Learning on Vertically Partitioned Data with Distributed Differential Privacy

Abstract

Analysis over distributed data requires collaboration of the database owners and also demands privacy protection. This paper focuses on the scenario where the database is *vertically* partitioned onto the data owners (referred to as VFL), e.g., an e-commerce platform and an online payment service collaborate to build a model to predict user behavior. To avoid revealing their private data, the data owners commonly participate in a cryptographic protocol such as secure multiparty computation to perform model fitting. However, the resulting model may still leak sensitive information in the presence of increasingly sophisticated data extraction attacks. A rigorous solution to address 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 to our knowledge, there does not yet exist an effective solution for this purpose that does not rely on any trusted party. Consequently, practitioners are left with rather basic approaches with suboptimal model utility. Can we achieve privacy-utility trade-offs for VFL with DP comparable to the case of differentially private model fitting in the centralized setting?

In this paper, we give a positive answer to the above question for a wide range of data analytics tasks, including principal component analysis and logistic regression. In particular, we follow the distributed DP framework that does not require any trusted party, and propose an effective solution, called the Skellam Quantization Mechanism (SQM), for evaluating functions under VFL that can be expressed as polynomials. We formally prove the correctness of our algorithm, and provide a rigorous analysis showing that it is able to match the privacy-utility trade-offs in the centralized setting. Extensive experiments confirm the strong performance of SQM on real-world datasets.

ACM Reference Format:

. 2018. Learning on Vertically Partitioned Data with Distributed Differential Privacy. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Privacy-preserving data analysis and machine learning over partitioned databases has received great attention in the database community in recent years [34–37, 41, 47, 48, 72, 76, 77], where a dataset containing sensitive information is scattered among multiple database owners, who must keep their private information confidential and yet would like to collaborate to perform some analysis on the sensitive data or build a machine learning model with strong predictive power. This motivates federated learning (FL) [52]. In FL, data can be horizontally partitioned into multiple databases (i.e., each party/database owner possesses a subset of the records) or vertically partitioned (each party/database owner possesses a subset of the attributes). This paper focuses on the latter, commonly referred to as vertical FL (VFL) [49], which has promising practical applications. For instance, the search engine department and the online payment service department of Google may possess users' search history and financial transactions, respectively, and their user bases often have large overlaps. VFL enables these database owners to jointly train a machine learning model to predict user behavior (e.g., whether to allow a user to pay using a credit card), without sharing data with each other or relying on any trusted third party.

Earlier FL protocols involve a centralized coordinator who interacts with the data owners to build a model collaboratively [53, 66]. A concern with this approach is that the coordinator might infer sensitive information on the participants' data through the FL training process. This can be addressed through a cryptographic solution such as secure multiparty computation (MPC) and homomorphic encryption (HE) which preserves the confidentiality of each data owner (e.g., see [36, 41, 51, 76, 77]). However, the analysis results (e.g., the trained model) may still leak private information of the underlying database, especially in the presence of increasingly powerful data extraction attacks [40, 55], which might lead to violations of privacy regulations such as GDPR (https://gdpr-info.eu/) and CCPA (https://oag.ca.gov/privacy/ccpa). To address this issue, a rigorous framework is to compute the model with differential privacy (DP) [22, 25], a strong privacy standard that has been successfully adopted in practice [30, 54, 60].

Enforcing DP on vertically partitioned databases (or VFL) is a rather challenging task. Specifically, DP mechanisms commonly inject random noise into the analysis results (e.g., parameters of the trained model), where the scale of the noise is calibrated to the *sensitivity* of the analysis function, which corresponds to the maximum influence of an individual record on the analysis result. This ensures that the adversary cannot infer with high confidence whether or not a particular record is present in the underlying data, thus providing plausible deniability to the individuals involved in the data. In the literature, DP has been extensively studied in the centralized (e.g., see [10, 23, 81]) and local setting (e.g., see [17, 73, 74, 79]). The former assumes a centralized party who has access to all records and their attributes in the database, which is inherently incompatible to the distributed setting. The latter, referred to as local DP, perturbs the individual data records rather than the analysis results, which tends to require excessive noise, leading to low utility.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

A more effective alternative to local DP is the distributed DP framework, which combines MPC protocols with local DP [9, 15, 29]), and injects a far lower amount of noise (meaning higher result utility) than local DP, while providing a strong privacy guarantee. The overall idea of distributed DP is to use MPC as a simulation of a trusted party, who could perform the analysis and noise injection over the partitioned database in a centralized fashion. The problem, however, is that existing distributed DP solutions are limited to HFL; to our knowledge, there does not yet exist an effective distributed DP mechanism for VFL or vertically partitioned databases. Consequently, practitioners are left with very noisy local DP solutions. Hence, the main research question in this paper is: can we satisfy differential privacy in a VFL setup, and achieve high model utility comparable to the centralized setting, without relying on any trusted party?

We provide a positive answer to the above question by proposing effective solutions for VFL with distributed DP and no trusted party, which are applicable when the function of interest (e.g., the model fitting process) can be expressed as a polynomial with respect to the data [1]. In particular, we demonstrate that our solution apply to classical data analytics tasks such as principal component analysis (PCA) and logistic regression, and achieve result utility comparable to centralized DP setting.

Challenges. Designing an effective distributed DP mechanism for VFL is highly non-trivial, which involves overcoming several major challenges. First, we note that DP is usually designed under the continuous domain (e.g., the Laplace and Gaussian noise [25, 27]) that requires infinite precision to represent whereas computer systems as well MPC systems are implemented over discrete fields or floating-point numbers (e.g., [45, 63]). For MPC systems, it is even more difficult to operate over high-precision numbers, which demands for more computation and communication costs. Due to such a discrepancy between the theoretical analysis of DP and the practical implementation environment, naive implementation of (approximate) continuous DP mechanisms on discrete systems often leads to privacy violations, as pointed out in [42, 43, 56].

Furthermore, computation over finite-precision numbers also complicates the sensitivity analysis (originally performed on the continuous domain) that is also critical to establishing rigorous DP guarantees. As per the IEEE standard [1], results of numerical computation are required to be exactly rounded, which may lead to underestimation of the sensitivity (e.g., 0.9 is rounded to 1) and privacy violations. This sensitivity issue can be resolved in the centralized setting, where the trusted data curator could enforce an artificial clipping on the computation result to make sure the sensitivity is still bounded by some target constant (e.g., see [2] for individual gradient clipping). In VFL, this operation cannot be performed by a single party, as she/he would need to know the noise-less analysis result on the data partitioned by multiple parties in the first place, leading to privacy violations.

Discrete counterparts of continuous DP mechanisms are proposed to resolve the precision issue in DP. For example, the discrete Gaussian mechanism can be used as a drop-in replacement for the continuous Gaussian mechanism in DP applications [12]. Extending this to the distributed setting, however, is not trivial. Securely sampling discrete noises using MPC incurs privacy overheads or utility degradation, according to the current implementation [44, 75]. Besides, sampling noises using MPC is also vulnerable to timing attacks [42], leading to privacy violations–as the outcome of the sampled DP noise is highly correlated to the running time of the sampling protocol, on observing the execution time, the adversary can infer the injected DP noise and recover the noiseless result with high confidence.

Our solution. We address the above challenges with a generic framework for evaluating polynomials with distributed DP, where the participants utilize a general-purpose MPC protocol BGW [8] to perturb polynomial functions with integer-valued DP noises. We outline the idea of our solution next.

First, to avoid the aforementioned precision issue, both the function evaluation and noise injection are performed over integers. Regarding function evaluation, each party first pre-processes his/her local data to discrete integers. In this way, the computation is performed over the discrete integer domain from the very beginning, avoiding discrepancies between analysis and practical implementations. However, pre-processing the original data to integers could increase the sensitivity of the analysis function as the input domain expands. For a value of 0.01 may be rounded to 1, leading a $100x$ sensitivity increase and significantly higher amount of random noise to satisfy DP. We note that in general, such error in the sensitivity computation could be intractable, if not handled carefully. For example, 0.01 may also be rounded to 0, producing $0 \times 1000 = 0$, which could lead to either a sensitivity increase or decrease, depending on the leading sign. Note that although the above problem is also present for distributed DP for FL with horizontally partitioned databases (i.e., HFL), existing solutions for such a setting (e.g., [3, 6, 43]) do not apply to our problem, as they are mostly limited to estimating the linear sum of private data items where each item is possessed by a single party, who, in turn, can control the sensitivity of the rounded data item through clipping. On the contrary, as we need to evaluate polynomials of the inputs in the VFL setting, allowing a single party to process the sensitive (intermediate) outcome lead to privacy violations in the first place. We explain this issue in more detail in Section 7.

We address this problem through local quantization. In particular, each party up-scales the original inputs by a large factor before rounding so as to ensure the aforementioned bad outcomes due to rounding have a negligible impact on the overall result utility and privacy guarantee. The intuition is that the impact due to rounding the scaled inputs is negligible compared with scaling itself. 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 approach 0 as we increase the scaling factor (i.e., increasing the quantization granularity). However, this intuitive idea does not perform well when computing a general multi-dimensional polynomial (where each dimension is a polynomial containing monomials of different degrees) since scaling the inputs would lead to different scaling

[1]The original framework of distributed DP [15] is to securely shuffle locally perturbed data to amplify privacy. Here we twist its original meaning to achieving central-DP like privacy-utility trade-off in *distributed settings* without assuming trusted parties.

amplitudes for the monomials of different degrees, leading to high overall sensitivity. We tackle this situation with a modification–we ensure that each monomial is scaled by (roughly) the same factor no matter 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 larger than 1. This multiplication is done as a pre-processing step on the constant coefficients of the polynomial to be evaluated and does not lead to additional DP costs, since the description of the polynomial is public. Overall, the above described local quantization allow us to obtain a tractable sensitivity analysis over the quantized data and the sensitivity overhead it introduced approaches 0 as we increase the quantization granularity.

Next, we come to DP noise generation. We want the overall DP noise follows the Skellam distribution (takes value on $\{0, \pm 1, \pm 2\}$), which is known to have comparable privacy-utility trade-off as the continuous Gaussian noise [3, 6]. To do that, we let each party to privately sample a "share" of Skellam noise on the local side. When the local noise generation and data quantization are done, the parties collaboratively use MPC to evaluate the function of interest on the quantized data and in the mean time, inject all the locally sampled noises to the outcome. Since Skellam is closed under summation (i.e., the summation of two independent Skellam is still a Skellam), the overall noise injected into the outcome still follows a Skellam distribution, whose variance is n times larger than the locally generated Skellam noise (n is the total number of parties). As the overall DP noise is the aggregation of n local noises contributed by different parties, no single party is able to break the privacy guarantees just because she/he knows the outcome of one local noise. Furthermore, the generation of DP noise is also immune to 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), is its main techniques are the local qunatization and the Skellam noises. We show that our SQM matches the asymptotic performance (i.e., privacy-utility trade-off) of the centralized setting without relying on any trusted party, for evaluating polynomials over a vertically partitioned databases. Overall, to preserve DP, the database owners only need to quantize their inputs and sample some integer-valued DP noises using existing libraries (e.g., see TensorFlow Privacy's implementation) on the local side, and then collaboratively run the MPC protocol as a black box, to evaluate the polynomial on the quantized data and inject the aggregate DP noise to the outcome. Here, differential privacy imposes little overhead over the secure computation in FL, which is needed in the first place to keep the local data unseen. Accordingly, the DP analysis of our SQM also does not rely on the implementation of the MPC protocol BGW that we are using as a black box.

In the following, Section 2 provides necessary background on DP and FL. Section 3 formally defines the problem of computing a polynomial function with distributed DP and VFL, clarifies the privacy requirements in the presence of an adversarial coordinator and/or FL participants, and explains why most existing VFL solutions fail to satisfy these requirements. Section 4 presents the proposed solution SQM. In Section 5, we apply SQM to PCA computation, and prove that it achieves optimal privacy-utility trade-offs; meanwhile, we also apply it to logistic regression, and show that its noise due to data partitioning can be arbitrarily small. Section 6 contains an extensive set of experiments, confirming the high result utilities of SQM. Finally, Section 8 concludes the paper with future directions.

2 Preliminaries

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

Polynomials and Monomials. A polynomial is a mathematical expression consisting of variables and coefficients, which involves only the operations of addition, subtraction, multiplication, and positive-integer powers of variables. An example polynomial on $\mathbf{x} = (x[1], x[2], x[3])$ is $f(\mathbf{x}) = (x[1])^3 + 1.5 \cdot x[2]x[3] + 2$. A monomial (also called 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 the monomial components it contains. In the above example, the degree of $f(\mathbf{x})$ is 3.

2.1 Differential Privacy

Differential Privacy (DP) [25] 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)-Differential Privacy [25]). *A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy (DP) if for any neighboring databases $\mathbf{X} \sim \mathbf{X}'$ from the input domain and any set of outputs O, it holds that*

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

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

Definition 2 (Rényi Differential Privacy [57]). *A randomized mechanism $\mathcal{M}$ satisfies (α, τ)-Rényi differential privacy (RDP) for some $\alpha \in (0, 1) \cup (1, \infty)$ if for any neighboring input databases $\mathbf{X} \sim \mathbf{X}'$ it holds that*

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

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

Given a function of interest F, a canonical approach to make it differentially private is to inject random noise (perturbation) into its output. The scale of the noise is calibrated to the sensitivity of the mechanism [25], denoted as $S(F)$.

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

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

Lemma 1 (Gaussian noise preserves RDP [57]). *For any d-dimensional function with bounded $\mathcal{L}_2$ sensitivity Δ_2, injecting Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_d)$ to its output achieves $(\alpha, \frac{\alpha\Delta_2^2}{2\sigma^2})$-RDP.*

RDP guarantees can be converted to (ϵ, δ)-DP ones [12, 57]; the composition of multiple RDP mechanisms also satisfies RDP [57]; and running an RDP mechanism on a random subset of the input amplifies the original guarantees [58]. Both DP frameworks are preserved under post-processing [25].

2.2 Secure Multiparty Computation

Secure multiparty computation (MPC) [80] 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., in [11, 59]).

In this work, we use the classic BGW protocol [8] as a black box MPC protocol for evaluating any polynomial function rather than a specific one. Interested readers are referred to [61] for more detailed background. BGW has mature and efficient implementations: as reported in [7], evaluating 90,000 gates (which handle the summation of 30,000 multiplications between bits) with BGW takes less than 2 seconds.

Combining DP with MPC. Although MPC ensures confidentiality of the inputs, the output of the computed function may still leak private information [40, 55]. 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, however, 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 [42, 56] as mentioned in Section 1. Addressing this issue requires sampling integer-valued noise to satisfy DP, explained next.

2.3 Integer-Valued DP Noise

The state-of-the-art for enforcing DP with integer-valued noise is the Skellam mechanism [3, 6, 70], which injects Skellam noise into an integer-valued function to preserve DP. Specifically, the Skellam noise is obtained as the difference between two independent Poisson variables of the same parameters. We write $\mathbf{Z} \sim Sk^d(\mu)$ if $\mathbf{Z}$ is obtained as $\mathbf{U} - \mathbf{V}$, where both $\mathbf{U}$ and $\mathbf{V}$ are independently sampled from $Pois^d(\mu)$, which is the ensemble of d independent $Pois(\mu)$.

Lemma 2 (Skellam noise preserves RDP [3, 6]). *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 leads to bounded Rényi divergence on neighboring inputs. For any integer $\alpha > 1$, we have*

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

Comparing Lemma 2 (ignoring the min term which is often dominated by the preceding one) with Lemma 1, we can see that Skellam noise almost preserves the same level of privacy as the continuous Gaussian when injecting the same amount of noise, which is also supported with empirical evidence in [6]. Besides, the aggregate of n independent Skellam noises sampled from $Sk(\frac{\mu}{n})$ follows the distribution of $Sk(\mu)$, since Poisson is closed under summation. We will utilize these properties to design our solutions.

3 Problem Formulation

3.1 Vertical FL for Polynomial Functions

We consider federated learning over a vertically partitioned database, namely, VFL. 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 $\mathbf{X}$ (namely, $\mathbf{X}[:, j]$). 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

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

for some function $f : \mathbb{R}^n \to \mathbb{R}^d$, where d is the dimensionality of the function's output. In this work, we consider the case when f is a polynomial function. For example, the covariance matrix of the input (from which she/he can compute the principal components), which 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.

3.2 Privacy Requirements

We consider the semi-honest threat model. Specifically, the adversary follows the specification of mechanism $\mathcal{M}$ while trying to infer the private inputs of other parties. Here, the adversary could be either the server or a client j ($j \in [n]$), and its only adversarial behavior is to conduct a computation based on the observation it has received and its own input. Meanwhile, we assume the existence of an error-free and secure channel between each pair of clients. These assumptions have been commonly adopted in the distributed DP literature [3, 43].

In what follows, we formalize the privacy requirements under Rényi DP; the classical (ϵ, δ)-DP can be obtained with a standard conversion [12] from RDP. We consider two levels of privacy requirements: server-observed DP that concerns the curious server's observation $\mathcal{M}_{\text{server}}(\mathbf{X})$ and client-observed DP which concerns the client's observation $\mathcal{M}_{\text{client}_j}(\mathbf{X})$. Server-observed RDP prevents the server from inferring records in $\mathbf{X}$.

Definition 3 (server-observed RDP). *A randomized mechanism $\mathcal{M}$ satisfies (α, τ_{server}) server-observed RDP if it holds that*

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

where "$\sim$" represents the neighboring relation of two databases (resp. matrices) that one can be obtained from the other by adding/removing one record (resp. row).

Client-observed RDP prevents a client from inferring other clients' local data partitions.

Definition 4 (client-observed RDP). *A randomized mechanism $\mathcal{M}$ satisfies (α, τ_{client}) client-observed RDP if for any client j, it holds that*

$$\sup_{\mathbf{X}\sim\mathbf{X}'} D_\alpha(\mathcal{M}_{client_j}(\mathbf{X}) \,\|\, \mathcal{M}_{client_j}(\mathbf{X}')) \leq \tau_{client}. \tag{6}$$

where "$\sim$" represents the neighboring relation of two databases (resp. matrices) that one can be obtained from the other by replacing one record (resp. row).

Next, we clarify the main difference between Definitions 3 and 4. In server-observed DP, we adopt the *unbounded* neighboring database definition [24], where a neighboring database $\mathbf{X}'$ is obtained from $\mathbf{X}$ by removing/adding one record. This definition also protects the information of m (the number of individual records in the input database) from the server, which is also used in previous works in horizontal FL [3, 6, 43]. On the other hand, in client-observed DP, we use the *bounded* DP definition [25], where the size of $\mathbf{X}$ (namely, m) is not considered private information. This is because 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), to which the server does not have access.

In addition, we emphasize that there is no general conversion rule between τ_{server} and τ_{client}, formalized as the following proposition.

Proposition 1. *A mechanism that satisfies server-observed DP can be non-private for a curious client. Similarly, a mechanism that satisfies client-observed DP can be non-private for a curious server.*

We sketch the proof by construction. For the first statement, we let a single client (say 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 server-observed DP, but not client-observed DP with respect to 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 client-observed DP for all clients, but not-server-observed DP. Hence, it is necessary to consider both Definitions 3 and 4, if a client wants to protect her/his data from the server and other clients simultaneously.

Remark. Note that existing solutions [62, 78] on enforcing DP in VFL simply let a trusted client (or a trusted third party) inject random noise into the sensitive (i.e., non-private) result. Clearly, such methods rely on the existence of a trusted party, and do not satisfy our privacy definitions.

3.3 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 [29, 43]. 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 $O(1)$. Considering there are m such outer products in total, the overall error is $O(m)$. This is far higher than the centralized DP setting in which noise is calibrated with the sensitivity of the covariance matrix, which is constant.

4 Proposed Solution

In this section, we present the general Skellam Quantization Mechanism for VFL with distributed DP. The main idea is to let each client generate a small DP noise locally, and then utilize MPC to combine the local noises, amplifying the privacy guarantee and achieving comparable privacy-utility trade-off as in the centralized setting, without assuming trusted parties. The detailed proofs of our claims are in the full version [5].

4.1 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 $F(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$ with the privacy constraints specified as in Definitions 3 and 4. Without loss of generality we have assumed 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 the privacy guarantee. 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 next.

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

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 $\sum_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$.
```

Algorithm 2: Data Pre-processing/Quantization

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

The processed data portions of all clients constitute database $\hat{\mathbf{X}}$, as follows.

$$\hat{\mathbf{X}} = (\hat{\mathbf{X}}[:, 1], \ldots, \hat{\mathbf{X}}[:, n]). \tag{7}$$

Skellam 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

$$\hat{y}(\mathbf{X}) = \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \hat{f}(\hat{\mathbf{x}}) + \sum_{j=1}^{n} \mathbf{Z}_j. \tag{8}$$

The result $\hat{y}(\mathbf{X})$ is then sent to the server (Line 6).

Finally, the server computes $\tilde{y} = \frac{1}{\gamma^\lambda}\hat{y}(\mathbf{X})$, obtaining the estimation 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 noises. From the server's perspective, $\hat{y}$ is perturbed by $Z \sim Sk(\mu)$, 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 by $Z^* \sim Sk\left(\frac{n-1}{n}\mu\right)$ (since she knows the outcome of her local noise) which is close to $Sk(\mu)$ when n is large. Hence, our privacy requirements are satisfied with the noise aggregate injected.

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 compared with the continuous Gaussian mechanism (see Lemma 1). 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 5. We first analyze the error due to rounding and neglect the requirement of privacy for the moment (i.e., setting μ to 0).

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

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

Lemma 3 implies the following error guarantee on Algorithm 1.

Lemma 4. *Consider input database $\mathbf{X}$ such that each $\mathbf{x} \in \mathbf{X}$ satisfies $\|\mathbf{x}\|_2 \leq c$, one-dimensional monomial f with degree $\lambda \geq 1$, scaling parameter $\gamma \geq 100\lambda$, 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.

Due to the triangle inequality, Lemma 3 also implies that the sensitivity overhead of evaluating f on the quantized data becomes negligible when γ is large, giving the following privacy guarantees.

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

$$\tau_{server} = \frac{\alpha\Delta^2}{4\mu} + \min\left(\frac{(2\alpha-1)\Delta^2 + 6\Delta}{16\mu^2}, \frac{3\Delta}{4\mu}\right), \tag{9}$$

and (α, τ_{client}) client-observed-RDP with

$$\tau_{client} = \frac{\alpha n\Delta^2}{(n-1)\mu} + \min\left(\frac{(2\alpha-1)n^2\Delta^2 + 3n^2\Delta}{4(n-1)^2\mu^2}, \frac{3n\Delta}{2(n-1)\mu}\right). \tag{10}$$

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

We sketch the proof for Eq. (9) that corresponds to server-observed privacy. The proof for Eq. (10) that corresponds to client-observed privacy 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/remove 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 proof then follows from applying Lemma 2 with sensitivity set to the above difference.

When computing the privacy guarantees, the min term in either Eq. (9) or (10) is dominated by the first term outside the min function when μ is large (i.e., under strong privacy constraints). Hence, to achieve (α, τ)-RDP, regarding the adversarial server, it suffices to set μ to roughly $\frac{\alpha}{2} \cdot \frac{\gamma^{2\lambda} \max |f(\mathbf{x})|^2 + O(\gamma^{2\lambda-1})}{2\tau}$.

In other words, the overall noise contributed by all clients for perturbing the function F computed over the quantized data is roughly γ^λ times the noise needed to perturb function F computed over the original data. In addition, we also recall from Lemma 4 that the approximation error is also $o(\gamma^\lambda)$. Both overheads in privacy and approximation error will become negligible for a large γ, considering that the server will down-scale the perturbed outcome by γ^λ. In other words, after post-processing by the server, the overall noise variance is roughly

$$\frac{\alpha}{2} \cdot \frac{\max |f(\mathbf{x})|^2 + o(1)}{2\tau},$$

which is comparable to the Gaussian mechanism (see Lemma 1). Next, we show how to generalize SQM to the more general case where the function of interest f is a multi-dimensional polynomial.

4.2 SQM for Multi-dimensional Polynomials

We consider the general case when f is a d-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

$$f_t(\mathbf{x}) = \sum_{l=1}^{v_t} a_t[l] \cdot \Pi_{j=1}^{n} x[j]^{B_t[l,j]}, \tag{11}$$

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 (namely, 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 (namely, $\sum_{j=1}^{n} B_t[l, j]$ 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 downscales 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.

Idea. The idea is to ensure that each monomial in each dimension is scaled by the same factor no matter its original degree, so that we can utilize our established analysis (Lemmas 4 and 3) to quantify the overall sensitivity for computing the d-dimensional polynomial as a whole. Our solution goes as follows. We first identify the monomial of the largest degree among all dimensions. Without loss of generality, we denote this degree as λ, and refer to it as the degree of the d-dimensional polynomial f. For each private input $x[i, j]$, it is scaled by some factor $\gamma > 1$ and then randomly rounded to its nearest integers, using Algorithm 2. This operation would result in different scaling factors for monomials of different degrees.

To compensate for such differences, next, 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 integers, 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, the server would need to post-process the result by a multiplication of $\frac{1}{\gamma^{\lambda+1}}$.

With slight modifications on Lemmas 4 and 3, we can show that the approximation error for each monomial term is still in $o(1)$ when γ is large enough, which means that 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 are in Lines 1-3 of Algorithm 3, where the *server* process the coefficients using Algorithm 2 (the processed coefficients are publicly known to all clients) with different scaling parameters for monomials of different degrees, and in Line 11, where the server down-scales the outcome by $\frac{1}{\gamma^{\lambda+1}}$. Next, we provide the privacy analysis for Algorithm 3.

Lemma 6. *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 with*

$$\tau_{server} = \frac{\alpha \Delta_2^2}{4\mu} + \min\left(\frac{(2\alpha - 1)\Delta_2^2 + 6\Delta_1}{16\mu^2}, \frac{3\Delta_1}{4\mu}\right), \tag{12}$$

and (α, τ_{client}) client-observed-RDP with

$$\tau_{client} = \frac{\alpha n \Delta_2^2}{(n-1)\mu} + \min\left(\frac{(2\alpha - 1)n^2\Delta_2^2 + 3n^2\Delta_1}{4(n-1)^2\mu^2}, \frac{3n\Delta_1}{2(n-1)\mu}\right), \tag{13}$$

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 6 and 5. The sensitivity overhead for evaluating a d-dimensional polynomial

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 $\sum_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$.
```

function, and each dimension of it is a polynomial containing v_t monomials. This means that, than the overhead in terms of both error and sensitivity (recall Lemmas 4 and 3) 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, we conclude that for evaluating general multi-dimensional polynomials, SQM can achieve comparable utility-privacy trade-offs as the continuous Gaussian in the centralized setting.

5 Applications

In this section, we instantiate our proposed SQM on the tasks of principal component analysis and logistic regression. The detailed proofs of our claims are in the full version [5].

5.1 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

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

Now we turn to the sensitivity of computing the covariance matrix $\tilde{\mathbf{C}}$ on the processed inputs corresponding to $\mathbf{X}'$ and $\mathbf{X}$. Since for every record we have that $\|\mathbf{X}[i, :]\|_2 \le c$, the processed version (output by Algorithm 2) satisfies $\|\hat{\mathbf{X}}[i, :]\|_2 \le \sqrt{\gamma^2c^2 + n}$. The following holds.

$$\|\hat{\mathbf{X}}^T\hat{\mathbf{X}} - \hat{\mathbf{X}}'^T\hat{\mathbf{X}}'\|_F \le \sum_j \left(\hat{\mathbf{X}}[m, j]\right)^2 \le \gamma^2c^2 + n. \tag{14}$$

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. (14) with Lemma 2, we have the DP guarantees.

Lemma 7. *With scaling parameter γ and noise parameter μ, computing the noisy covariance matrix over using Algorithm 3 satisfies (α, τ_{server}) server-observed RDP with*

$$\tau_{server} = \frac{\alpha\Delta_2^2}{4\mu} + \min\left(\frac{(2\alpha-1)\Delta_2^2 + 6\Delta_1}{16\mu^2}, \frac{3\Delta_1}{4\mu}\right), \tag{15}$$

and (α, τ_{client}) client-observed RDP with

$$\tau_{client} = \frac{\alpha n\Delta_2^2}{(n-1)\mu} + \min\left(\frac{(2\alpha-1)n^2\Delta_2^2 + 3n^2\Delta_1}{4(n-1)^2\mu^2}, \frac{3n\Delta_1}{2(n-1)\mu}\right), \tag{16}$$

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 [12]. Finally, we present the utility result for the obtained principal components. For better comparison with the strong central DP baseline [28], we formalize it under (ϵ, δ)-DP.

Lemma 8. *Let $\mathbf{V}$ be the rank-k subspace of the original matrix $\mathbf{X}$ and let $\tilde{\mathbf{X}}$ be the principal rank-k subspace of matrix $\tilde{\mathbf{C}}$ 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 that the obtained subspace $\tilde{\mathbf{C}}$ satisfies*

$$\|\mathbf{X}\tilde{\mathbf{V}}\|_F^2 \ge \|\mathbf{X}\mathbf{V}\|_F^2 - O(k\sqrt{n}\sqrt{\mu_{\epsilon,\delta}}), \tag{17}$$

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

As a comparison, the optimal error rate in the centralized setting [28] is also $O(k\sqrt{n}\sqrt{\log(1/\delta)}/\epsilon)$. For (α, τ)-RDP, we substitute $\sqrt{\mu_{\epsilon,\delta}}$ with $\sqrt{\mu} \approx \sqrt{\alpha/\tau} \cdot \Delta_2/2$ (recall Lemma 7) and the error is still in $O(k\sqrt{n})$, indicating the optimality of our solution.

5.2 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 [82]. 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 [46] 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

$$g((\mathbf{x}, y)) = \sigma(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x} - y \cdot \mathbf{x}. \tag{18}$$

Reduction to polynomial evaluation. Due to the sigmoid function, the computation of Eq. (18) cannot be written as a polynomial of $(\mathbf{x}, y)$. Here we follow [82] to approximate the sigmoid function as polynomials using Taylor series: $\sigma(u) = \sum_{h=0}^{\infty} \left(\frac{\sigma^{(h)}(u)|_{u=0}}{h!} \cdot u^h\right)$, where $\sigma^{(h)}(u)|_{u=0}$ represents the h-th order derivative of $\sigma(u)$ evaluated at $u = 0$. Here, we consider $H = 1$ (we will see soon that it is enough to obtain high utility). Then we can write $\sigma(u) \approx \frac{1}{2} + \frac{1}{4} \cdot u$, and approximate the gradient with $g((\mathbf{x}, y)) \approx \frac{1}{2} \cdot \mathbf{x} + \frac{1}{4} \cdot (\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x} - y \cdot \mathbf{x}$, which is a polynomial of the input $(\mathbf{x}, y)$.

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

$$f(\mathbf{w}, (\mathbf{x}, y)) = \frac{1}{2} \cdot \mathbf{x} + \langle \frac{\mathbf{w}}{4}, \mathbf{x}\rangle \cdot \mathbf{x} - y \cdot \mathbf{x}. \tag{19}$$

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

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

Analysis. We provide privacy guarantees for the above procedure.

Lemma 9. *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 with*

$$\tau_{server} = \frac{R}{\alpha - 1} \cdot \log\Bigg((1-q)^{\alpha-1}(\alpha q - q + 1) + \sum_{l=2}^{\alpha} \binom{\alpha}{l} (1-q)^{\alpha-l} q^l e^{(l-1)\tau_l}\Bigg), \tag{20}$$

and satisfies (α, τ_{client}) client-observed *RDP with*

$$\tau_{client} = \frac{\alpha R n \Delta_2^2}{(n-1)\mu} + R \min\left(\frac{(2\alpha-1)n^2\Delta_2^2 + 3n^2\Delta_1}{4(n-1)^2\mu^2}, \frac{3n\Delta_1}{2(n-1)\mu}\right), \tag{21}$$

where

$$\Delta_2 = \sqrt{\left(\frac{3}{4}\gamma^3\right)^2 + 9\gamma^5 d + 36\gamma^4}, \textit{ and } \Delta_1 = \min(\Delta_2^2, \sqrt{d}\Delta_2). \tag{22}$$

For each $l = 2, \ldots, \alpha$, we have

$$\tau_l = \frac{l\Delta_2^2}{4\mu} + \min\left(\frac{(2l-1)\Delta_2^2 + 6\Delta_1}{16\mu^2}, \frac{3\Delta_1}{4\mu}\right). \tag{23}$$

In Eq. (22), $\frac{3}{4}$ 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. (19)). With the scaling parameter γ, this sensitivity becomes $\frac{3}{4}\gamma^3 + o(\gamma^3)$, where the small-oh overhead corresponds to the $\sqrt{36\gamma^4 + 9\gamma^5 d}$ shown in Eq. (22). 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 sensitivities in Eq. (22) to Lemma 2 and then applying the results on privacy amplification by subsampling and composition theorems [57, 58, 83] give us the privacy guarantee with repsect to a curious server (namely, Eq. (20)). Here the subsampling is performed on the record level, which is aligned with our privacy definition in Definition 3. A more advanced analysis on the user level privacy (see [32] 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.

On discretization. In our SQM and its instantiations, we have used an explicit discretization procedure (Algorithm 2) for preprocessing 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 (e.g., see [13]), as we have mentioned in Section 1. We would also like to emphasize that unlike the gradient sum estimation in horizontal FL [3, 6, 43] where each client can independently clip their local gradient to enforce a sensitivity upper bound, in the setting of vertical FL, we cannot trust a party to perform such clipping on an individual gradient, which contains private information regarding all clients' data. Hence, we have enforced an explicit local pre-processing for the data to obtain tractable sensitivity analysis.

Error-efficiency trade-off. With privacy fixed, the parameter γ effectively controls the trade-off between error/utility and efficiency. With a smaller choice of γ, the relative sensitivity overhead is large which requires larger DP noises to preserve privacy, whereas the overall costs for computing the target function over coarse-grained inputs using MPC is cheaper. On the other hand, when γ is larger, the relative sensitivity overhead is smaller, whereas the overall costs for computing the target function over fine-grained inputs using MPC is more expensive.

Overhead due to DP. In SQM, the quantization procedures for the input data and coefficients and the generation of Skellam noises can be done in parallel efficiently. These two procedures, together with the operation for combining the local Skellam noises injection (performing n additions in BGW), constitute the overhead for preserving differential privacy on top of doing MPC, which protects the clients' data from being observed during FL. Considering that the cost for n additions might be much smaller than the cost for computing the covariance matrix of $m \times m$ for PCA, or approximating the gradients for a batch of samples in logisitic regression, the overhead due to DP is not significant, compared with MPC itself, which may be required in FL in the first place to prevent the data from being observed.

Outsourcing the computation. Besides repeatedly participating in MPC, which requires synchronicity and incurs high computation and communication costs, the clients can also opt to outsource the computation [65]–distribute their processed data and locally-generated DP noises to non-colluding servers (e.g., to 2 servers) using secret sharing [64] and let the servers perform the computation (e.g., using the two-party MPC protocol ABY [20]). Such an outsourced computation was also adopted in the recent federated analytics (FA) [39, 67]. We note that the original framework of anonymous aggregation [67] only considers linear functions, and our solution for polynomial evaluation can be seen as a generalization.

6 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 [3, 6, 43], we measure the performance in terms of privacy-utility trade-offs for the fitted models, and use a single machine (with A5000 GPU, 64x AMD EPYC 16-core CPU, and 500GB memory) to simulate the distributed setting where each party is assumed to have a secure and noiseless channel for communication.

6.1 Principal Component Analysis

We compare our solution SQM for PCA with the centralized-DP solution [28] and the distributed baseline described in Section 3.3 (shown as "Centralized" and "VFL-Baseline" in the figures, respectively). In particular, the strong performance of the centralized algorithm [28] is regarded as the goal (the upper limit) for our mechanism to achieve. We test on the KDDCUP dataset [68] with $m = 195666$ and $n = 117$, as well as on two high dimensional datasets, CiteSeer [38] with $m = 2110$ and $n = 3703$ and Gene [33] 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 fix δ to 10^{-5} and vary ϵ. For our mechanism, we also vary the scaling parameter γ. For the high-dimensional datasets, CiteSeer and Gene, we choose a relatively larger γ to maintain the signal-to-noise ratio.

We report the average utility over 20 independent runs in Figure 1, from which it is clear that our solution consistently outperforms the local DP baseline. The performance of SQM is close to the centralized DP competitor under various parameter settings, especially when γ is large. This confirms the optimality of SQM. In particular, the performance of SQM increases with the scaling parameter 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).

6.2 Logistic Regression

We evaluate SQM for logistic regression on 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 [21]. 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.

We compare the performance of SQM against the baseline VFL solution with local DP, which first perturbs the dataset $\mathbf{X}$ on the client side and then trains the LR model on the perturbed dataset (denoted as "VFL-Baseline" in the figure). In addition, we use the popular centralized-DP mechanism, DPSGD [2], to establish a performance upper limit (denoted as "Centralized" in the figure). We omit other centralized-DP mechanisms since they perform similarly to DPSGD, and our goal is not to evaluate DP mechanisms for the centralized setting. For SQM, we also vary the scaling parameter γ. 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 local DP baseline where the DP noise is directly injected into the training data, we train the model until convergence. Similar to PCA, the privacy level is calculated based on server-observed privacy. We report the average test accuracy over 20 independent runs in Figure 2.

From Figure 2, we can see that SQM significantly outperforms the baseline solution under all parameter settings for all datasets (the baseline's accuracy is constantly below 60%). In addition, even when ϵ is small, our solution can still maintain a decent utility. In addition, given a sufficient amount of privacy budget, SQM achieves comparable performance as the centralized-DP approach. Specifically, the accuracy drop of our solution with $\gamma = 2^{13}$ compared with the centralized solution is negligible for $\gamma \geq 1$. When $\gamma = 2^{10}$, our solution still maintains decent utilities across different privacy parameter settings.

Effect of scaling parameter. We study how γ influences the sensitivity overhead and hence the scale of DP noise for the analysis

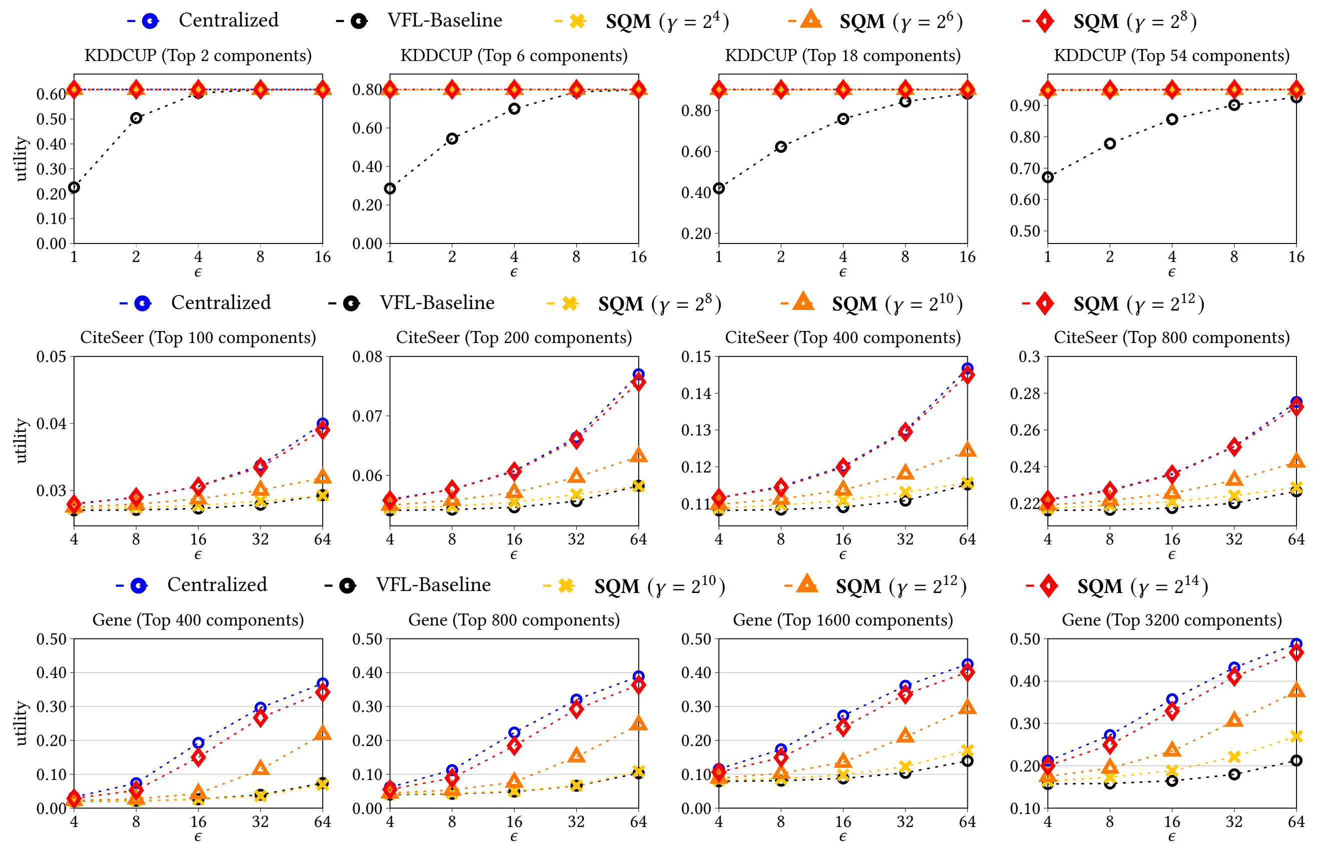

Figure 1: Performance for PCA on multiple datasets for varying numbers of top components and DP constraints.

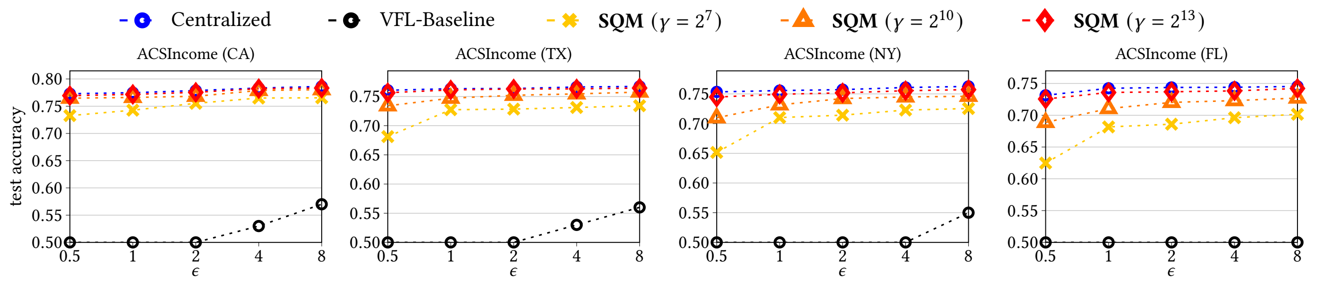

Figure 2: Performance for logistic regression on multiple datasets under different DP constraints. The accuracy of "VFL-Basline" is constantly below 0.6; and hence, is truncated from the figures.

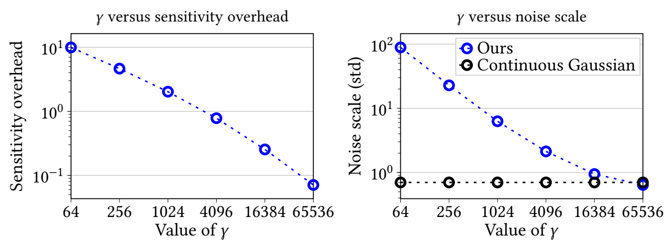

Figure 3: Effect of parameter γ on the sensitivity overhead and the noise scale for SQM on logistic regression with target privacy parameters fixed to $\epsilon = 1$ and $\delta = 10^{-5}$.

in Lemma 9. We compute the $\mathcal{L}_2$ sensitivity overhead for different choice of γ ranging from $\{64, 256, 1024, 4096, 16394, 65536\}$. Recall that the original sensitivity for the gradient sum is $\frac{3}{4}$ and Eq. (22), then we can compute this overhead as $\sqrt{(\frac{3}{4})^2 + \frac{9d}{\gamma} + \frac{36}{\gamma^2}} - \frac{3}{4}$, with $d = 800$. Here, we have consider the normalized gradient sum that is post-processed by the server for better reference.

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

From Figure 3, it is clear that as we increase the scaling paraemter γ, both the sensitivity overhead and noise overhead compared with the centralized Gaussian quickly decreases to 0 (note that the y-axis is in log scale). This result explains why the performance of our SQM on LR approaches the centralized competitor in Figure 2, as we increase γ. Similar explanations also apply to our experiments on PCA, and here we have omitted them due to space constraints.

7 Related Work

Federated learning over vertically partitioned databases has been extensively studied in the database community, e.g., in [14, 19, 34–37, 47, 48, 50, 72, 76, 77]. Various algorithms, frameworks, and systems have been proposed. [36, 47, 76] train tree-based models under VFL, where models are trained using on their partitioned data without explicitly sharing it; Fu et al. [37] propose a communication-efficient framework for selecting features; Li et al. [48] study K-means clustering with local DP constraints; [34, 47, 77] propose and implement efficient and practical VFL systems. We refer interested readers to [49] for a comprehensive survey of the literature.

An orthogonal work [50] exploits the privacy risk in VFL, and shows that without proper privacy protection, an adversary could easily recover some features of the underlying training records, by querying the trained model through prediction APIs only. This work has highlighted the significance that we need to incorporate record-level privacy protection techniques into VFL.

Preserving data privacy for analyzing databases jointly owned by several clients predates the framework of DP [25]. The seminal work [69] analyzes association rules of attributes for the two-party scenario, and the subsequent work [26] extends the problem to more than two clients under the privacy notion of *bounded confidence increase*, an extension of p_0–to–p_1 privacy breach [31]. Due to the differences in privacy definitions and problem formalizations, our work is not directly comparable with theirs.

Most recent works that try to enforce DP for VFL adopt different privacy requirements and threat models than ours. Hence, they are not directly comparable with ours. Specifically, in works such as [62, 78], 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 to the outcome produced by MPC. This approach leads to privacy violations since the client who performs the noise injection could infer information about the clients' inputs from the exact result before noise injection, as exploited in recent works [40, 55].

Wu et al. [76] let the clients use shared randomness to jointly sample a Laplace noise using MPC. After that, the noise is used to perturb the sensitive results. Note that such a mechanism is also non-private in the presence of a curious client, who, by observing the Laplace noise, can infer information about the database. In addition, the finite-precision representations of continuous DP noises could also lead to privacy violations, as discovered in [56].

The idea of utilizing cryptographic protocols to enhance DP originates from horizontal FL [15, 29], which, in turn, inspires combinations of cryptography with DP (e.g., see [16, 41]) and drives the development of integer-valued DP mechanisms (e.g., see [3, 6, 12]). The most relevant works with ours are [3, 6], where the untrusted server wants to estimate the sum of private data items possessed by multiple clients. In their approach, the clients randomly round the up-scaled private data vectors and then perturb them with local Skellam noises. After that, the clients use SecAgg [11] to amplify the DP guarantees by aggregating the perturbed data vectors, and share the aggregate outcome with the server, who then down-scales the outcome to obtain an unbiased estimate.

However, this approach does not apply to our problem of evaluating polynomial functions over vertically partitioned databases. The key problem is *non-linearity*. First, we cannot convert the function of interest (which is a polynomial of inputs) to a linear sum of individual components, each of which can be computed and perturbed by a single client without accessing other clients' data. In addition, for polynomials of different degrees, scaling the inputs by a common factor will lead to different magnitudes of amplification for monomials of different degrees, making a tight sensitivity analysis intractable. Our mechanism SQM resolves these issues, achieving comparable privacy-utility trade-offs as the central DP solution without assuming any trusted party.

Recent work [75] proposes an MPC algorithm for untrusted clients to collectively sample a discrete Gaussian noise [12], a substitute of Skellam for performing integer-valued DP noise injection. Combining it with our SQM could further enhance the client-observed level of privacy, as each client will have no information about the noise outcome. However, there are several caveats. First, generating DP noises using MPC could be prone to side-channel attacks, as the outcome of the random variable is strongly correlated to the overall runtime of the MPC protocol, as exploited in [42]. Another issue is the computational overhead. In our solutions, the clients can generate local noises efficiently in an offline manner while avoiding side-channel attacks. Furthermore, the current implementation [75] focuses on the traditional (ϵ, δ)-DP and also incurs a non-negligible overhead on the privacy parameter δ, which could lead to privacy overheads in compositional mechanisms where the private data is repeatedly queried.

8 Conclusion

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

Regarding future work directions, we plan to further enhance the privacy protection of our mechanism regarding adversarial clients when performing record-level subsampling, by preventing the clients from knowing exactly which records are sampled. We also plan to extend our mechanism to other application scenarios, such as analyzing distributed social networks–when finding the largest eigenvalues and the corresponding eigenvectors of a private data matrix, the power iteration method that performs *matrix multiplications* without explicitly solving for the eigenvalues and eigenvectors is more efficient for large matrices.

References

[1] 2008. IEEE Standard for Floating-Point Arithmetic - Redline. *IEEE Std 754-2008 (Revision of IEEE Std 754-1985) - Redline* (2008), 1–82.
[2] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. 308–318.
[3] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The Skellam Mechanism for Differentially Private Federated Learning. In *NeurIPS*. 5052–5064.
[4] B. Balle and Yu-Xiang Wang. 2018. Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising. In *ICML*.
[5] Ergute Bao, Fei Wei, Xiaokui Xiao, Yin Yang, Tianyu Pang, and Chao Du. 2024. *Secure and Effective Federated Learning with Distributed Differential Privacy for Vertically Partitioned Data (technical report)*. Retrieved May 1, 2024 from https://github.com/erguteb/VLDB_2025_submission/blob/main/technical_report.pdf
[6] Ergute Bao, Yizheng Zhu, Xiaokui Xiao, Yin Yang, Beng Chin Ooi, Benjamin Hong Meng Tan, and Khin Mi Mi Aung. 2022. Skellam Mixture Mechanism: a Novel Approach to Federated Learning with Differential Privacy. *PVLDB* (2022), 2348–2360.
[7] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. 2016. Optimizing Semi-Honest Secure Multiparty Computation for the Internet. Cryptology ePrint Archive.
[8] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *STOC*. 1–10.
[9] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. 2017. Prochlo: Strong Privacy for Analytics in the Crowd. In *SOSP*. 441–459.
[10] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. 2005. Practical privacy: the SuLQ framework. In *PODS*. 128–138.
[11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. 1175–1191.
[12] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. 2020. The Discrete Gaussian for Differential Privacy. In *NeurIPS*.
[13] Sílvia Casacuberta, Michael Shoemate, Salil Vadhan, and Connor Wagaman. 2022. Widespread Underestimation of Sensitivity in Differentially Private Libraries and How to Fix It. In *CCS*. 471–484.
[14] Yihang Cheng, Lan Zhang, JunYang Wang, Xiaokai Chu, Huang Dongbo, and Lan Xu. 2024. FedMix: Boosting with Data Mixture for Vertical Federated Learning. In *ICDE*. To appear.
[15] Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed Differential Privacy via Shuffling. In *EUROCRYPT*. 375–403.
[16] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. 2020. Crypt?: Crypto-Assisted Differential Privacy on Untrusted Servers. In *SIGMOD*. 603–619.
[17] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal Release Under Local Differential Privacy. In *SIGMOD*. 131–146.
[18] Guozheng Dai, Zhonggen Su, and Hanchao Wang. 2023. Tail Bounds on the Spectral Norm of Sub-Exponential Random Matrices. In *Random Matrices: Theory and Applications*, Vol. 13. 2350013.
[19] Wenrui Dai, Xiaoqian Jiang, Luca Bonomi, Yong Li, Hongkai Xiong, and Lucila Ohno-Machado. 2022. VERTICOX: Vertically Distributed Cox Proportional Hazards Model Using the Alternating Direction Method of Multipliers. *TKDE* 34, 2 (2022), 996–1010.
[20] Daniel Demmler, T. Schneider, and Michael Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*.
[21] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. In *NeurIPS*. 6478–6490.
[22] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *PODS*. 202–210.
[23] Wei Dong, Dajun Sun, and Ke Yi. 2023. Better than Composition: How to Answer Multiple Relational Queries under Differential Privacy. *PACMMOD* (2023), 155–163.
[24] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*. 1–12.
[25] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.
[26] Cynthia Dwork and Kobbi Nissim. 2004. Privacy-Preserving Datamining on Vertically Partitioned Databases. In *CRYPTO*. 528–544.
[27] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *FTTCS* 9, 3-4 (2014), 211–407.
[28] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. 2014. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *STOC*. 11–20.
[29] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*. 2468–2479.
[30] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*. 1054–1067.
[31] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. 2003. Limiting privacy breaches in privacy preserving data mining. In *PODS*. 211–222.
[32] Juanru Fang and Ke Yi. 2024. Privacy Amplification by Sampling under User-level Differential Privacy. In *PACMMOD*.
[33] Samuele Fiorini. 2016. Gene expression cancer RNA-Seq. UCI Machine Learning Repository.
[34] Fangcheng Fu, Xupeng Miao, Jiawei Jiang, Huanran Xue, and Bin Cui. 2022. Towards communication-efficient vertical federated learning training via cache-enabled local updates. *PVLDB* (2022), 2111–2120.
[35] Fangcheng Fu, Yingxia Shao, Lele Yu, Jiawei Jiang, Huanran Xue, Yangyu Tao, and Bin Cui. 2021. VF2Boost: Very Fast Vertical Federated Gradient Boosting for Cross-Enterprise Learning. In *SIGMOD*. 563–576.
[36] Fangcheng Fu, Huanran Xue, Yong Cheng, Yangyu Tao, and Bin Cui. 2022. BlindFL: Vertical Federated Machine Learning without Peeking into Your Data. In *SIGMOD*. 1316–1330.
[37] Rui Fu, Yuncheng Wu, Quanqing Xu, and Meihui Zhang. 2023. FEAST: A Communication-efficient Federated Feature Selection Framework for Relational Data. *PACMMOD* (2023).
[38] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. CiteSeer: An Automatic Citation Indexing System. In *JCDL*. New York, NY, USA, 89–98.
[39] Google Research. 2020. Federated Analytics: Collaborative Data Science Without Data Collection. (2020). https://research.google/pubs/pub48576/
[40] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *CCS*. 603–618.
[41] Ziyue Huang, Yuan Qiu, Ke Yi, and Graham Cormode. 2022. Frequency estimation under multiparty differential privacy: one-shot and streaming. *PVLDB* (2022), 2058–2070.
[42] Jiankai Jin, Eleanor McMurtry, Benjamin I. P. Rubinstein, and Olga Ohrimenko. 2021. Are We There Yet? Timing and Floating-Point Attacks on Differential Privacy Systems. In *S&P*. 473–488.
[43] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. In *ICML*. 5201–5212.
[44] Hannah Keller, Helen Möllering, Thomas Schneider, Oleksandr Tkachenko, and Liang Zhao. 2024. Secure Noise Sampling for DP in MPC with Finite Precision. In *ARES*.
[45] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *CCS*. 1575–1590.
[46] J. Kiefer and J. Wolfowitz. 1952. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics* 23, 3 (1952), 462 – 466.
[47] Xiaochen Li, Yuke Hu, Weiran Liu, Hanwen Feng, Li Peng, Yuan Hong, Kui Ren, and Zhan Qin. 2022. OpBoost: a vertical federated tree boosting framework based on order-preserving desensitization. *PVLDB* (2022), 202–215.
[48] Zitao Li, Tianhao Wang, and Ninghui Li. 2023. Differentially Private Vertical Federated Clustering. *PVLDB* (2023), 1277–1290.
[49] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. 2024. Vertical Federated Learning: Concepts, Advances, and Challenges. In *TKDE*. 1–20.
[50] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature Inference Attack on Model Predictions in Vertical Federated Learning. In *ICDE*. 181–192.
[51] Samuel Maddock, Graham Cormode, Tianhao Wang, Carsten Maple, and Somesh Jha. 2022. Federated Boosted Decision Trees with Differential Privacy. In *CCS*. 2249–2263.
[52] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.
[53] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.
[54] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*. 19–30.
[55] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *S&P*. 691–706.
[56] Ilya Mironov. 2012. On Significance of the Least Significant Bits for Differential Privacy. In *CCS*. 650–661.
[57] Ilya Mironov. 2017. Rényi Differential Privacy. In *CSF*. 263–275.
[58] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *CoRR* abs/1908.10530 (2019).
[59] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *S&P*. 19–38.
[60] Kobbi Nissim. 2021. Privacy: From Database Reconstruction to Legal Theorems. In *PODS*. 33–41.

[61] Goldreich Oded. 2009. *Foundations of Cryptography*. USA.

[62] Thilina Ranbaduge and Ming Ding. 2022. Differentially Private Vertical Federated Learning. *CoRR* abs/2211.06782 (2022).

[63] Deevashwer Rathee, Anwesh Bhattacharya, Rahul Sharma, Divya Gupta, Nishanth Chandran, and Aseem Rastogi. 2022. SecFloat: Accurate Floating-Point meets Secure 2-Party Computation. In *S&P*. 576–595.

[64] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613.

[65] Zihao Shan, Kui Ren, Marina Blanton, and Cong Wang. 2018. Practical Secure Computation Outsourcing: A Survey. *ACM Comput. Surv.* 51, 2 (2018).

[66] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *CCS*. 1310–1321.

[67] Kunal Talwar, Shan Wang, Audra McMillan, Vojta Jina, Vitaly Feldman, Bailey Basile, Aine Cahill, Yi Sheng Chan, Mike Chatzidakis, Junye Chen, et al. 2023. Samplable Anonymous Aggregation for Private Federated Data Analysis. *arXiv preprint arXiv:2307.15017* (2023).

[68] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *CISDA*. 1–6.

[69] Jaideep Vaidya and Chris Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In *KDD*. 639–644.

[70] Filipp Valovich and Francesco Aldà. 2017. Computational Differential Privacy from Lattice-Based Cryptography. In *NuTMiC*, Vol. 10737. 121–141.

[71] Roman Vershynin. 2011. Spectral norm of products of random and deterministic matrices. *Probability Theory and Related Fields* 150 (2011), 471–509.

[72] Junhao Wang, Lan Zhang, Anran Li, Xuanke You, and Haoran Cheng. 2022. Efficient Participant Contribution Evaluation for Horizontal and Vertical Federated Learning. In *ICDE*. 911–923.

[73] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *ICDE*. 638–649.

[74] Tianhao Wang, Bolin Ding, Min Xu, Zhicong Huang, Cheng Hong, Jingren Zhou, Ninghui Li, and Somesh Jha. 2020. Improving Utility and Security of the Shuffler-based Differential Privacy. *PVLDB* (2020), 3545 – 3558.

[75] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. 2023. Securely Sampling Discrete Gaussian Noise for Multi-Party Differential Privacy. In *CCS*. 2262–2276.

[76] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy Preserving Vertical Federated Learning for Tree-based Models. *PVLDB* 13, 11 (2020), 2090–2103.

[77] Yuncheng Wu, Naili Xing, Gang Chen, Tien Tuan Anh Dinh, Zhaojing Luo, Beng Chin Ooi, Xiaokui Xiao, and Meihui Zhang. 2023. Falcon: A Privacy-Preserving and Interpretable Vertical Federated Learning System. *PVLDB* (2023), 2471–2484.

[78] Depeng Xu, Shuhan Yuan, and Xintao Wu. 2021. Achieving Differential Privacy in Vertically Partitioned Multiparty Learning. In *Big Data*. 5474–5483.

[79] Min Xu, Bolin Ding, Tianhao Wang, and Jingren Zhou. 2020. Collecting and analyzing data jointly from multiple services under local differential privacy. *PVLDB* 13, 12 (2020), 2760–2772.

[80] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets. In *FOCS*. 162–167.

[81] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2014. PrivBayes: private data release via bayesian networks. In *SIGMOD*. 1423–1434.

[82] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional Mechanism: Regression Analysis under Differential Privacy. *PVLDB* (2012), 1364–1375.

[83] Yuqing Zhu and Yu-Xiang Wang. 2019. Poission Subsampled Rényi Differential Privacy. In *ICML*. 7634–7642.

A Additional Background

A.1 Differential Privacy

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

Lemma 10 (Analytic Gaussian Mechanism [4]). *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 erfc() *denotes the complementary error function. Namely,*

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

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

Lemma 11 (Converting (α, τ)-RDP to (ϵ, δ)-DP [12]). *Given a mechanism $\mathcal{M}$ satisfies (α, τ)-RDP for any $\alpha \in (1, \infty)$, it satisfies (ϵ, δ)-DP for $\delta > 0$ and*

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

The composition of multiple RDP mechanisms also satisfies RDP.

Lemma 12 (Composition Lemma for RDP [57]). *If mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_T$ satisfies $(\alpha, \tau_1), \ldots, (\alpha, \tau_T)$-RDP, respectively, then $\mathcal{M}_1 \circ \ldots \circ \mathcal{M}_T$ satisfies $(\alpha, \sum_{t=1}^T \tau_t)$-RDP.*

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

Lemma 13 (Subsampling for RDP [58]). *Let $\mathcal{M}$ be a mechanism that satisfies (l, τ_l)-RDP for $l = 2, \ldots, \alpha$ ($\alpha \in \mathbb{Z}, \alpha > 2$), and S_q be a procedure that uniformly samples each record of the input data with probability q. Then $\mathcal{M} \circ S_q$ satisfies (α, τ)-RDP with*

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

A.2 The BGW Protocol

For completeness, we briefly review the idea of BGW [8] in the following. A building block of BGW is Shamir's secret sharing algorithm [64], 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 [64], BGW [8] implements a three-phase execution.

(1) First, each party distributes her private input as secret shares to other clients using Shamir's algorithm [64]. 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.

B 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 3.3, this method applies to arbitrary tasks but incurs high errors.

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 $\sigma$.
1 for each client $j \in [n]$ do
2     $\tilde{\mathbf{X}}[:,j] \leftarrow \mathbf{X}[:,j] + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_m)$.
3     Client $j$ sends the perturbed $\tilde{\mathbf{X}}[:,j]$ to the server.
4 The server reconstructs $\tilde{\mathbf{X}} = (\tilde{\mathbf{X}}[:,1],\ldots,\tilde{\mathbf{X}}[:,n])$.
```

We present the privacy guarantees for Algorithm 4.

Lemma 14. *For any noise parameter σ, and integer $\alpha > 1$, Algorithm 4 satisfies (α, τ_{server}) server-observed RDP and (α, τ_{client}) server-observed RDP with $\tau_{server} = \frac{\alpha c^2}{2\mu}$, and $\tau_{client} = \frac{\alpha 2c^2}{\mu}$.*

The proof follows from Lemma 1. Note that τ_{server} and τ_{client} defer by a factor of 2^2. This is because, in client-observed DP, the sensitivity for releasing the input data is two times that in server-observed DP. The corresponding (ϵ, δ)-DP guarantees can be obtained using Lemma 10.

C Proofs

Proof of Lemma 3. 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

$$(\hat{x})^\lambda \le \gamma^\lambda x^\lambda + 2\lambda\gamma^{\lambda-1}. \tag{24}$$

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

$$(\hat{x})^\lambda \le \gamma^\lambda x^\lambda + 2\lambda x^{\lambda-1}\gamma^{\lambda-1}. \tag{25}$$

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

$$(\hat{x})^\lambda \ge \gamma^\lambda x^\lambda - 2\lambda\gamma^{\lambda-1}, \tag{26}$$

when $0 \le x < 1$, and

$$(\hat{x})^\lambda \ge \gamma^\lambda x^\lambda - 2\lambda x^{\lambda-1}\gamma^{\lambda-1}. \tag{27}$$

Here we note that the term $\lambda x^{\lambda-1}$ in Eq. (25) and (27) is bounded by the constant $\lambda c^{\lambda-1}$, which becomes negligible with respect to $\gamma^{\lambda-1}$ when γ is large. Combining Eq. (24), (25), (26), (27), we conclude that

$$(\hat{x})^\lambda = \gamma^\lambda x^\lambda + O(\gamma^{\lambda-1}). \tag{28}$$

□

Before we present the proof of Lemma 8, we first prove the following.

Lemma 15. *Let $\mathbf{V}_k$ be the rank-k subspace of the original matrix $\mathbf{X}$ and let $\tilde{\mathbf{V}}_k$ be the principal rank-k subspace of matrix $\tilde{\mathbf{C}}$ obtained from Algorithm 3 with scaling parameter $\gamma \gg n$ and noise parameter μ. Then with high probability, we have that*

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \ge \|\mathbf{X}\mathbf{V}_k\|_F^2 - O(k\sqrt{n}\sqrt{\mu}), \tag{29}$$

Proof of Lemma 15. 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.

$$\tilde{\mathbf{C}} = \gamma^2\mathbf{X}^T\mathbf{X} + \underbrace{2\gamma\mathbf{E}^T\mathbf{X} + \mathbf{E}^T\mathbf{E} + \mathbf{N}}_{\text{denoted as } \mathbf{H}}, \tag{30}$$

where $\mathbf{N}$ is the symmetric random matrix, where each entry on the upper diagonal is independently sampled from $\text{Sk}(\mu)$. 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 $\tilde{\mathbf{V}}_k$ (not $\mathbf{V}_k$) is the principal subspace of matrix $\tilde{\mathbf{C}}$, we have

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

Hence,

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

which is equivalent to

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \ge \|\mathbf{X}\mathbf{V}_k\|_F^2 - 2k\|\frac{1}{\gamma^2}\cdot\mathbf{H}\|_2. \tag{31}$$

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. [71] 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 $\mathbf{E} = \mathbf{EI}$ 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 $\text{Sk}(\mu)$. The distribution of $\text{Sk}(\mu)$ is obtained by taking the difference of two independent Poisson random variates sampled from $\text{Pois}(\mu)$. The moment generating function of $Z \sim \text{Sk}(\mu)$ is written as follows

$$\mathbb{E}[e^{\lambda Z}] = e^{\mu(e^{\lambda}+e^{-\lambda}-2)}. \tag{32}$$

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. [18] 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 7), the factor of $\sqrt{\mu}$ gets canceled when dividing it by γ^2 in Eq. 31, leaving $O(\sqrt{n}\sqrt{\mu})$ in the end. □

The proof of Lemma 8 then follows from Lemma 15 and the fact that to achieve (ϵ, δ)-server-observed DP, it suffices to set $\mu_{\epsilon,\delta} = b^2\log(1/\delta)/\epsilon^2$, where b is some constant [70].

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

Lemma 16. *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*

$$\|\mathbf{X}\tilde{\mathbf{V}}_k\|_F^2 \geq \|\mathbf{X}\mathbf{V}_k\|_F^2 - O(k\sqrt{n\alpha/\tau}). \tag{33}$$

Proof to Lemma 9. We sketch the proof as follows. We first obtain the upper bound for the $\mathcal{L}_2$ norm of Eq. (19) (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. (24), (25), (26), (27)). 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. (19) is bounded by $\sqrt{\gamma^6(\frac{3}{4})^2 12\gamma^5\sqrt{d}\frac{3}{4}\sqrt{d} + 36\gamma^4\|\mathbf{x}\|_2^2 +}$. $\sqrt{d}\frac{3}{4}$ and $\sqrt{d}$ corresponds to the maximum $\mathcal{L}_1$ norms of $f(\mathbf{w}, (\mathbf{x}, y))$ and $\mathbf{x}$, respectively. The rest of proof then follows from applying Lemma 2 to the computed $\mathcal{L}_2$ and $\mathcal{L}_1$ sensitivities and then use the results on privacy amplification by subsampling and composition theorems [57, 58, 83]. We note that τ_{client} does not benefit from subsampling, since each client already knows which record is placed in the sampled batch. □