 , and share the outcome (i.e., perturbed gradient sum) with the server, which then updates $\mathbf{w}$ accordingly, followed by clipping.

Lemma 7 (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*

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

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

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. (9)). 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 1 and then applying the results on privacy amplification by subsampling and composition theorems [55], [71], [72] 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 data partitioning. The server-level privacy guarantees of SQM depend on the quantization level, which is characterized by γ, and the overall scale of the injected noises contributed by all clients, which is characterized by μ. In other words, the number of clients and data partitioning has no impact on the performance (i.e., privacy-utility trade-off) of our SQM. This is due to our design of SQM, where the clients collectively simulate a trusted central server — who performs the noise injection — using MPC. The client-level privacy depends on an additional factor of $\frac{P}{P-1}$ (where P represents the number of clients), since each client knows the local noise she injected. We note that this factor $\frac{P}{P-1}$ converges to 1 as the number of clients P increases. In our experiments, we focus on evaluating the performance of SQM under fixed server-level privacy constraints (following prior work in horizontal federated learning [39]), and assume that each client partitions one column of the input data matrix, without loss of generality.

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 [73], as we have mentioned in Section I. In our SQM, the sensitivity overhead due to rounding (recall Algorithm 2) is at most 1 in each dimension of the quantized data. Through careful analysis, we show that this overhead results in negligible error in the final result — it appears in the small-oh term in Lemma 4.

On complexities. We briefly explain the complexities of running SQM, with BGW [59] as the underlying MPC protocol. Recall that the data matrix $\mathbf{X}$ has m rows and n columns, and we assume columns of $\mathbf{X}$ are evenly partitioned to P parties.

For PCA, we use BGW to compute $\mathbf{X}^T\mathbf{X}+\sum_{j=1}^{P}\mathbf{Z}_j$, where $\mathbf{Z}_j$ is the $n\times n$ noise matrix generated by client j. The overall time complexity is $O(n^2 m\log m)$; the computation complexity per client is $O(mP+n^2 m\log m/P+n^2)$; and the communication complexity is $O(n^2 mP\log\gamma)$. For LR, in each epoch, we use BGW to compute m inner products for $(n-1)$-dimensional vectors, and m additions and m subtractions for scalars, and m multiplications between a scalar and an $(n-1)$-dimensional vector, and then m summations for $(n-1)$-dimensional vectors. It also involves P summations for aggregating the local Skellam noises. As a result, the overall time complexity is $O(m(n-1)\log m)$; the computation complexity per client is $O(m(n-1)P+m(n-1)\log m/P)$; and the communication complexity is $O(m(n-1)P\log m\log\gamma)$. Overall, for both LR and PCA, the cost due to enforcing DP, which involves P summations over the Skellam noises, is negligible compared with the costs needed for computing the noise-less function using MPC, which avoids revealing the client's data in plaintext form. We provide simulation results in the next section.

Extension to more complicated functions. We note that although our SQM is restricted to evaluating polynomials, it can be extended to other functions that can be approximated as polynomials, e.g., the gradients of complicated neural networks [74]. Overall, if the target function can be approximated by a polynomial with arbitrarily small error, then using our SQM, we can evaluate this function under a VFL setup with differential privacy, where the error overhead (due to approximation and discretization) is arbitrarily small compared with the centralized setting. To compute the gradients of neural networks in general, simple approximations such as the Taylor expansion do not give a small approximation error, and more delicate approximations are needed. Finding polynomial approximations to complicated functions is also one of the major challenges in applying cryptography to deep learning (e.g., see [63], [74]–[76]). Another issue is the cost of running cryptography protocols, e.g., the attention layer alone in a transformer would take Gigabytes of communication and hours to compute in a secure manner [77] (not to mention larger model structures). Both aspects are beyond our scope.

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 [39]–[41], we focus on measuring the performance for the fitted models in terms of privacy-utility trade-offs. We use a single Ubuntu machine with 64 CPUs 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) 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], [78], [79]). However, their threat models and privacy requirements largely differ from ours and hence, they are not directly comparable with ours (see Section VII for detailed explanations). Overall, we want to demonstrate that SQM achieves comparable utility-privacy trade-off as the central-DP baseline, as we increase the quantization granularity (characterized by γ). Other local DP solutions (e.g., [5], [80]), no matter how advanced, exhibit notable performance gaps between the central-DP baseline and are omitted.

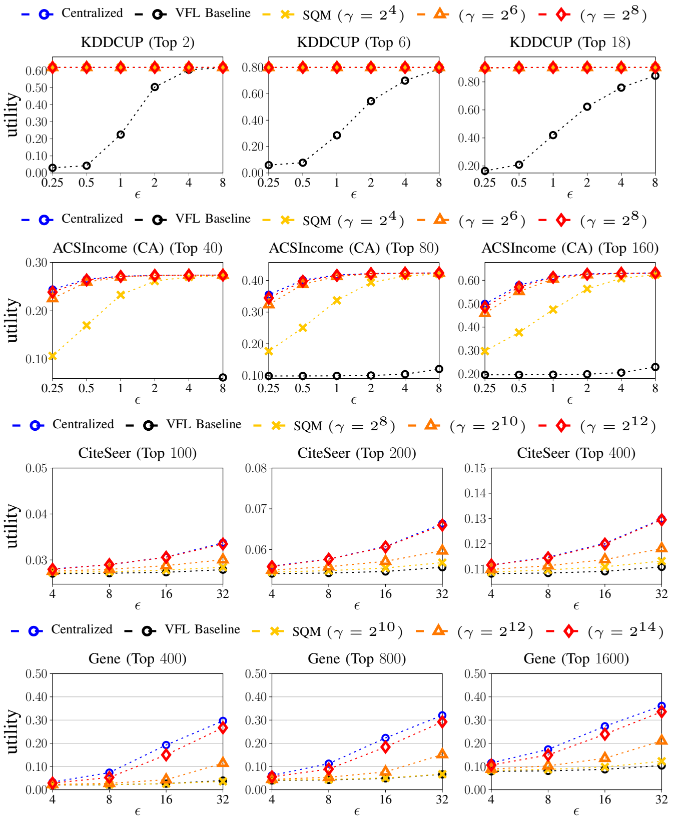

Fig. 2: Performance of SQM on PCA, for different datasets and varying numbers of top components and DP constraints.

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 [54]. 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 [81] with $m = 195666$ and $n = 117$, ACSIncome (CA) with $m \approx 100000$ and $n = 800$, and high dimensional datasets, CiteSeer [82] with $m = 2110$ and $n = 3703$ and Gene [83] with $m = 801$ and $n = 20531$. All datasets are partitioned to n clients. We report the utility 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

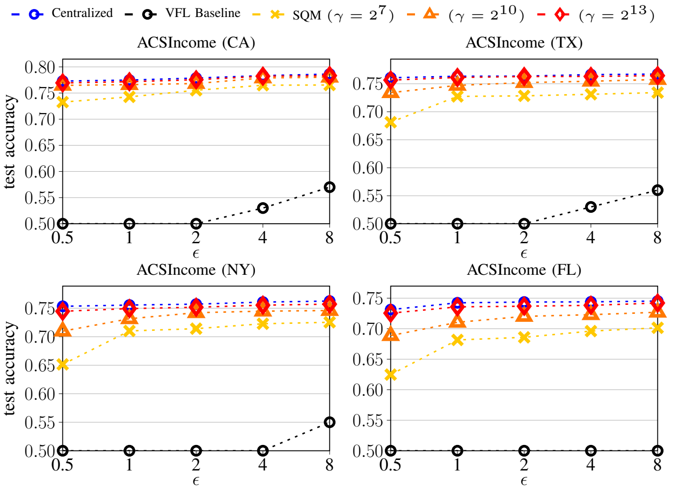

Fig. 3: Performance of SQM on logistic regression, for different datasets and DP constraints.

TABLE I: Overall time costs of SQM and overhead due to DP. The database contains $m = 1000$ records. The n attributes are evenly partitioned among $P = 4$ clients.

Task: principal component analysis (PCA)

Data dimension (n)	Overall time (sec)	Time for noise injection (sec)
20	1.54	0.41
100	16.25	0.54
500	393.47	4.38
2500	10152.09	103.76

Task: logistic regression (LR)

Data dimension (n)	Overall time (sec)	Time for noise injection (sec)
20	1.03	0.40
100	1.43	0.41
500	3.65	0.42
2500	14.73	0.47

TABLE II: Overall time costs of SQM and overhead due to DP. The database contains $m = 1000$ records. The $n = 500$ attributes are evenly partitioned among P clients.

Task: principal component analysis (PCA)

Number of clients (P)	Overall time (sec)	Time for DP (sec)
4	200.33	4.19
10	388.55	17.14
20	699.58	58.67

Task: logistic regression (LR)

Number of clients (P)	Overall time (sec)	Time for DP (sec)
4	2.24	0.42
10	4.66	1.08
20	8.58	2.23

TABLE III: Overall time costs of SQM and overhead due to DP. The database contains m records. The $n = 500$ attributes are evenly partitioned among $P = 4$ clients.

Task: principal component analysis (PCA)

Number of records (m)	Overall time (sec)	Time for DP (sec)
20	12.13	4.23
100	41.31	4.24
500	200.33	4.19
2500	1020.55	4.91

Task: logistic regression (LR)

Number of records (m)	Overall time (sec)	Time for DP (sec)
20	0.99	0.42
100	1.19	0.42
500	2.24	0.42
2500	7.96	0.42

fix δ to 10^{-5} and vary ϵ. For KDDCUP and ACSIncome (CA), we vary ϵ from 0.25 to 8. For the high-dimensional CiteSeer and Gene, we vary ϵ from 4 to 32. Our choices of ϵ align with the common practices in real-world applications. To give a few examples, Apple sets ϵ within the range of 2 to 16, depending on the type of user data to collect [84], and the 2020 US Census adopts an ϵ close to 20 [85]. For SQM, we also vary the scaling parameter γ. For the high-dimensional CiteSeer and Gene, we choose relatively larger γ's to control the sensitivity overhead, maintaining a high signal-to-noise ratio.

The average utility over 20 independent runs is in Figure 2. 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 [86]. 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 3.

In Figure 3, 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,

TABLE IV: Comparison with existing VFL DP solutions.

Threat model: *benign* clients and *honest-but-curious* server	
Approach (which client(s) sample the DP noise)	Task
[3] (n clients using shared randomness)	Decision tree
[78] (one of the clients)	Logistic regression
[79] (one of the clients)	Logistic regression
Threat model: *honest-but-curious* clients and server	
Approach (which client(s) sample the DP noise)	Task
[5] (n clients independently – local DP)	K-means clustering
Ours (n **clients independently – distributed DP)**	Polynomial evaluation

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.

On the overhead due to DP. We assess the overall time cost of our SQM for PCA and LR tasks by comparing the primary computational cost-covariance matrix calculation for PCA and gradient computation for LR—with the additional cost of enforcing DP via noise injection, represented by summing P noise vectors/matrices (where P is the number of clients). We use a single Ubuntu machine with 64 CPUs to *simulate* BGW protocol under a distributed environment where each party is assumed to have a secure and noiseless channel for communication. We fix the time cost for message passing from one client to another to 0.1 second. We fix $\gamma = 18$. Our results in Tables I, II, and III show that the overhead due to DP is negligible in general.

VII. Related Work

Federated learning over vertically partitioned databases has been extensively studied in the database community, e.g., see [1]–[10], [12], [87]. We refer readers to [14] for a comprehensive survey. Preserving data privacy for analyzing databases jointly owned by several clients predates differential privacy (DP) and federated learning (FL) [88]–[90]. 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.

Amplification of DP in horizontal FL. The idea of utilizing cryptographic protocols to enhance utility under fixed DP constraints originates from distributed DP in horizontal federated learning [43], [44]. The most relevant works with ours are [40], [41], where 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.

Other DP mechanisms in vertical FL. Recent works that try to enforce DP for VFL adopt different threat models (assumption for adversaries) than ours. Specifically, in works such as [78], [79], 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 leads to privacy violations in our threat model (where no party can be trusted) 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 our setting, as the clients can infer the noise-less result based on the sampled noise. Li et al. [5] study K-means clustering under the local DP model; as a consequence, their solution does not achieve centralized-DP like privacy-utility trade-offs. Extending our solution to their problem may narrow the performance gap to centralized-DP. The caveat is that SQM supports only polynomials for now, while their current solution for K-means clustering involves computing the minimum for a set of values (which cannot be expressed as a polynomial). We leave this extension of SQM as future work. We summarize the main differences between our work and the above ones in Table IV.

Numerical issues. It is important that the practical implementation of a DP mechanism aligns with its theoretical privacy analysis; otherwise, it may result in privacy violations. As revealed in Mironov's seminal work [50], the naive implementation of the continuous Laplace noise in finite-precision computers does not preserve DP. Integer-valued noises are a solution to this numerical issue, e.g., see [39], [40], [51], [91], [92]. In this work, we adopt the Skellam noise [40], [41] that can implemented in distributed systems with finite computation and communication capabilities.

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.

For future work, we plan to extend our mechanism to other application scenarios, such as training large-scale models and analyzing distributed social networks over distributed and sensitive user data.

Acknowledgment

The authors would like to thank Weiran Liu for useful discussions about approximating target functions with polynomials in secure multiparty computation.

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

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

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[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] 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.

[55] I. Mironov, "Rényi differential privacy," in *CSF*, 2017, pp. 263–275.

[56] A. C. Yao, "How to generate and exchange secrets," in *FOCS*, 1986, pp. 162–167.

[57] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *S&P*, 2017, pp. 19–38.

[58] G. Oded, *Foundations of Cryptography*, USA, 2009.

[59] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *STOC*, 1988, p. 1–10.

[60] A. Ben-Efraim, Y. Lindell, and E. Omri, "Optimizing semi-honest secure multiparty computation for the internet," Cryptology ePrint Archive, 2016.

[61] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *ESORICS*, 2008, p. 192–206.

[62] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *CCS*, 2018, p. 35–52.

[63] 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.

[64] E. Bao, F. Wei, Y. Yang, X. Xiao, T. Pang, and C. Du. (2025) Towards learning on vertically partitioned data with distributed differential privacy. [Online]. Available: https://github.com/erguteb/erguteb.github.io/blob/main/files/ICDE_p534_technical_report.pdf

[65] D. Demmler, T. Schneider, and M. Zohner, "Aby - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[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] R. Vershynin, "Spectral norm of products of random and deterministic matrices," *Probability Theory and Related Fields*, vol. 150, pp. 471–509, 2011.

[68] 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.

[69] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *PVLDB*, p. 1364–1375, 2012.

[70] 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.

[71] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled gaussian mechanism," *CoRR*, vol. abs/1908.10530, 2019.

[72] Y. Zhu and Y.-X. Wang, "Poission subsampled Rényi differential privacy," in *ICML*, 2019, pp. 7634–7642.

[73] 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.

[74] R. E. Ali, J. So, and A. S. Avestimehr, "On polynomial approximations for privacy-preserving and verifiable relu networks," *NeurIPS PPML Workshop*, 2020.

[75] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *CCS*, 2017, p. 619–631.

[76] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: private inference on transformers," in *NeurIPS*, 2022.

[77] Y. Akimoto, K. Fukuchi, Y. Akimoto, and J. Sakuma, "Privformer: Privacy-preserving transformer with mpc," in *Euro S&P*, 2023, pp. 392–410.

[78] D. Xu, S. Yuan, and X. Wu, "Achieving differential privacy in vertically partitioned multiparty learning," in *Big Data*, 2021, pp. 5474–5483.

[79] T. Ranbaduge and M. Ding, "Differentially private vertical federated learning," *CoRR*, vol. abs/2211.06782, 2022.

[80] D. Wang and J. Xu, "Principal component analysis in the local differential privacy model," *TCS*, vol. 809, pp. 296–312, 2020.

[81] 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.

[82] 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.

[83] S. Fiorini, "Gene expression cancer RNA-Seq," UCI Machine Learning Repository, 2016.

[84] Apple. (2016) Differential privacy overview. [Online]. Available: https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf

[85] S. Garfinkel. (2022) Differential privacy and the 2020 us census. [Online]. Available: https://mit-serc.pubpub.org/pub/differential-privacy-2020-us-census/release/2

[86] F. Ding, M. Hardt, J. Miller, and L. Schmidt, "Retiring adult: New datasets for fair machine learning," in *NeurIPS*, 2021, pp. 6478–6490.

[87] 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.

[88] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD*, 2002, p. 639–644.

[89] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *CRYPTO*, 2004, pp. 528–544.

[90] A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining," in *PODS*, 2003, p. 211–222.

[91] 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.

[92] C. Ilvento, "Implementing the exponential mechanism with base-2 differential privacy," in *CCS*, 2020, p. 717–742.

[93] B. Balle and Y.-X. Wang, "Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising," in *ICML*, 2018.

[94] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[95] 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 8 (Analytic Gaussian Mechanism [93])**.** *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.

Lemma 9 (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 RDP mechanisms also satisfies RDP.

Lemma 10 (Composition Lemma for RDP [55])**.** *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.

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 $\sigma$.
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 (Subsampling for RDP [71])**.** *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*

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

B. The BGW Protocol

For completeness, we briefly review the idea of BGW [59] in the following. A building block of BGW is Shamir's secret sharing algorithm [94], 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 [94], BGW [59] implements a three-phase execution.

1) First, each party distributes her private input as secret shares to other clients using Shamir's algorithm [94]. 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.

C. Baseline Solution for VFL with DP

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, this method applies to arbitrary tasks but incurs high errors.

We present the privacy guarantees for Algorithm 4.

Lemma 12. *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 $\tau_{client} = \frac{\alpha 2 c^2}{\mu}$.*

The proof follows from the RDP guarantee of Gaussian noises. Note that τ_{server} and τ_{client} defer by a factor of 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 8.

D. Proofs

Proof of Lemma 2. We first prove the case when the input record x one dimensional with $|x| \le c$. 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 $x \ge 0$. We first compute the upper bound for $(\hat{x})^\lambda$. We have

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

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

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

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

$$(\hat{x})^\lambda \le \gamma^\lambda x^\lambda + 2\lambda\gamma^{\lambda-1}. \tag{10}$$

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

$$(\hat{x})^\lambda \le \gamma^\lambda x^\lambda + 2\lambda x^{\lambda-1}\gamma^{\lambda-1}. \tag{11}$$

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

$$(\hat{x})^\lambda \ge \gamma^\lambda x^\lambda - 2\lambda\gamma^{\lambda-1}, \tag{12}$$

when $0 \le x < 1$, and

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

Here we note that the term $\lambda x^{\lambda-1}$ in Eq. (11) and (13) is bounded by the constant $\lambda c^{\lambda-1}$, which becomes negligible with respect to $\gamma^{\lambda-1}$ when γ is large. Combining

Eq. (10), (11), (12), (13), we conclude that $(\hat{x})^{\lambda} = \gamma^{\lambda} x^{\lambda} + O(\gamma^{\lambda-1})$. □

Before we present the proof of Lemma 6, we first prove the following.

Lemma 13. *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*

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - O(k\sqrt{n}\sqrt{\mu}), \tag{14}$$

Proof of Lemma 13. 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.

$$\tilde{\mathbf{C}} = \gamma^2\mathbf{X}^T\mathbf{X} + \underbrace{2\gamma\mathbf{E}^T\mathbf{X} + \mathbf{E}^T\mathbf{E} + \mathbf{N}}_{\text{denoted as } \mathbf{H}}, \tag{15}$$

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

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - 2k\|\frac{1}{\gamma^2}\cdot\mathbf{H}\|_2. \tag{16}$$

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. [67] 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 $\mathbb{E}[e^{\lambda Z}] = e^{\mu(e^{\lambda}+e^{-\lambda}-2)}$. 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. [68] 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 5), the factor of $\sqrt{\mu}$ gets canceled when dividing it by γ^2 in Eq. 16, leaving $O(\sqrt{n}\sqrt{\mu})$ in the end. □

The proof of Lemma 6 then follows from Lemma 13 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 [95].

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 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 discretization parameter $\gamma \gg n$. Then Algorithm 3 satisfies (α, τ)- server-observed RDP and with high probability, we have that*

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - O(k\sqrt{n\alpha/\tau}). \tag{17}$$

Proof to Lemma 7. We sketch the proof as follows. We first obtain the upper bound for the $\mathcal{L}_2$ norm of Eq. (9) (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. (10), (11), (12), (13)). Similarly, for the second term $\frac{\mathbf{w}}{4}, \mathbf{x}\rangle\cdot\mathbf{x}$ (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. (9) 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 1 to the computed $\mathcal{L}_2$ and $\mathcal{L}_1$ sensitivities and then use the results on privacy amplification by subsampling and composition theorems [55], [71], [72]. We note that τ_{client} does not benefit from subsampling, since each client already knows which record is placed in the sampled batch. □