# Dropout-Resilient Secure Multi-Party Collaborative Learning with Linear Communication Complexity

**Xingyu Lu** **Hasin Us Sami** **Başak Güler**

Department of Electrical and Computer Engineering
University of California, Riverside
xlu065@ucr.edu hsami003@ucr.edu bguler@ece.ucr.edu

## Abstract

Collaborative machine learning enables privacy-preserving training of machine learning models without collecting sensitive client data. Despite recent breakthroughs, communication bottleneck is still a major challenge against its scalability to larger networks. To address this challenge, we propose PICO, the first collaborative learning framework with linear communication complexity, significantly improving over the quadratic state-of-the-art, under formal information-theoretic privacy guarantees. Theoretical analysis demonstrates that PICO slashes the communication cost while achieving equal computational complexity, adversary resilience, robustness to client dropouts, and model accuracy to the state-of-the-art. Extensive experiments demonstrate up to $91\times$ reduction in the communication overhead, and up to $7\times$ speed-up in the wall-clock training time compared to the state-of-the-art. As such, PICO addresses a key technical challenge in multi-party collaborative learning, paving the way for future large-scale privacy-preserving learning frameworks.

## 1 INTRODUCTION

Privacy-preserving collaborative machine learning (PPML) allows multiple data owners to collaborate to train ML models without sharing their data. PPML can greatly improve ML performance by increasing the volume and diversity of data, without compromising privacy (Mohassel and Zhang, 2017; Al-Rubaie and Chang, 2019). It can even foster novel applications in which data is rare and collaboration has traditionally been limited due to privacy concerns, such as the treatment of rare diseases (Nosowsky and Giordano, 2006; Telenti and Jiang, 2020).

Recently, secure multi-party computing (MPC) has become a popular candidate for PPML (Mohassel and Zhang, 2017). Secure MPC protocols are based on a cryptographic primitive known as *secret sharing*, where parties inject randomness to local datasets before sharing it with others (Yao, 1982; Shamir, 1979). Computations are then performed on the secret shared data (Ben-Or et al., 1988; Damgård and Nielsen, 2007; Beerliová-Trubíniová and Hirt, 2008). The injected randomness is reversible, i.e., parties can decode the computations performed on the secret shared data to recover the true computation results, preserving model accuracy. Secure MPC provides strong information-theoretic privacy guarantees, such that no information about the datasets is revealed beyond the final model (even if adversaries have unbounded computational power) (Nikolaenko et al., 2013; Gascón et al., 2017; Mohassel and Rindal, 2018; Wagh et al., 2018). The main limitation of such information-theoretic PPML protocols is the intensive client communications needed to perform secure computations, preventing scalability to larger networks.

Coding theory offers a promising approach to the design of information-theoretic PPML (Yu et al., 2019; So et al., 2020, 2021). This approach, called *Lagrange Coded Computing (LCC)*, first *encodes* the datasets using a Lagrange interpolation polynomial. The encoding operation injects randomness and (computational) redundancy within the local computations, and provides information-theoretic privacy, resilience against client dropouts, and reduces the training load per client. The training computations are then performed on the encoded data, *as if they were performed on the clear data*. After multiple training rounds, the final model is *decoded* using polynomial interpolation, by collecting the computations (performed over encoded data) from individual clients. By doing so, an order-of-magnitude speed-up is achieved in the training time compared to state-of-the-art MPC baselines, where for the latter the training load per client is as large as centralized training

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. 

(over the collection of all client datasets) (So et al., 2020).

**Communication bottleneck and "degree explosion".** The major challenge against the scalability of information-theoretic PPML is its *communication complexity*, which is quadratic in the number of clients. This is caused by the multiplication operations associated with gradient computations. Specifically, interpolating a polynomial $f$ of degree $\deg(f)$ requires collecting at least $\deg(f)+1$ interpolation points. As such, decoding the final model from the local computations requires computations to be collected from at least $N \geq \deg(f)+1$ clients. On the other hand, the multiplication operations during gradient computations lead to an exponential growth in the polynomial degree, leading to a *degree explosion* after multiple training rounds. This necessitates an expensive *degree reduction step* with a quadratic communication overhead (after each training round), preventing scalability to large networks.

**Contributions.** To address this challenge, we propose PICO, the first information-theoretic PPML framework with linear communication complexity. The key intuition behind PICO is an online-offline communication trade-off, where we trade-off expensive online (data-dependent) communications with offline (data-agnostic) communications. For the online phase, we develop a novel degree reduction mechanism, which reduces the quadratic communication overhead of LCC to linear. For the offline phase, we reduce the communication overhead by reducing the *volume* of variables communicated by each client. Communicating each variable has a quadratic cost, but the *total number of variables* scales inversely with the number of clients, leading to a linear amortized overhead. As such, in a network of $N$ clients, PICO incurs an $O(N)$ communication complexity both offline and online, as opposed to the $O(N^2)$ online communication complexity of the state of the art, while achieving the same computational complexity, accuracy, dropout-resilience, and privacy guarantees.

Our theoretical analysis provides formal guarantees for information-theoretic privacy, correctness, and the key performance trade-offs in terms of the communication and computation complexity, adversary resilience, client dropouts, and training time. We perform extensive experiments to evaluate the performance of PICO, by implementing a distributed multi-client network for various image classification tasks. We then demonstrate the communication/computation volume and the wall-clock training time of PICO with respect to state-of-the-art benchmarks, identify the impact of key system parameters and trade-offs, and present the model convergence and accuracy.

Our contributions can be summarized as follows:

- We introduce PICO, the first privacy-preserving collaborative learning framework with linear communication complexity, under strong end-to-end information-theoretic privacy guarantees.
- We propose an online-offline communication trade-off for privacy-preserving collaborative learning, where communication is divided into online (data-dependent) and offline (data-agnostic) components. We develop the first degree reduction mechanism for LCC with linear online communication overhead.
- Our theoretical analysis presents formal information-theoretic privacy guarantees (for end-to-end training), and shows that PICO cuts the communication overhead while achieving the same computation complexity, adversary resilience, robustness to client dropouts, and model accuracy of the state-of-the-art.
- Our experiments demonstrate up to $91\times$ reduction in the communication overhead, and up to $7\times$ speed-up in the wall-clock training time compared to the state-of-the-art, while achieving the same adversary and dropout resilience, and model accuracy.

## 2 Related Work

**Secure Aggregation.** Recently, MPC mechanisms have also been utilized for model aggregation in distributed and federated learning, known as *secure aggregation*, where parties learn the sum of client models/gradients after each (global) training round, but without observing the individual models/gradients, (Bonawitz et al., 2017; Bell et al., 2020; So et al., 2022; Zhao and Sun, 2021). In contrast, our focus is on *end-to-end* PPML, where parties can learn only the *final model* (after multiple rounds), and no intermediate model/gradient should be revealed during training. Beyond the information-theoretic setting, there are two complementary approaches to PPML.

**Homomorphic Encryption (HE)** allows computations to be performed on encrypted data, when adversaries have bounded computational power (Gentry and Boneh, 2009; Gentry, 2009; Gilad-Bachrach et al., 2016; Hesamifard et al., 2017; Graepel et al., 2012; Yuan and Yu, 2014; Chabanne et al., 2017; Li et al., 2017; Kim et al., 2018; Wang et al., 2018; Han et al., 2019). HE can tolerate a large number of adversaries (compared to secure MPC). As a trade-off, privacy guarantees depend on the size of the encrypted data; stronger guarantees increase the data size, leading to higher computation load per client. As such, HE is typically utilized for the inference task in ML (rather than the more computation-intensive training).

**Differential Privacy (DP)** is a noisy release mechanism that protects the privacy of personally identifiable information by injecting (irreversible) noise to the computations, so that an adversary observing the released model cannot backtrack a certain individual's information (Dwork et al., 2006; Chaudhuri and Monteleoni, 2009; Shokri and Shmatikov, 2015; Abadi et al., 2016; Pathak et al., 2010; McMahan et al., 2018; Rajkumar and Agarwal, 2012; Jayaraman et al., 2018). Stronger privacy guarantees require more noise to be added to the computations, leading to a (model) accuracy-privacy trade-off in distributed settings.

Recent works show that DP can also be combined with information-theoretic PPML, to reduce the amount of noise that should be added by each client, thus improving model accuracy (Chen et al., 2022b,a; Kairouz et al., 2021). As such, although beyond the focus of the current work, we note that our techniques can also be combined with and benefit DP as an interesting future direction.

## 3 PROBLEM FORMULATION

**Collaborative logistic regression.** Our focus is on collaborative logistic regression in a network of $N$ clients. Client $i$ holds a local dataset $\mathcal{X}_i$ consisting of $|\mathcal{X}_i| = m_i$ data points, where each data point has $d$ features, along with the corresponding labels $\mathbf{y}_i \in \{0,1\}^{m_i}$. The collection of the individual datasets $\mathcal{X} \triangleq (\mathcal{X}_1, \ldots, \mathcal{X}_N)$ is represented by a matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{X}| \times d}$ consisting of $|\mathcal{X}| = \sum_{i=1}^{N} m_i$ data points, with the corresponding labels $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N) \in \{0,1\}^{|\mathcal{X}|}$. The goal is to train a logistic regression model $\mathbf{w}$ jointly over $\mathbf{X}$ (the datasets of all $N$ clients), by minimizing a cross entropy loss function:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} (-y_i \log \hat{y}_i - (1-y_i) \log(1-\hat{y}_i))) \quad (1)$$

where $\hat{y}_i = g(\mathbf{x}_i \times \mathbf{w}) \in (0,1)$ is the probability of label $i$ being equal to 1, $\mathbf{x}_i \in \mathbb{R}^d$ denotes the $i^{th}$ row of $\mathbf{X}$ (features of data point $i$), and $g(x) \triangleq 1/(1+e^{-x})$ is the sigmoid function. The model is trained via gradient descent, by updating it in the negative direction of the gradient $\nabla \mathcal{L}(\mathbf{w}) \triangleq \frac{1}{|\mathcal{X}|} \mathbf{X}^T (g(\mathbf{X} \times \mathbf{w}) - \mathbf{y})$,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta}{|\mathcal{X}|} \mathbf{X}^T (g(\mathbf{X} \times \mathbf{w}^{(t)}) - \mathbf{y}) \quad (2)$$

where $\eta$ is the learning rate, $\mathbf{w}^{(t)}$ is the estimated model parameters from training round $t$, and function $g(\cdot)$ is applied element-wise over $\mathbf{X} \times \mathbf{w}^{(t)}$. We consider a decentralized communication topology, where clients can communicate through peer-to-peer or broadcast links. At each training round, up to $D$ clients may drop out from the system due to poor connectivity (or unavailability). We do not assume the existence of a trusted third party or central coordinator. Our system model is presented in Fig. 1.

**Threat model.** We consider an *honest-but curious* adversary model, as is the most common model in PPML (So et al., 2020; Mohassel and Zhang, 2017; Nikolaenko et al., 2013; Gascón et al., 2017), where adversaries follow the protocol but try to reveal additional information about the local datasets of honest clients, using the messages exchanged during training. Up to $T$ clients are adversarial, who may collude with one another. The set of adversarial and honest clients are denoted by $\mathcal{T}$ and $\mathcal{H}$, respectively.

**Information-theoretic privacy.** Our focus is on information-theoretic privacy, i.e., to ensure that the adversaries learn no information about the local datasets of honest clients, beyond the final model (Yu et al., 2019; Mohassel and Zhang, 2017; So et al., 2020). Formally, this condition can be stated as,

$$I(\{\mathcal{X}_i, \mathbf{y}_i\}_{[N]\setminus\mathcal{T}}; \mathcal{M}_{\mathcal{T}} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) = 0 \quad (3)$$

for all $\mathcal{T}$ such that $|\mathcal{T}| \leq T$, where $I$ is the mutual information, and $\mathcal{M}_{\mathcal{T}}$ is the collection of all messages received or generated by the adversaries, and $J$ is the total the number of training rounds. Our framework is bound to finite field operations, and we assume that all datasets are represented in a finite field $\mathbb{F}_q$ of integers modulo a large prime $q$, as detailed in Appendix H. Accordingly, all operations in the sequel are carried out within $\mathbb{F}_q$.

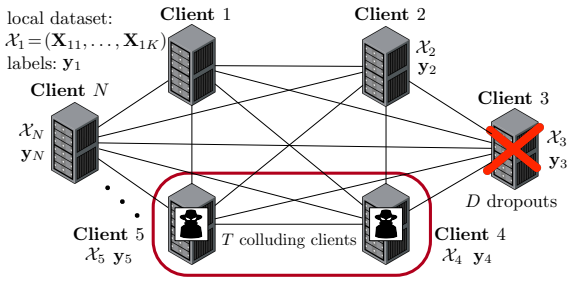



Figure 1: **System model.** The multi-client learning setup of PICO. Client $i \in [N]$ holds a dataset $\mathcal{X}_i$ with labels $\mathbf{y}_i$.

**Limitations of the state-of-the-art.** To solve (1) with the information-theoretic guarantees from (3), the state-of-the-art is the COPML framework from So et al. (2020), which builds on LCC (Yu et al., 2019). In this setup, the dataset $\mathbf{X}$ is first partitioned into $K$ equal-sized parts $(\mathbf{X}_1, \ldots, \mathbf{X}_K)$. The $K$ parts are then *encoded* using a Lagrange interpolation polynomial, by combining them with $T$ random matrices $\mathbf{R}_1, \ldots, \mathbf{R}_T$, to hide their true value against up to $T$ adversaries. At the end, client $i$ obtains a coded dataset $\widetilde{\mathbf{X}}_i = u(\mathbf{X}_1, \ldots, \mathbf{X}_K, \mathbf{R}_1, \ldots, \mathbf{R}_T, \alpha_i)$, whose size is $(1/K)^{th}$ of the original dataset $\mathbf{X}$. Client $i$ then computes the gradients on the encoded dataset $\widetilde{\mathbf{X}}_i$, *as if they were computing on the original dataset*. After $J$ training rounds, the final model $\mathbf{w}^{(J)}$ can be reconstructed from the computations performed on the encoded datasets, as long as $N \geq D + (\deg f)(K + T - 1) + 1$, where $\deg f$ quantifies the degree of the polynomial after $J$ training rounds. $K$ is called the *degree of parallelization*; as the network size $N$ grows, one can select a larger $K$ to speed up training, as each client needs to process only $(1/K)^{th}$ of the dataset $\mathbf{X}$. On the other hand, the polynomial degree $\deg f$ grows exponentially after each training round, due to the multiplications needed for the gradient in (2). To prevent a degree explosion, an expensive *degree reduction step* has to be carried out after each training round, with a quadratic communication overhead in the number of clients. This limits scalability to larger networks.

**Main problem.** Our goal is to address this challenge, where we ask the following question:

- *How can we develop a scalable PPML framework to*

Table 1: Comparison of the total communication overhead (across all clients) for PICO and COPML, where $K = O(N)$, $T = O(N)$, and $m_i = m$ for all $i \in [N]$.

| | COPML | PICO |
|---|---|---|
| 1.Dataset encoding | $O(N^2dm)$ | $O(Ndm)$ |
| 2.Label encoding | $O(N^2m+N^2d)$ | $O(Nd)$ |
| 3.Model initialization | $O(N^2d)$ | $O(Nd)$ |
| 4.Model encoding | $O(N^2dJ)$ | $O(NdJ)$ |
| 5.Gradient cmp./model update | $O(N^2dJ)$ | $O(NdJ)$ |

*solve* (1) *with linear communication complexity, under formal information-theoretic guarantees from* (3)*?*

To address this challenge, in this work we introduce PICO, a privacy-preserving collaborative learning framework with linear communication complexity.

The key contribution of PICO is a novel encoding and degree reduction strategy that incurs a linear communication overhead ($O(N)$ broadcast), as opposed to the quadratic ($O(N^2)$ point-to-point) overhead of the state-of-the-art.

**An offline-online trade-off.** To do so, PICO decouples communication into *online (data-dependent)* and *offline (data-agnostic) phases.* Online phase depends on the datasets, hence can only take place after training starts. For this phase, we introduce a novel degree reduction mechanism for LCC, that reduces the communication overhead from $O(N^2)$ (point-to-point) to $O(N)$ (broadcast). The remaining communication-intensive operations are offloaded to the offline phase, where the communicated variables are independent from the training data, such as randomness generation for the encoding operation. Then, the number of communicated variables is reduced through an efficient randomness generation mechanism, using linear transformations with MDS (maximum distance separable) matrices (also related to hyperinvertible matrices from (Beerliová-Trubíniová and Hirt, 2008)). Here, the communication overhead for each variable is quadratic, but the *total number of variables* is *inversely proportional to the number of clients*, overall achieving an $O(N)$ amortized communication overhead. In doing so, PICO preserves the **same computation complexity as the state-of-the-art**; due to the reduced number of variables, the additional computations with MDS matrices do not increase the overall complexity. The flexible modular architecture also provides further opportunities for system-wide performance optimization. The offline phase can take place in advance when the network load is low, or can be overlapped with other components of training. We next describe the details of PICO.

## 4 PICO FRAMEWORK

PICO consists of five main components, consisting of online and offline phases as shown in Fig. 2. Table 1 presents the communication overhead of each component with respect to COPML (So et al., 2020). For ease of presentation, we describe the offline and online phases sequentially, to show how the variables generated in the former are utilized in the latter. We note, however, that each offline phase is independent from past online/offline phases. Specifically, variables generated in the offline phase of any component do not depend on the online/offline phases of previous components, hence **all offline phases can be executed in parallel**. We now describe the details of each component.

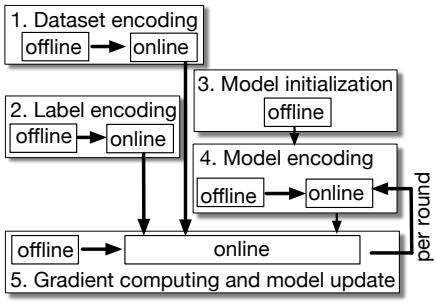



Figure 2: Flowchart of PICO.

**1. Dataset Encoding.** Initially, clients encode their datasets using locally generated randomness. The goal of the encoding process is two-fold. First, it hides the dataset contents against adversaries. Second, it reduces the size of the data each client should process during training. Specifically, each client computes the gradient on an encoded dataset $\widetilde{\mathbf{X}}_i$, whose size is $(1/K)^{th}$ of the original dataset $\mathbf{X}$. As the network size $N$ increases, one can select a larger $K$, reducing the computation load for training per worker (called the *parallelization gain*) to speed up training. As opposed to the expensive gradient computations, the encoding operations are much cheaper (we focus on linear encoding mechanisms) and we delegate the more intensive computations to the offline phase. The encoding process consists of the following offline and online phases.

**(Offline)** Initially, clients agree on $N+K+T$ distinct publicly known parameters $\{\alpha_j\}_{j\in[N]}$ and $\{\beta_j\}_{j\in[K+T]}$ from $\mathbb{F}_q$, where $\alpha_j \neq \beta_k$ for all $j \in [N]$ and $k \in [K+T]$. Client $i$ then generates $K+T$ random matrices $\{\mathbf{R}_{ik}\}_{k\in[K]}$, and $\{\mathbf{V}_{ik}\}_{k\in\{K+1,\ldots,K+T\}}$, each of size $\frac{m_i}{K} \times d$, encodes them using a Lagrange polynomial of degree $K+T-1$:

$$u_i(z) = \sum_{k\in[K]} \mathbf{R}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T} \mathbf{V}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} \quad (4)$$

and sends to each client $j \in [N]$ an encoded matrix,

$$\widetilde{\mathbf{R}}_{ij} \triangleq u_i(\alpha_j) \quad (5)$$

which corresponds to (4) evaluated at $\alpha_j$. The additional randomness $\{\mathbf{V}_{i,K+1}, \ldots, \mathbf{V}_{i,K+T}\}$ is to hide the true value of $\mathbf{R}_{ik}$ against up to $T$ adversaries.

**(Online)** In the online phase, client $i \in [N]$ partitions its local dataset $\mathcal{X}_i$ into $K$ submatrices $(\mathbf{X}_{i1}, \ldots, \mathbf{X}_{iK})$, where

$\mathbf{X}_{ik} \in \mathbb{F}_q^{\frac{m_i}{K} \times d}$ for all $k \in [K]$, and broadcasts,

$$\hat{\mathbf{X}}_{ik} = \mathbf{X}_{ik} - \mathbf{R}_{ik} \quad \forall k \in [K]. \tag{6}$$

After receiving $\{\hat{\mathbf{X}}_{jk}\}_{j\in[N],k\in[K]}$, client $i$ generates an encoded dataset:

$$\widetilde{\mathbf{X}}_i \triangleq \sum_{k\in[K]} (\hat{\mathbf{X}}_{1k}^T, \ldots, \hat{\mathbf{X}}_{Nk}^T)^T \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i - \beta_l}{\beta_k - \beta_l} + (\widetilde{\mathbf{R}}_{1i}^T, \ldots, \widetilde{\mathbf{R}}_{Ni}^T)^T \tag{7}$$

Intuitively, the encoding operation from (7) cancels the additive randomness due to $\{\mathbf{R}_{jk}\}_{k\in[K]}$, and at the same time embeds the entire dataset $\mathbf{X}$ in a Lagrange polynomial,

$$u(z) \triangleq \sum_{k\in[K]} \mathbf{X}_k \prod_{l\in[K+T]\setminus\{k\}} \frac{z - \beta_l}{\beta_k - \beta_l} + \sum_{k=K+1}^{K+T} \mathbf{V}_k \prod_{l\in[K+T]\setminus\{k\}} \frac{z - \beta_l}{\beta_k - \beta_l} \tag{8}$$

such that $\mathbf{X}_k \triangleq (\mathbf{X}_{1k}^T, \ldots, \mathbf{X}_{Nk}^T)^T$ and $\mathbf{V}_k \triangleq (\mathbf{V}_{1k}^T, \ldots, \mathbf{V}_{Nk}^T)^T$, where $u(\beta_k) = \mathbf{X}_k$ for all $k \in [K]$, and client $i \in [N]$ obtains the encoded dataset $\widetilde{\mathbf{X}}_i = u(\alpha_i)$. The $T$ random matrices $\{\mathbf{V}_k\}_{k\in\{K+1,\ldots,K+T\}}$ hide the true values of the local datasets against up to $T$ adversaries.

Eq. (7) reflects a key intuition behind PICO. To construct an encoded matrix $\widetilde{\mathbf{X}}_i = u(\alpha_i)$ using the Lagrange polynomial (8), (So et al., 2020) lets each client secret share their dataset $\mathcal{X}_i$ using Shamir's secret sharing (quadratic overhead). Then, encoding is done using the secret shared datasets. Instead, PICO lets each client broadcast a *masked* dataset as in (6) (linear overhead), where the true dataset is hidden by random matrices $\mathbf{R}_{ik}$. Then, the encoded matrix $\widetilde{\mathbf{X}}_i$ is constructed by using the additional randomness (5) generated during the offline phase. This dramatically reduces the online communication overhead, by moving the more intensive communications to the offline phase.

**2. Label Encoding.** In addition to encoding the datasets, clients also encode the *labels* to preserve their privacy, through the following offline and online phases.

**(Offline)** Client $i$ generates $K$ uniformly random vectors $\{\mathbf{a}_{ik}\}_{k\in[K]}$ of size $\frac{d}{(N-T)K}$, and sends an encoded vector,

$$\widetilde{\mathbf{a}}_{ij} = \sum_{k\in[K]} \mathbf{a}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l} + \sum_{k=K+1}^{K+T} \mathbf{b}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l} \tag{9}$$

to client $j \in [N]$, where $\mathbf{b}_{ik}$ for $k \in \{K+1, \ldots, K+T\}$ are uniformly random vectors of size $\frac{d}{(N-T)K}$. After receiving $\{\widetilde{\mathbf{a}}_{ji}\}_{j\in[N]}$, client $i$ combines them to form a new (larger dimensional) encoded vector,

$$\widetilde{\mathbf{a}}_i \triangleq (\mathbf{M} \otimes \mathbf{I}) \times (\widetilde{\mathbf{a}}_{1i}^T, \ldots, \widetilde{\mathbf{a}}_{Ni}^T)^T \tag{10}$$

$$= \sum_{k\in[K]} \mathbf{a}_k \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i - \beta_l}{\beta_k - \beta_l} + \sum_{k=K+1}^{K+T} \mathbf{b}_k \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i - \beta_l}{\beta_k - \beta_l} \tag{11}$$

where $\otimes$ is the Kronecker product, $\mathbf{a}_k \triangleq (\mathbf{M} \otimes \mathbf{I}) \times (\mathbf{a}_{1k}^T, \ldots, \mathbf{a}_{Nk}^T)^T$, and $\mathbf{b}_k \triangleq (\mathbf{M} \otimes \mathbf{I}) \times (\mathbf{b}_{1k}^T, \ldots, \mathbf{b}_{Nk}^T)^T$, $\mathbf{I}$ is a $\frac{d}{(N-T)K} \times \frac{d}{(N-T)K}$ identity matrix, and

$$\mathbf{M} = \begin{bmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{N-T} & \ldots & \lambda_{N-T}^{N-1} \end{bmatrix} \tag{12}$$

is a $(N-T) \times N$ MDS matrix, where $\lambda_1, \ldots, \lambda_{N-T}$ are distinct public parameters from $\mathbb{F}_q$. The key intuition is that, to generate an encoded vector of size $\frac{d}{K}$, each client only sends $\frac{d}{(N-T)K}$ parameters to every other client, as opposed to sending $\frac{d}{K}$ parameters as in conventional LCC (Yu et al., 2019). The final encoded vector is then generated by combining the *lower-dimensional* encoded vectors received from all $N$ clients, using the MDS matrix $\mathbf{M}$.

Next, client $i \in [N]$ sends a secret share $[\mathbf{a}_{ik}]_j$ of $\mathbf{a}_{ik}$ to client $j \in [N]$, for all $k \in [K]$, using Shamir's $T$-out-of-$N$ secret sharing. After receiving the secret shares $\{[\mathbf{a}_{jk}]_i\}_{j\in[N]}$, client $i$ constructs a new (larger dimensional) secret share $[\mathbf{a}_k]_i \triangleq (\mathbf{M} \otimes \mathbf{I})([\mathbf{a}_{1k}]_i^T, \ldots, [\mathbf{a}_{Nk}]_i^T)^T$. The goal is again to generate a secret share of dimension $\frac{d}{K}$, where each client sends only a vector of size $\frac{d}{(N-T)K}$ (as opposed to $\frac{d}{K}$). The details of Shamir's secret sharing and cryptographic primitives are provided in Appendix A.

**(Online)** Let $y_l$ denote the label for data point $\mathbf{x}_l \in \mathcal{X}_i$. First, client $i$ partitions $\sum_{l\in\mathcal{X}_i} \mathbf{x}_l^T y_l$ into $K$ equal-sized parts $(\mathbf{y}_{i1}, \ldots, \mathbf{y}_{iK})$, and sends an encoded vector

$$\widetilde{\mathbf{y}}_{ij} \triangleq \sum_{k\in[K]} \mathbf{y}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l} + \sum_{k=K+1}^{K+T} \mathbf{r}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l} \tag{13}$$

to client $j \in [N]$, where $\mathbf{r}_{ik} \in \mathbf{F}_q^{\frac{d}{K}}$ are uniformly random vectors[1]. After receiving $\{\widetilde{\mathbf{y}}_{ji}\}_{j\in[N]}$, client $i$ broadcasts:

$$\hat{\mathbf{a}}_i \triangleq \sum_{j\in[N]} \widetilde{\mathbf{y}}_{ji} - \widetilde{\mathbf{a}}_i \tag{14}$$

which is also a Lagrange polynomial of degree $K+T-1$. As such, upon receiving $\hat{\mathbf{a}}_i$ from any $K+T$ clients, all clients can recover $\sum_{j\in[N]} \mathbf{y}_{jk} - \mathbf{a}_k$ for all $k \in [K]$ (via polynomial interpolation), and compute a secret share of $\mathbf{X}^T\mathbf{y} = \sum_{l\in\mathcal{X}_i} \mathbf{x}_l^T y_l$ as follows,

[1]Typically $d \gg N$ in real-world tasks (Codella et al., 2018).

$$[\mathbf{X}^T\mathbf{y}]_i \triangleq ((\sum_{j\in[N]} \mathbf{y}_{j1}-\mathbf{a}_1+[\mathbf{a}_1]_i)^T,\ldots, (\sum_{j\in[N]} \mathbf{y}_{jK}-\mathbf{a}_K+[\mathbf{a}_K]_i)^T)^T \quad (15)$$

**3. Model Initialization.** Model $\mathbf{w}^{(0)}$ at time $t=0$ is initialized randomly (offline), without revealing it to any client. To do so, client $i$ generates a random vector $\mathbf{w}_i^{(0)}$ of size $\frac{d}{N-T}$, and sends a secret share $[\mathbf{w}_i^{(0)}]_j$ to client $j\in[N]$ using Shamir's $T$-out-of-$N$ secret sharing. After receiving $[\mathbf{w}_j^{(0)}]_i$ for $j\in[N]$, client $i$ constructs a new (larger) share,

$$[\mathbf{w}^{(0)}]_i \triangleq (\mathbf{M}\otimes\mathbf{I})\times(([\mathbf{w}_1^{(0)}]_i)^T,\ldots,([\mathbf{w}_N^{(0)}]_i)^T)^T \quad (16)$$

which represents a secret share of the initial model,

$$\mathbf{w}^{(0)} = (\mathbf{M}\otimes\mathbf{I})\times((\mathbf{w}_1^{(0)})^T,\ldots,(\mathbf{w}_N^{(0)})^T)^T \quad (17)$$

where $\mathbf{I}$ is a $\frac{d}{N-T}\times\frac{d}{N-T}$ identity matrix.

**4. Model Encoding.** At the beginning of each round, client $i$ holds a secret share $[\mathbf{w}^{(t)}]_i$ of the current state of the model $\mathbf{w}^{(t)}$. Initially at $t=0$, $[\mathbf{w}^{(0)}]_i$ is generated as described during the model initialization in (16). For all other training rounds (i.e., $t>0$), $[\mathbf{w}^{(t)}]_i$ is obtained after model updating from (34). At each round, clients then *encode* the model using the secret shares $[\mathbf{w}^{(t)}]_i$, to ensure that gradients can later be computed on the encoded dataset and model. At the end, client $i$ learns an encoded model $\widetilde{\mathbf{w}}_i^{(t)}$. Model encoding consists of the following phases.

**(Offline)** Client $i\in[N]$ generates $T+1$ uniformly random vectors[2] $\mathbf{r}_i$ and $\{\mathbf{v}_{ik}\}_{k\in\{K+1,\ldots,K+T\}}$ of size $\frac{d}{N-T}$. Then, $\mathbf{r}_i$ is secret shared using Shamir's T-out-of-N secret sharing, by sending a share $[\mathbf{r}_i]_j$ to each client $j\in[N]$. Next, client $i$ *encodes* $\mathbf{r}_i$ by generating a Lagrange polynomial,

$$v_i(z) \triangleq \sum_{k\in[K]} \mathbf{r}_i \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T} \mathbf{v}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} \quad (18)$$

and sends each client $j\in[N]$ an encoded vector,

$$\widetilde{\mathbf{r}}_{ij} \triangleq v_i(\alpha_j) \quad (19)$$

By combining the received $\{\widetilde{\mathbf{r}}_{ji},[\mathbf{r}_j]_i\}_{j\in[N]}$, client $i$ generates a (larger dimensional) encoded vector,

$$\widetilde{\mathbf{r}}_i \triangleq (\mathbf{M}\otimes\mathbf{I})\times(\widetilde{\mathbf{r}}_{1i}^T,\ldots,\widetilde{\mathbf{r}}_{Ni}^T)^T, \quad (20)$$

and a secret share of $\mathbf{r}\triangleq(\mathbf{M}\otimes\mathbf{I})\times(\mathbf{r}_1^T,\ldots,\mathbf{r}_N^T)^T$,

$$[\mathbf{r}]_i \triangleq (\mathbf{M}\otimes\mathbf{I})\times([\mathbf{r}_1]_i^T,\ldots,[\mathbf{r}_N]_i^T)^T, \quad (21)$$

**(Online)** Client $i$ initially broadcasts,

$$[\widehat{\mathbf{w}}^{(t)}]_i \triangleq [\mathbf{w}^{(t)}]_i - [\mathbf{r}]_i = [\mathbf{w}^{(t)}-\mathbf{r}]_i \quad (22)$$

[2]For simplicity, we omit the time index from random vectors, and note that at each round, a new set of random vectors is used.

After receiving $\{[\widehat{\mathbf{w}}^{(t)}]_i\}_{i\in[N]}$, each client can decode,

$$\widehat{\mathbf{w}}^{(t)} = \mathbf{w}^{(t)} - \mathbf{r} \quad (23)$$

via polynomial interpolation. Using (23), client $i$ then constructs an encoded model,

$$\widetilde{\mathbf{w}}_i^{(t)} \triangleq \sum_{k\in[K]} \widehat{\mathbf{w}}^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i-\beta_l}{\beta_k-\beta_l} + \widetilde{\mathbf{r}}_i \quad (24)$$

Intuitively, the encoding operation in (24) embeds the model $\mathbf{w}^{(t)}$ in a Lagrange polynomial,

$$v(z) \triangleq \sum_{k\in[K]} \mathbf{w}^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T} \mathbf{v}_k^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} \quad (25)$$

such that $\mathbf{v}_k\triangleq(\mathbf{M}\otimes\mathbf{I})\times(\mathbf{v}_{1k}^T,\ldots,\mathbf{v}_{Nk}^T)^T$, where $v(\beta_k)=\mathbf{w}^{(t)}$ for $k\in[K]$, and client $i$ obtains an encoded model $\widetilde{\mathbf{w}}_i^{(t)}=v(\alpha_i)$. The random vectors $\{\mathbf{v}_k\}_{k\in\{K+1,\ldots,K+T\}}$ hide the true value of $\mathbf{w}^{(t)}$ against up to $T$ adversaries.

**5. Gradient Computing and Model Update.** The last component of PICO is local gradient computation and model update, using the encoded datasets and model. At the end, client $i$ learns a secret share $[\mathbf{w}^{(t+1)}]_i$ of the model $\mathbf{w}^{(t+1)}$ for the next training round. PPML frameworks that utilize polynomial embeddings to hide sensitive data are bound to polynomial computations (So et al., 2020; Yu et al., 2019). As the sigmoid function in (1) is not a polynomial, it is often approximated with a polynomial $\hat{g}(x)=\sum_{i=0}^r\theta_i x^i$ for some public coefficients $\{\theta_i\}_{i\in[r]}$ fitted via least squares, using which one can rewrite (2) as:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta}{|\mathcal{X}|}\mathbf{X}^T(\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})-\mathbf{y}). \quad (26)$$

The degree $r$ quantifies the accuracy of approximation (Brinkhuis and Tikhomirov, 2005). Then, the offline and online phases of this stage proceed as follows.

**(Offline)** Client $i\in[N]$ generates $C\triangleq(2r+1)(K+T-1)+1$ random vectors $\mathbf{u}_{ik}$ of size $\frac{d}{N-T}$, which are then *encoded* using Lagrange polynomial of degree $C-1$,

$$\phi_i(z) \triangleq \sum_{k\in[C]} \mathbf{u}_{ik} \prod_{l\in[C]\setminus\{k\}} \frac{z-\theta_l}{\theta_k-\theta_l} \quad (27)$$

where $\theta_k=\beta_k$ (hence $\phi_i(\beta_k)=\mathbf{u}_{ik}$) for $k\in[K]$, and $\theta_k$ for $k\in\{K+1,\ldots,C\}$ are distinct public parameters from $\mathbb{F}_q$. Client $i$ then sends an encoded vector,

$$\widetilde{\mathbf{u}}_{ij} \triangleq \phi_i(\alpha_j) \quad (28)$$

to client $j\in[N]$. After receiving $\{\widetilde{\mathbf{u}}_{ji}\}_{j\in[N]}$, client $i$ constructs a new (larger dimensional) encoded vector,

$$\widetilde{\mathbf{u}}_i \triangleq (\mathbf{M}\otimes\mathbf{I})\times(\widetilde{\mathbf{u}}_{1i}^T,\ldots,\widetilde{\mathbf{u}}_{Ni}^T)^T \quad (29)$$

which encodes $\{\mathbf{u}_k\}_{k\in[C]}$ within a Lagrange polynomial,

$$\phi(z) \triangleq \sum_{k\in[C]} \mathbf{u}_k \prod_{l\in[C]\setminus\{k\}} \frac{z-\theta_l}{\theta_k-\theta_l} \tag{30}$$

such that $\mathbf{u}_k \triangleq (\mathbf{M}\otimes\mathbf{I})\times(\mathbf{u}_{1k}^T,\ldots,\mathbf{u}_{Nk}^T)^T$, where $\phi(\theta_k)=\mathbf{u}_k$ for all $k\in[C]$, and client $i$ obtains an encoded vector $\widetilde{\mathbf{u}}_i=\phi(\alpha_i)$. Client $i$ then secret shares the sum $\sum_{k\in[K]}\mathbf{u}_{ik}$, by sending client $j\in[N]$ a share,

$$[\sum_{k\in[K]}\mathbf{u}_{ik}]_j \triangleq \sum_{k\in[K]}\mathbf{u}_{ik} + \sum_{l\in[T]}\gamma_j^l\mathbf{z}_{il} \tag{31}$$

where $\mathbf{z}_{il}$ are uniformly random vectors, and $\{\gamma_j\}_{j\in[N]}$ are distinct public parameters. After receiving $[\sum_{k\in[K]}\mathbf{u}_{jk}]_i$ for $j\in[N]$, client $i$ generates a secret share of $\sum_{k\in[K]}\mathbf{u}_k$,

$$[\sum_{k\in[K]}\mathbf{u}_k]_i \triangleq (\mathbf{M}\otimes\mathbf{I})\times([\sum_{k\in[K]}\mathbf{u}_{1k}]_i^T,\ldots,[\sum_{k\in[K]}\mathbf{u}_{Nk}]_i^T)^T$$

**(Online)** Using the encoded dataset $\widetilde{\mathbf{X}}_i$ and model $\widetilde{\mathbf{w}}_i^{(t)}$, client $i$ computes a local gradient,

$$f(\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{w}}_i^{(t)}) \triangleq \widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i\times\widetilde{\mathbf{w}}_i^{(t)}) \tag{32}$$

and broadcasts $\hat{\mathbf{u}}_i \triangleq \widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i\times\widetilde{\mathbf{w}}_i^{(t)}) - \widetilde{\mathbf{u}}_i$, where the gradient is hidden by $\widetilde{\mathbf{u}}_i$. From (8) and (25), one can define a univariate polynomial $h(z)=f(u(z),v(z))=u(z)^T\hat{g}(u(z)\times v(z))$ of degree $C-1$, where,

$$h(\beta_k)=f(u(\beta_k),v(\beta_k))=f(\mathbf{X}_k,\mathbf{w}^{(t)})=\mathbf{X}_k^T\hat{g}(\mathbf{X}_k\times\mathbf{w}^{(t)})$$

for all $k\in[K]$, and client $i\in[N]$ computes,

$$h(\alpha_i)=f(u(\alpha_i),v(\alpha_i))=f(\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{w}}_i^{(t)})=\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i\times\widetilde{\mathbf{w}}_i^{(t)})$$

Define $\psi(z)\triangleq h(z)-\phi(z)$. Note that $\hat{\mathbf{u}}_i=\psi(\alpha_i)$, hence after receiving $\hat{\mathbf{u}}_i$ from any set of at least $\deg(\psi)+1=C$ clients, each client can reconstruct $\psi(z)$ via polynomial interpolation, and compute a secret share of the *true* gradient $\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})=\sum_{k\in[K]}\mathbf{X}_k^T\hat{g}(\mathbf{X}_k\times\mathbf{w}^{(t)})$ as follows,

$$\begin{aligned}[\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})]_i &\triangleq \sum_{k\in[K]}\psi(\beta_k) + [\sum_{k\in[K]}\mathbf{u}_k]_i \\ &= \mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)}) + \sum_{l\in[T]}\gamma_i^l\mathbf{z}_l,\end{aligned} \tag{33}$$

where $\mathbf{z}_l\triangleq(\mathbf{M}\otimes\mathbf{I})\times(\mathbf{z}_{1l}^T,\ldots,\mathbf{z}_{Nl}^T)^T$. Note that PICO is bound to polynomial operations, consisting only of finite field addition and multiplications, whereas (2) requires a division. To handle this, one can either treat all model updates as operations in the integer domain, by assuming a sufficiently large field size, as detailed in Appendix D, or utilize the secure truncation operation from (Catrina and Saxena, 2010) as described in Appendix I and update the model according to (26),

$$[\mathbf{w}^{(t+1)}]_i = [\mathbf{w}^{(t)}]_i - \frac{\eta}{|\mathcal{X}|}([\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})]_i - [\mathbf{X}^T\mathbf{y}]_i) \tag{34}$$

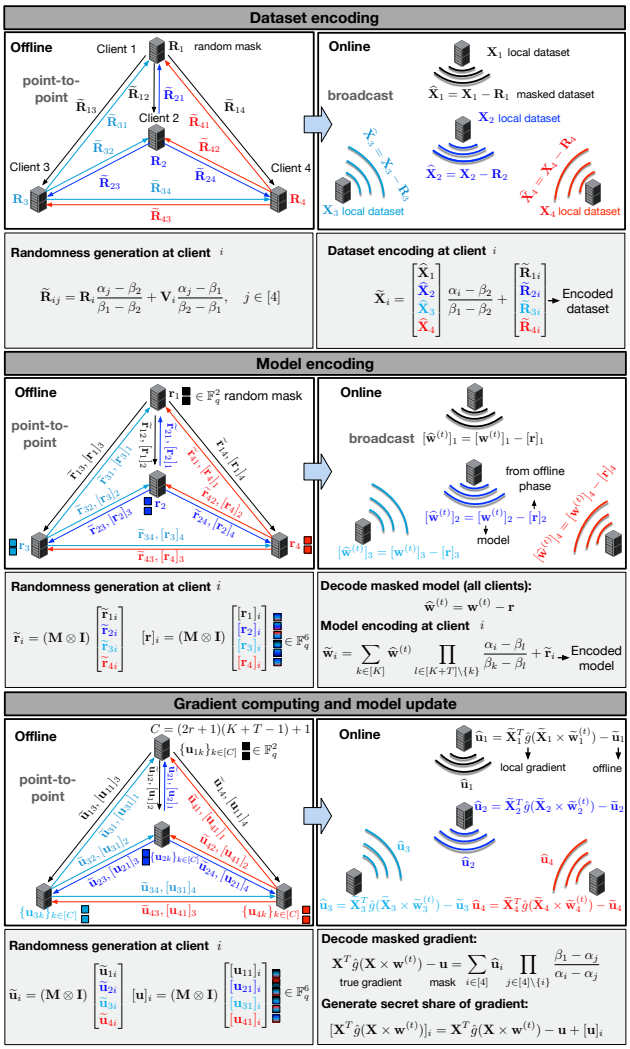



Figure 3: **Motivating example.** An illustrative example for PICO with $N=4$, $d=6$, and $K=T=1$. The randomness generated in the offline phase is utilized in the online phase.

which reduces the required field size (albeit with weaker privacy) in practice (So et al., 2020). In our theoretical analysis in Section 5, we assume a sufficiently large field size and consider the former, whereas we utilize the latter in our experiments from Section 6.

**Final Model Recovery.** After $J$ training rounds, clients collect the secret shares $\{[\mathbf{w}^{(J)}]_i\}_{i\in[N]}$ to decode $\mathbf{w}^{(J)}$.

Our overall algorithm is given in Appendix B. Appendix C illustrates the variables generated at each phase. In Fig. 3, we demonstrate an illustrative example for PICO, by letting $N=4,d=6,K=T=1$. For notational simplicity, we exclude the subscripts corresponding to $k\in[K]$ (as $K=1$) whenever possible. We then present the offline/online phases for the dataset and model encoding stages, along with gradient computing and model update, to demonstrate how the smaller dimensional random vectors generated by each client are combined to generate large dimensional shared randomness.

## 5 THEORETICAL ANALYSIS

We first present the formal privacy guarantees from (3).

**Theorem 1.** *(Information-theoretic privacy) PICO guarantees information theoretic-privacy:*

$$I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_\mathcal{T} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) = 0 \quad (35)$$

*where $\mathcal{M}_\mathcal{T}$ denotes the collection of all messages received or generated by the adversaries throughout the training.*

*Proof.* The proof is provided in Appendix D. □

We next present the total communication complexity (across all clients), and the per-client computation complexity. To explicitly demonstrate the complexity with respect to the number of clients, we let $m_i = m$ for $i \in [N]$.

**Theorem 2.** *(Communication complexity) The total communication complexity of PICO is $O(Ndm + \frac{N^2}{K}d + NdJ)$ in the online phase, and $O(\frac{N^2}{K}dm + \frac{N^2}{N-T}dJ)$ in the offline phase. With $K = O(N)$ and $T = O(N)$, the communication complexity (offline+online) is linear in the number of clients, which is $O(Ndm + NdJ)$.*

*Proof.* The proof is provided in Appendix E. □

**Theorem 3.** *(Computation complexity) PICO incurs a per-client computation overhead of $O(Nmd + N\frac{d}{K}\log^2(K+T)\log\log(K+T) + J\frac{Nm}{K}(d+r) + Jdr(K+T)\log^2 r(K+T)\log\log r(K+T))$ in the online phase, and $O(Nd\frac{m}{K}\log^2(K+T)\log\log(K+T) + JN\frac{d}{N-T}\log^2 r(K+T)\log\log r(K+T) + JNd)$ in the offline phase.*

*Proof.* The proof is provided in Appendix F. □

In Appendix F, we also compare the computational complexity of PICO with COPML, and show that PICO reaches the same computational complexity as COPML.

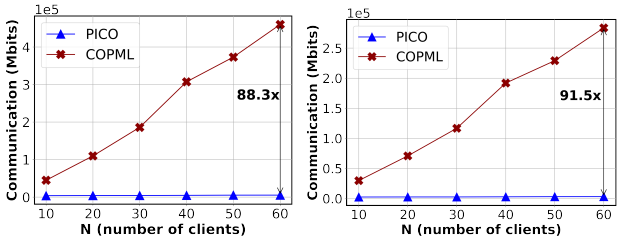
The *recovery threshold* is defined as the minimum number of clients needed for correct recovery of the final model.

**Theorem 4.** *(Recovery threshold) In a network of $N$ clients, where up to $T$ clients are adversarial, and up to $D$ clients may drop out (or are unavailable) in each training round, the recovery threshold of PICO is $N \geq D + (2r+1)(K+T-1)+1$.*

*Proof.* The minimum number of clients is determined by the number of local computations required for polynomial interpolation, which, from Section 4 is given by $N - D \geq (2r+1)(K+T-1)+1$ (same as COPML). Hence, PICO achieves the same adversary robustness ($T$), dropout resilience ($D$), and parallelization ($K$) guarantees, while also slashing the communication overhead. □

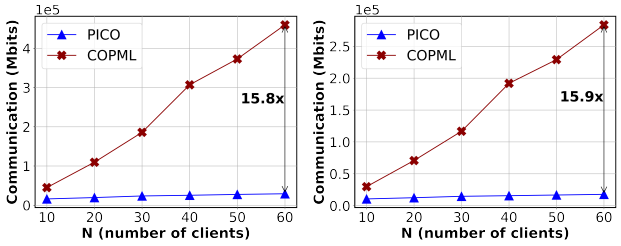
**Remark 1.** *PICO can also be applied to the simpler linear regression problem, with the same protocol steps.*

We next show that the finite field training operations preserve the target model update from (26).




(a) CIFAR-10. (b) MNIST.

Figure 4: Online communication overhead.




(a) CIFAR-10. (b) MNIST.

Figure 5: Online+offline communication overhead.

**Theorem 5.** *(Correctness) PICO correctly recovers the target model from (26), within a sufficiently large field $\mathbb{F}_q$.*

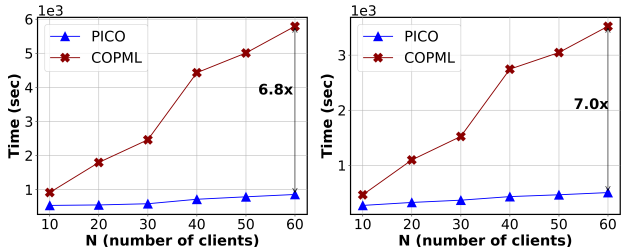
*Proof.* The proof is given in Appendix G. □

## 6 EXPERIMENTS

**Setup.** We train a logistic regression model for binary classification on CIFAR-10 (Krizhevsky and Hinton, 2009) (on classes *plane* and *car*), and MNIST (LeCun et al., 2010) (on digits 0 and 1), with dataset sizes $(m, d) = (9019, 3073)$ and $(11432, 785)$, respectively, distributed evenly across clients. The multi-client network is implemented using the `MPI4Py` Message Passing Interface (MPI) (Dalcín et al., 2005). The hyperparameters are $J = 50$ and $\eta = 1.4 \times 10^{-7}$, respectively. Additional experimental details are provided in Appendix I.

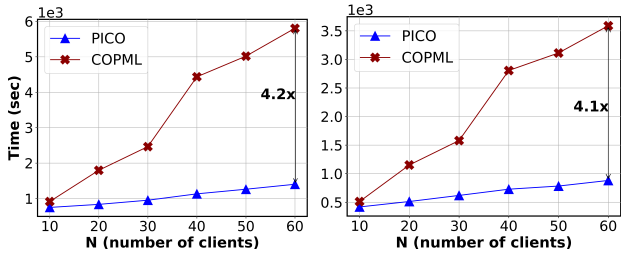
**Benchmark.** We evaluate the performance with respect to the state-of-the-art multi-party logistic regression framework with end-to-end information-theoretic privacy (with $N > 4$), which is the COPML framework from So et al. (2020). For both frameworks, we leverage the secure truncation operation suggested in So et al. (2020) to reduce the size of the finite field during model update. Furthermore, we optimize (speed up) COPML by leveraging the grouping strategy suggested in So et al. (2020), which partitions clients into groups of size $T+1$, and communicates the secret shares only between clients within the same group.

**Performance evaluation.** To ensure correct recovery of the final model, the number of clients (for both PICO and COPML) must satisfy the recovery threshold from Thm. 4. In accordance with So et al. (2020), we then let $r = 1$, and first consider the scenario where the degree of privacy ($T$) and parallelization ($K$) are (almost) equal, by letting $N =$




(a) CIFAR-10. (b) MNIST.

Figure 6: Online wall-clock training time.




(a) CIFAR-10. (b) MNIST.

Figure 7: Online+offline wall-clock training time.

$3(K+T-1)+1$ with $T = \lfloor \frac{N-3}{6} \rfloor$ and $K = \lfloor \frac{N+2}{3} \rfloor - T$. Similar to So et al. (2020), the bandwidth and finite field size are set as 40Mbps and $q = 2^{26} - 5$, respectively.

**Communication overhead.** We first compare the online communication overhead (in Mbits) of PICO and COPML in Fig. 4. This includes all communication during the online phases throughout training. We observe that PICO significantly decreases the communication overhead, by up to $88.3\times$ and $91.5\times$ on CIFAR-10 and MNIST, respectively. Note that some one-time communications (i.e., secret sharing the dataset/labels) were omitted in So et al. (2020), which we also include as they are data-dependent. In Fig. 5, we compare the overall (online+offline) communication overhead of PICO with COPML (where all communication is online), and observe a reduction by up to $15.8\times$ on CIFAR-10 and $15.9\times$ on MNIST.

**Wall-clock training time.** In Fig. 6, we compare the wall-clock training time of PICO and COPML, including all (online) communication and computations. We observe that PICO speeds-up the training time by up to $6.8\times$ and $7\times$ on CIFAR-10 and MNIST, respectively. In Fig. 7, we demonstrate the overall wall-clock time by including all online and offline operations. We observe that PICO reduces the overall wall-clock time by up to $4.2\times$ and $4.1\times$ for CIFAR-10 and MNIST, respectively. Therefore, the additional offline operations (performed in PICO) to speed-up the online training time does not increase the overall (offline+online) wall-clock time compared to COPML.

**Model accuracy.** In Fig. 8, we compare the model test accuracy with respect to both COPML and conventional logistic regression (representing our target accuracy) for $N = 60$ on CIFAR-10. We observe that PICO achieves comparable accuracy to both COPML and conventional logistic regression. Note that for conventional logistic regression, training is done without any privacy constraints, in a centralized setting where all data is located at a single party.

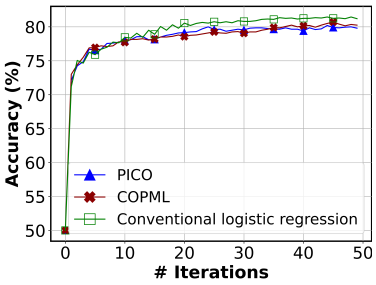
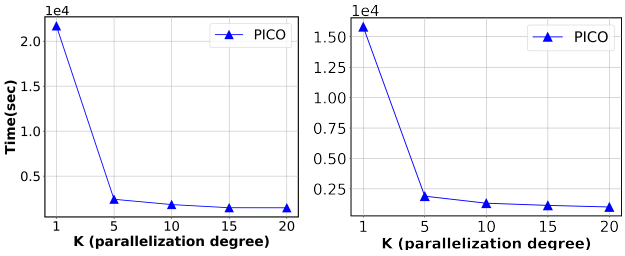



Figure 8: Model convergence (CIFAR-10).




(a) CIFAR-10. (b) MNIST.

Figure 9: Online+offline wall-clock training time.

**Impact of K (degree of parallelization).** In Fig. 9, we demonstrate the role of parameter $K$ on the overall (offline+online) wall-clock training time of PICO (including all communication and computations), by letting $N = 60$ and varying $K$. As $K$ increases, training time decreases, as the size of the encoded dataset processed by each client is proportional to $1/K$ (reducing the training load per client). Fig. 9 also illustrates a trade-off between parallelization (accordingly, training time) and adversary resilience, as increasing $K$ decreases the maximum number of adversaries $T$ that can be tolerated, as shown in Thm. 4.

## 7 CONCLUSION

In this work, we introduced PICO, a collaborative learning framework with linear communication complexity, under the information-theoretic privacy setting. PICO can achieve an order of magnitude reduction in the communication overhead, while providing the same accuracy, dropout-resilience and privacy guarantees of the state-of-the-art.

## 8 ACKNOWLEDGEMENTS

This research was sponsored in part by the OUSD (R&E)/RT&L under Cooperative Agreement Number W911NF-20-2-0267, NSF CAREER Award CCF-2144927, and the UC Regents Faculty Fellowship. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies of the U.S. Government.

## REFERENCES


Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Al-Rubaie, M. and Chang, J. M. (2019). Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58.

Beerliová-Trubíniová, Z. and Hirt, M. (2008). Perfectly-secure MPC with linear communication complexity. In *Theory of Cryptography Conference*, pages 213–230. Springer.

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., and Raykova, M. (2020). Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269.

Ben-Or, M., Goldwasser, S., and Wigderson, A. (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC'88)*, page 1–10.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191.

Brinkhuis, J. and Tikhomirov, V. (2005). *Optimization: insights and applications*. Princeton University Press.

Catrina, O. and Saxena, A. (2010). Secure computation with fixed-point numbers. In *International Conference on Financial Cryptography and Data Security*, pages 35–50. Springer.

Chabanne, H., de Wargny, A., Milgram, J., Morel, C., and Prouff, E. (2017). Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, 2017:35.

Chaudhuri, K. and Monteleoni, C. (2009). Privacy-preserving logistic regression. In *Adv. in Neural Inf. Proc. Sys.*, pages 289–296.

Chen, W.-N., Choo, C. A. C., Kairouz, P., and Suresh, A. T. (2022a). The fundamental price of secure aggregation in differentially private federated learning. In *International Conference on Machine Learning*, pages 3056–3089. PMLR.

Chen, W.-N., Ozgur, A., and Kairouz, P. (2022b). The poisson binomial mechanism for unbiased federated learning with secure aggregation. In *International Conference on Machine Learning*, pages 3490–3506. PMLR.

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 168–172. IEEE.

Dalcín, L., Paz, R., and Storti, M. (2005). Mpi for python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115.

Damgård, I. and Nielsen, J. B. (2007). Scalable and unconditionally secure multiparty computation. In *Annual International Cryptology Conference*, pages 572–590. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer.

Gascón, A., Schoppmann, P., Balle, B., Raykova, M., Doerner, J., Zahur, S., and Evans, D. (2017). Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Tech.*, 2017(4):345–364.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178.

Gentry, C. and Boneh, D. (2009). *A fully homomorphic encryption scheme*, volume 20. Stanford University, Stanford.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2016). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Int. Conf. on Machine Learning*, pages 201–210.

Graepel, T., Lauter, K., and Naehrig, M. (2012). ML confidential: Machine learning on encrypted data. In *Int. Conf. on Information Security and Cryptology*, pages 1–21. Springer.

Han, K., Hong, S., Cheon, J. H., and Park, D. (2019). Logistic regression on homomorphic encrypted data at scale. *Annual Conf. on Innovative App. of Artificial Intelligence (IAAI-19)*.

Hesamifard, E., Takabi, H., and Ghasemi, M. (2017). CryptoDL: Deep neural networks over encrypted data. *arXiv:1711.05189*.

Jayaraman, B., Wang, L., Evans, D., and Gu, Q. (2018). Distributed learning without distress: Privacy-preserving empirical risk minimization. *Advances in in Neural Information Processing Systems*, pages 6346–6357.

Kairouz, P., Liu, Z., and Steinke, T. (2021). The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pages 5201–5212. PMLR.

Kedlaya, K. S. and Umans, C. (2011). Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802.

Kim, A., Song, Y., Kim, M., Lee, K., and Cheon, J. H. (2018). Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11(4):83.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

LeCun, Y., Cortes, C., and Burges, C. (2010). MNIST handwritten digit database. *http://yann. lecun. com/exdb/mnist*.

Li, P., Li, J., Huang, Z., Gao, C.-Z., Chen, W.-B., and Chen, K. (2017). Privacy-preserving outsourced classification in cloud computing. *Cluster Computing*, pages 1–10.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2018). Learning differentially private recurrent language models. In *Int. Conf. on Learning Representations*.

Mohassel, P. and Rindal, P. (2018). ABY 3: A mixed protocol framework for machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52.

Mohassel, P. and Zhang, Y. (2017). SecureML: A system for scalable privacy-preserving machine learning. In *38th IEEE Symposium on Security and Privacy*, pages 19–38. IEEE.

Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., and Taft, N. (2013). Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy*, pages 334–348.

Nosowsky, R. and Giordano, T. J. (2006). The health insurance portability and accountability act of 1996 (hipaa) privacy rule: implications for clinical research. *Annu. Rev. Med.*, 57:575–590.

Pathak, M., Rane, S., and Raj, B. (2010). Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Inf. Processing Systems*, pages 1876–1884.

Rajkumar, A. and Agarwal, S. (2012). A differentially private stochastic gradient descent algorithm for multiparty classification. In *Int. Conf. on Artificial Intelligence and Statistics (AISTATS'12)*, volume 22, pages 933–941, La Palma, Canary Islands.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

Shokri, R. and Shmatikov, V. (2015). Privacy-preserving deep learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321.

So, J., Güler, B., and Avestimehr, A. S. (2021). Codedprivateml: A fast and privacy-preserving framework for distributed machine learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):441–451.

So, J., Güler, B., and Avestimehr, S. (Dec. 2020). A scalable approach for privacy-preserving collaborative machine learning. In *Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems, NeurIPS*.

So, J., He, C., Yang, C.-S., Li, S., Yu, Q., E Ali, R., Guler, B., and Avestimehr, S. (2022). Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720.

Telenti, A. and Jiang, X. (2020). Treating medical data as a durable asset. *Nature Genetics*, 52(10):1005–1010.

Wagh, S., Gupta, D., and Chandran, N. (2018). SecureNN: Efficient and private neural network training. Cryptology ePrint Archive, Report 2018/442.

Wang, Q., Du, M., Chen, X., Chen, Y., Zhou, P., Chen, X., and Huang, X. (2018). Privacy-preserving collaborative model learning: The case of word vector training. *IEEE Trans. on Knowledge and Data Engineering*, 30(12):2381–2393.

Yao, A. C. (1982). Protocols for secure computations. In *IEEE Symp. on Foundations of Computer Science*, pages 160–164.

Yu, Q., Li, S., Raviv, N., Kalan, S. M. M., Soltanolkotabi, M., and Avestimehr, S. A. (2019). Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*, pages 1215–1225. PMLR.

Yuan, J. and Yu, S. (2014). Privacy preserving backpropagation neural network learning made practical with cloud computing. *IEEE Trans. on Parallel and Dist. Sys.*, 25(1):212–221.

Zhao, Y. and Sun, H. (2021). Information theoretic secure aggregation with user dropouts. In *IEEE International Symposium on Information Theory, ISIT'21*.

## SUPPLEMENTARY MATERIAL

# A SHAMIR'S SECRET SHARING AND CRYPTOGRAPHIC PRIMITIVES

In PICO, all secret shares are generated using Shamir's secret sharing. Shamir's $T$-out-of-$N$ secret sharing protocol (Shamir, 1979) embeds a secret $x$ in a degree $T$ polynomial,

$$f(\zeta) = x + \zeta r_1 + \ldots + \zeta^T r_T \tag{36}$$

where each coefficient $\{r_k\}_{k\in[T]}$ is generated uniformly at random from $\mathbb{F}_q$. Then, each client $i \in [N]$ receives a secret share $f(\alpha_i) \triangleq [x]_i$, which corresponds to the polynomial $f(\zeta)$ evaluated at $\zeta = \alpha_i$. The secret $x$ can be reconstructed perfectly from any collection of $T+1$ secret shares using polynomial interpolation, as any polynomial of degree $T$ can be perfectly reconstructed from at least $T+1$ evaluation points. On the other hand, no information (in an information-theoretic sense) can be revealed about the secret $x$ from any group of $T$ or fewer shares. As such, Shamir's secret sharing preserves the information-theoretic privacy of $x$ against any collusions between up to $T$ clients.

Shamir's secret sharing supports addition and multiplication operations. Consider two variables $x$ and $x'$, where client $i \in [N]$ holds the secret shares $[x]_i$ and $[x']_i$, respectively. Then, addition and multiplication operations proceed as follows.

**Addition.** In order to compute a summation $x + x'$, client $i$ locally adds the secret shares $[x]_i$ and $[x']_i$:

$$[x]_i + [x']_i = (x + \alpha_i r_1 + \ldots + \alpha_i^T r_T) + (x' + \alpha_i r'_1 + \ldots + \alpha_i^T r'_T) \tag{37}$$

$$= (x + x') + \alpha_i(r_1 + r'_1) + \ldots + \alpha_i^T(r_T + r'_T) \tag{38}$$

$$\triangleq [x + x']_i \tag{39}$$

hence the resulting polynomial is also Shamir's secret sharing of the sum $x + x'$, masked by the random vectors $(r_1 + r'_1), \ldots, (r_T + r'_T)$, respectively. As such, Shamir's secret sharing satisfies a *computative property* over the addition function, i.e., the summation of two secret shares $[x]_i + [x']_i$ results in a secret share $[x + x']_i$ of the summation $x + x'$. This operation requires no communication across the clients. The final result $x + x'$ can be recovered by collecting the secret shares $[x + x']_i$ from any set of at least $T + 1$ clients, and using polynomial interpolation. This requires the total number of clients to be $N \geq T + 1$.

**Multiplication by a public constant.** In order to multiply a secret $x$ with a public constant $c$, each client locally multiplies its secret share $[x]_i$ with $c$, and obtains a secret share $[cx]_i$ of the resulting value $cx$, which follows from the fact that $c[x]_i = [cx]_i$. This operation requires no communication between the clients.

**Multiplication.** In order to compute a product $xx'$, client $i$ locally multiplies the secret shares:

$$[x]_i \times [x']_i = (x + \alpha_i r_1 + \ldots + \alpha_i^T r_T) \times (x' + \alpha_i r'_1 + \ldots + \alpha_i^T r'_T) \tag{40}$$

$$= xx' + \alpha_i(xr'_1 + x'r_1) + \ldots + \alpha_i^{2T}(r_T r'_T) \tag{41}$$

where the resulting polynomial now has degree $2T$. As a result, to recover the final result $xx'$, polynomial interpolation requires collecting the secret shares from at least $2T + 1$ clients. This requires the minimum number of clients to satisfy $N \geq 2T + 1$. Successive multiplication operations (such as successive training rounds) will further increase the degree (hence the minimum number of clients required), which is a major limitation. To avoid the degree explosion, after each multiplication round, clients have to carry out a degree reduction step to reduce the degree of the shares from (40), where new shares are generated corresponding to a polynomial of degree $T$ (Ben-Or et al., 1988). The main intuition behind the degree reduction step is the observation that (40) represents a polynomial of degree $2T$, hence the true product $xx'$ can be written as a linear function of $2T + 1$ secret shares from (40), i.e., $xx' = \sum_{i=1}^{2T+1} \lambda_i \langle xx' \rangle_i$ for some coefficients $\lambda_i$, where $\langle xx' \rangle_i \triangleq [x]_i[x']_i$. To perform degree reduction, each client then generates and distributes a $T$-out-of-$N$ secret share of $\langle xx' \rangle_i$, using Shamir's secret sharing, where client $i \in [N]$ sends a secret share $[\langle xx' \rangle_i]_j$ to client $j \in [N]$. Using the received secret shares, client $j$ then computes a new secret share $[xx']_j = \sum_{i=1}^{2T+1} \lambda_i [\langle xx' \rangle_i]_j = [\sum_{i=1}^{2T+1} \lambda_i \langle xx' \rangle_i]_j$ of $xx'$, where the final secret share $[xx']_j$ corresponds to a polynomial of degree $T$ (as opposed to 2T). On the other hand, this step has a quadratic communication overhead ($O(N^2)$ across the $N$ clients).

# B ALGORITHM

The offline and online procedures of PICO are presented in Algorithm 1 and Algorithm 2, respectively. The offline phase consists of randomness generation across the $N$ users, which will later be used for masking the datasets, models, and computations in the online phase. For clarity, the random vectors generated at round $t$ are identified with a superscript $(t)$.

**Algorithm 1:** PICO - Offline Phase

**Input:** Number of clients $N$, polynomial coefficients $(\alpha_1, \ldots, \alpha_N), (\beta_1, \ldots, \beta_K)$.

**Output:** Random masks $\{\widetilde{\mathbf{R}}_{ij}\}_{i,j\in[N]}$, $\{[\mathbf{a}_i]_j\}_{i,j\in[N]}$, $\{\widetilde{\mathbf{r}}_i^{(t)}, [\mathbf{r}^{(t)}]_i, [\mathbf{u}_i^{(t)}]_j\}_{i\in[N],t\in\{0,\ldots,J-1\}}$, random initial model $\{[\mathbf{w}^{(0)}]_i\}_{i\in[N]}$.

// **1. Dataset Encoding**

**for** *client* $i = 1, \ldots, N$ **do**
- Encode the random matrices $\{\mathbf{R}_{ik}\}_{k\in[K]}$, $\{\mathbf{V}_{ik}\}_{k\in\{K+1,\ldots,K+T\}}$ from (4).
- **for** $j = 1, \ldots, N$ **do**
  - Send the encoded matrix $\widetilde{\mathbf{R}}_{ij}$ to client $j$.

// **2. Label Encoding**

**for** *client* $i = 1, \ldots, N$ **do**
- Encode the random vectors $\{\mathbf{a}_{ik}\}_{k\in[K]}$, $\{\mathbf{b}_{ik}\}_{k\in\{K+1,\ldots,K+T\}}$ from (9).
- **for** $j = 1, \ldots, N$ **do**
  - Send the encoded vector $\widetilde{\mathbf{a}}_{ij}$ to client $j$.
  - Send a secret share $[\mathbf{a}_i]_j$ to client $j$ using Shamir's secret sharing.

**for** $i = 1, \ldots, N$ **do**
- Construct a larger dimensional encoded vector $\tilde{\mathbf{a}}_i \triangleq (\mathbf{M} \otimes \mathbf{I}) \times (\widetilde{\mathbf{a}}_{1i}^T, \ldots, \widetilde{\mathbf{a}}_{Ni}^T)^T$ from (10).
- Construct a larger dimensional secret share $[\mathbf{a}_k]_i \triangleq \mathbf{M} \times ([\mathbf{a}_{1k}]_i^T, \ldots, [\mathbf{a}_{Nk}]_i^T)^T$ for all $k \in [K]$.

// **3. Model Initialization**

**for** *client* $i = 1, \ldots, N$ **do**
- Generate a random vector $\mathbf{w}_i^{(0)}$ from $\mathbb{F}_q$.
- **for** $j = 1, \ldots, N$ **do**
  - Send a secret share $[\mathbf{w}_i^{(0)}]_j$ to client $j$ using Shamir's secret sharing.

**for** *client* $i = 1, \ldots, N$ **do**
- Initialize the model $[\mathbf{w}^{(0)}]_i$ using $\{[\mathbf{w}_j^{(0)}]_i\}_{j\in[N]}$ as given in (16).

**for** *iteration* $t = 0, \ldots, J-1$ **do**

// **4. Model Encoding**

- **for** *client* $i = 1, \ldots, N$ **do**
  - Encode the random vectors $\mathbf{r}_i^{(t)}$, $\{\mathbf{v}_{ik}^{(t)}\}_{k\in\{K+1,\ldots,K+T\}}$ as in (19).
  - **for** $j = 1, \ldots, N$ **do**
    - Send the encoded vector $\widetilde{\mathbf{r}}_{ij}^{(t)}$ to client $j$.
    - Send a secret share $[\mathbf{r}_i^{(t)}]_j$ to client $j$ using Shamir's secret sharing.
- **for** *client* $i = 1, \ldots, N$ **do**
  - Compute the coded vector, $\widetilde{\mathbf{r}}_i^{(t)}$ as given in (20).
  - Compute the secret share $[\mathbf{r}^{(t)}]_i$ after receiving $\{[\mathbf{r}_j^{(t)}]_i\}_{j\in[N]}$ as given in (21).

// **5. Gradient Computing and Model Update**

- **for** *client* $i = 1, \ldots, N$ **do**
  - Encode $\{\mathbf{u}_{ik}^{(t)}\}_{k\in(2r+1)(K+T-1)+1}$ as given in (27).
  - **for** $j = 1, \ldots, N$ **do**
    - Send the encoded vector $\widetilde{\mathbf{u}}_{ij}^{(t)}$ to client $j$.
    - Send a secret share $[\sum_{k\in[K]} \mathbf{u}_{ik}^{(t)}]_j$ to client $j$ using Shamir's secret sharing.
- **for** *client* $i = 1, \ldots, N$ **do**
  - Compute the coded vector, $\widetilde{\mathbf{u}}_i^{(t)}$ after receiving $\{\widetilde{\mathbf{u}}_{ji}^{(t)}\}_{j\in[N]}$ as given in (29).
  - Compute the secret share, $[\sum_{k\in[K]} \mathbf{u}_k^{(t)}]_i$ after receiving $\{[\sum_{k\in[K]} \mathbf{u}_{jk}^{(t)}]_i\}_{j\in[N]}$ from (31).

# C FLOWCHART OF PICO

Figure 10 demonstrates the generated offline/online variables for each component in PICO, along with the corresponding relations between the offline and online variables. The offline phase of any component is executed independently from the offline and online phases of other components.

**Algorithm 2:** PICO - Online Phase

**Input:** Dataset $(\mathcal{X}, \mathbf{y}) = ((\mathcal{X}_1, \mathbf{y}_1), \ldots, (\mathcal{X}_N, \mathbf{y}_N))$ distributed over $N$ clients.

**Output:** Model parameters $\mathbf{w}^{(J)}$ after $J$ training rounds.

// **1. Dataset Encoding**

**for** *client* $i = 1, \ldots, N$ **do**
  Partition the dataset $\mathcal{X}_i$ into $K$ equal-sized shards $(\mathbf{X}_{i1}, \ldots, \mathbf{X}_{iK})$.
  Broadcast the masked dataset $\hat{\mathbf{X}}_{ik} = \mathbf{X}_{ik} - \mathbf{R}_{ik}$ for $k \in [K]$.
**for** *client* $i = 1, \ldots, N$ **do**
  Generate the coded dataset $\widetilde{\mathbf{X}}_i$ from (7).

// **2. Label Encoding**

**for** *client* $i = 1, \ldots, N$ **do**
  Partition $\mathbf{y}_i = \left( \sum_{l \in \mathcal{X}_i} \mathbf{x}_l^T y_l \right)$ into K equal-sized shards $(\mathbf{y}_{i1}, \ldots, \mathbf{y}_{iK})$.
  **for** *client* $j = 1, \ldots, N$ **do**
    Encode $\{\mathbf{y}_{ik}\}_{k \in [K]}$ as described in (13), and send the encoded vector $\widetilde{\mathbf{y}}_{ij}$ to client $j$.
**for** *client* $i = 1, \ldots, N$ **do**
  Broadcast $\hat{\mathbf{a}}_i = \sum_{j \in [N]} \widetilde{\mathbf{y}}_{ji} - \widetilde{\mathbf{a}}_i$ from (14).
**for** *client* $i = 1, \ldots, N$ **do**
  Reconstruct $\sum_{j \in [N]} \mathbf{y}_{jk} - \mathbf{a}_i$ for all $k \in [K]$ using polynomial interpolation.
  Compute a secret share $[\mathbf{X}^T \mathbf{y}]_i$ of $\mathbf{X}^T \mathbf{y}$ as given in (15).
**for** *iteration* $t = 0, \ldots, J-1$ **do**
  // **4. Model Encoding**
  **for** $i = 1, \ldots, N$ **do**
    Broadcast $[\widehat{\mathbf{w}}^{(t)}]_i$ from (22).
  **for** $i = 1, \ldots, N$ **do**
    Decode $\widehat{\mathbf{w}}^{(t)} \triangleq \mathbf{w}^{(t)} - \mathbf{r}^{(t)}$ using polynomial interpolation.
    Compute the coded model $\widetilde{\mathbf{w}}_i^{(t)}$ in (24).
  // **5. Gradient Computing and Model Update**
  **for** *client* $i = 1, \ldots, N$ **do**
    Compute the gradient $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)}) = \widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)})$ in (32).
    Broadcast $\hat{\mathbf{u}}_i^{(t)} = \widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)}) - \mathbf{u}_i^{(t)}$.
  **for** *client* $i = 1, \ldots, N$ **do**
    Decode $\psi(\beta_k) = h(\beta_k) - \phi(\beta_k) = \mathbf{X}_k^T \hat{g}(\mathbf{X}_k \times \mathbf{w}^{(t)}) - \mathbf{u}_k^{(t)}$ for $k \in [K]$ via polynomial interpolation.
    Compute a secret share $[\mathbf{X}^T \hat{g}(\mathbf{X} \times \mathbf{w}^{(t)})]_i$ of the gradient $\mathbf{X}^T \hat{g}(\mathbf{X} \times \mathbf{w}^{(t)})$ as given in (33).
    Update the model with $[\mathbf{w}^{(t+1)}]_i$ from (34).

// **Final Model Recovery**

Collect the secret shares $[\mathbf{w}^{(J)}]_i$ from any $T+1$ clients.
Decode the final model $\mathbf{w}^{(J)}$ via polynomial interpolation.

# D INFORMATION-THEORETIC PRIVACY

*Proof.* For tractability of theoretical analysis, in this section we consider a sufficiently large field size, and assume all computations are performed in the domain of integers. Consider an arbitrary set of adversaries $\mathcal{T} \subseteq N$. For ease of exposition, we focus on the worst case scenario by setting $|\mathcal{T}| = T$, while noting that the same analysis holds for all $|\mathcal{T}| < T$. Let $\mathcal{M}_{\mathcal{T}}^1$ and $\mathcal{M}_{\mathcal{T}}^2$, denote the collection of all messages received by the adversaries during the dataset encoding (Stage 1), and label encoding (Stage 2) stages, respectively. Let $\mathcal{M}_{\mathcal{T}}^3$ denote the collection of all messages received by the adversaries during model initialization stage (Stage 3). Similarly, let $\mathcal{M}_{\mathcal{T}}^{4,t}$ denote the collection of all messages received

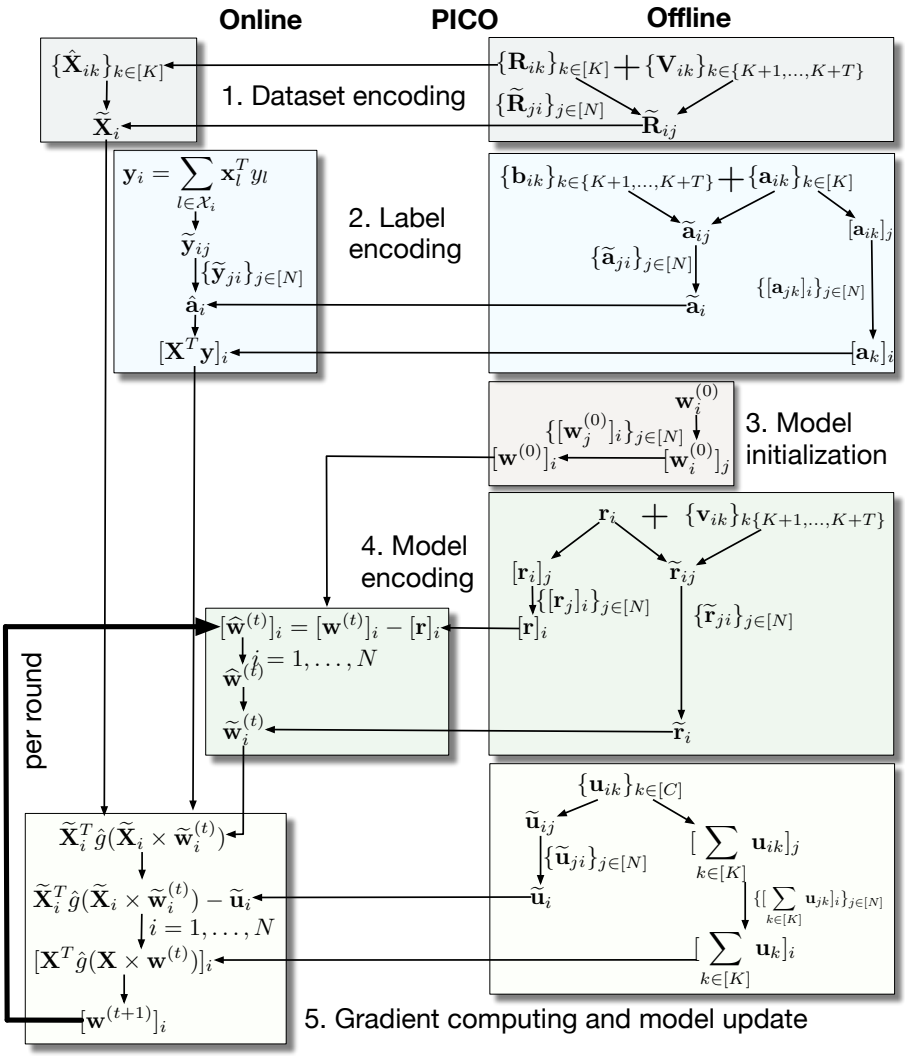



Figure 10: Detailed flowchart of PICO, for each client $i \in [N]$.

by the adversaries in model encoding stage (Stage 4) at training round $t \in \{0, \ldots, J-1\}$. Finally, let $\mathcal{M}_{\mathcal{T}}^{5,t}$ denote the collection of all messages received by the adversaries in gradient computing and model update stage (Stage 5) at training round $t \in \{0, \ldots, J-1\}$. Then, from the chain rule of mutual information (Cover and Thomas, 1999), one can rewrite the mutual information condition from (35) as follows:

$$
\begin{aligned}
& I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (42)\\
&= I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \mathcal{M}_{\mathcal{T}}^3, \cup_{t\in[J]}\mathcal{M}_{\mathcal{T}}^{4,t}, \cup_{t\in[J]}\mathcal{M}_{\mathcal{T}}^{5,t} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (43)\\
&= I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^1 | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})\\
&\quad + I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^2 | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})\\
&\quad + I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^3 | \mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})\\
&\quad + \sum_{t=0}^{J-1} I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^{4,t} | \mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \mathcal{M}_{\mathcal{T}}^3, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l}, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l}, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})\\
&\quad + \sum_{t=0}^{J-1} I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^{5,t} | \mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \mathcal{M}_{\mathcal{T}}^3, \cup_{l=0}^{t}\mathcal{M}_{\mathcal{T}}^{4,l}, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l}, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (44)
\end{aligned}
$$

We next investigate the individual terms in the summation (44).

**Stage 1: Dataset Encoding.** First, we start with the first term in (44), which corresponds to Stage 1 of PICO, i.e., encoding the datasets. From the description of Stage 1, the first term in the right hand side of (44) can be written as:

$$I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}^1|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{45}$$

$$= I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{46}$$

$$= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$- H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{47}$$

We next bound the first term in (149) as follows:

$$H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$

$$= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in\mathcal{H}\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{48}$$

$$\leq H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in\mathcal{H}\\ k\in[K]}}) \tag{49}$$

$$\leq \log\Big(q^{(\sum_{i\in\mathcal{H}}\frac{Tdm_i}{K})+(\sum_{i\in\mathcal{T}}dm_i)+(\sum_{i\in\mathcal{T}}\frac{Tdm_i}{K})+(\sum_{i\in\mathcal{H}}dm_i)}\Big) \tag{50}$$

$$= d\Big(\frac{T}{K}+1\Big)\Big(\sum_{i\in[N]}m_i\Big)\log q \tag{51}$$

where (49) holds since conditioning cannot increase entropy. Equation (50) follows from the fact that uniform distribution maximizes entropy, and that the entropy of a uniform random variable distributed over an alphabet $\mathcal{A}$ is equal to $\log|\mathcal{A}|$. For the second term in (149), we observe that:

$$H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{52}$$

$$= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{53}$$

$$= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}},\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}) \tag{54}$$

$$= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}|\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}})$$
$$+ H(\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}|\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}) + H(\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}) \tag{55}$$

$$= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}|\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}},\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}})$$
$$+ H(\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}) + H(\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}) \tag{56}$$

$$= H\Big(\Big\{\sum_{k=K+1}^{K+T}\mathbf{V}_{ik}\prod_{l\in[K+T]\setminus\{k\}}\frac{\alpha_j-\beta_l}{\beta_k-\beta_l}\Big\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}\Big) + \log(q^{Td\sum_{i\in\mathcal{T}}\frac{m_i}{K}}) + \log(q^{Kd\sum_{i\in[N]}\frac{m_i}{K}}) \tag{57}$$

$$= \sum_{i\in\mathcal{H}}H\Big(\Big\{\sum_{k=K+1}^{K+T}\mathbf{V}_{ik}\prod_{l\in[K+T]\setminus\{k\}}\frac{\alpha_j-\beta_l}{\beta_k-\beta_l}\Big\}_{j\in\mathcal{T}}\Big) + \frac{Td}{K}(\sum_{i\in\mathcal{T}}m_i)\log q + d(\sum_{i\in[N]}m_i)\log q \tag{58}$$

$$= \sum_{i\in\mathcal{H}}H(\{\mathbf{Z}_{ij}\}_{j\in\mathcal{T}}) + \frac{Td}{K}(\sum_{i\in\mathcal{T}}m_i)\log q + d(\sum_{i\in[N]}m_i)\log q \tag{59}$$

where (53) holds since given $\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]}$, there is no uncertainty remaining in $\{X_{ik}\}_{i\in[N],k\in[K]}$, (54) holds since the generated randomness is independent from the local datasets, (55) follows from the chain rule of entropy, (56) holds since the random matrices are generated independently where each element is distributed uniformly at random (and independent

from other elements) from the finite field $\mathbb{F}_q$. In (59), we define:

$$\mathbf{Z}_{ij} \triangleq \sum_{k=K+1}^{K+T} \mathbf{V}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l} \tag{60}$$

for all $i \in \mathcal{H}$ and $j \in \mathcal{T}$. In the following, we let the first N-T users be honest (the last T users are adversarial), i.e., $\mathcal{H} = [N-T]$ and $\mathcal{T} = \{N-T+1, \ldots, N\}$. The sole purpose of this assumption is notational simplicity, and the same analysis holds for any set of adversarial users $\mathcal{T}$ of size $T$. We also represent the Lagrange polynomial coefficients as:

$$\rho_{jk} \triangleq \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l} \tag{61}$$

for all $j \in [N]$ and $k \in [K+T]$. Then, from (60), one can write:

$$(\mathbf{Z}_{i,N-T+1}, \ldots, \mathbf{Z}_{i,N}) = (\mathbf{V}_{i,K+1}, \ldots, \mathbf{V}_{i,K+T}) \underbrace{\begin{bmatrix} \rho_{N-T+1,K+1} & \cdots & \rho_{N,K+1} \\ \vdots & \ddots & \vdots \\ \rho_{N-T+1,K+T} & \cdots & \rho_{N,K+T} \end{bmatrix}}_{\mathbf{\Gamma}} \tag{62}$$

where $\mathbf{\Gamma}$ is a $T \times T$ MDS matrix (hence is invertible), which follows from the MDS property of Lagrange polynomials as shown in Yu et al. (2019). An MDS matrix guarantees that (62) is a bijective mapping, in other words every $(\mathbf{V}_{i,K+1}, \ldots, \mathbf{V}_{i,K+T})$ maps to a unique $(\mathbf{Z}_{i,N-T+1}, \ldots, \mathbf{Z}_{i,N})$. As a result,

$$\begin{aligned} H(\{\mathbf{Z}_{ij}\}_{j\in\mathcal{T}}) &= H(\mathbf{Z}_{i,N-T+1}, \ldots, \mathbf{Z}_{i,N}) && (63)\\ &= H(\mathbf{V}_{i,K+1}, \ldots, \mathbf{V}_{i,K+T}) && (64)\\ &= \frac{Tdm_i}{K} \log q && (65) \end{aligned}$$

where (64) follows from (62) and that $\mathbf{\Gamma}$ is an MDS matrix, and (65) follows from the fact that each element of $\mathbf{V}_{ik}$ is distributed uniformly at random over the finite field $\mathbb{F}_q$. By combining (65) with (59), we have:

$$\begin{aligned} &H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)})\\ &= (\sum_{i\in\mathcal{H}} \frac{Tdm_i}{K} \log q) + \frac{Td}{K}(\sum_{i\in\mathcal{T}} m_i) \log q + d(\sum_{i\in\mathcal{T}} m_i) \log q && (66)\\ &= d\Big(\frac{T}{K} + 1\Big)\Big(\sum_{i\in[N]} m_i\Big) \log q && (67) \end{aligned}$$

Finally, by combining (50) and (67) with (149), we have:

$$\begin{aligned} 0 &\le I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^1 | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (68)\\ &= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})\\ &\quad - H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}} | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) && (69)\\ &\le d\Big(\frac{T}{K} + 1\Big)\Big(\sum_{i\in[N]} m_i\Big) \log q - d\Big(\frac{T}{K} + 1\Big)\Big(\sum_{i\in[N]} m_i\Big) \log q && (70)\\ &= 0 && (71) \end{aligned}$$

where the first inequality follows from the non-negativity of mutual information. Therefore, the first term in (44) satisfies the following:

$$I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^1 | \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) = 0 \tag{72}$$

**Stage 2: Label Encoding.** We next consider the second term in (44), which corresponds to the secret sharing of the labels. Without loss of generality, we represent the secret share of $\mathbf{a}_{ik}$ from user $i$ to user $j$ as follows:

$$[\mathbf{a}_{ik}]_j \triangleq \mathbf{a}_{ik} + \sum_{l\in[T]} \gamma_j^l \mathbf{z}_{ikl} \tag{73}$$

where $\mathbf{z}_{ikl}$ are random vectors of size $\frac{d}{K}$, where each element is distributed independently and uniformly at random from $\mathbb{F}_q$. Coefficients $\{\gamma_i\}_{i\in[N]}$ are distinct public parameters agreed in advance between all $N$ users, where $\gamma_i \in \mathbb{F}_q$ for all $i \in [N]$ such that $\{\gamma_i\}_{i\in[N]} \cap \{\beta_k\}_{k\in[K+T]} \cap \{\alpha_j\}_{j\in[N]} = \emptyset$. Using (73), we can rewrite the second term in (44) as follows:

$$\begin{aligned}
&I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^2 | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (74)\\
&= I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \{\hat{\mathbf{a}}_i\}_{i\in[N]}, \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}}\\
&\quad |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (75)\\
&= I(\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{H}}; \{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \{\sum_{j\in[N]} \mathbf{y}_{jk} - \mathbf{a}_k\}_{k\in[K]}, \{\sum_{j\in[N]} \mathbf{r}_{jk} - \mathbf{b}_k\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (76)\\
&= H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \{\sum_{j\in[N]} \mathbf{y}_{jk} - \mathbf{a}_k\}_{k\in[K]}, \{\sum_{j\in[N]} \mathbf{r}_{jk} - \mathbf{b}_k\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) && (77)\\
&- H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \{\sum_{j\in[N]} \mathbf{y}_{jk} - \mathbf{a}_k\}_{k\in[K]}, \{\sum_{j\in[N]} \mathbf{r}_{jk} - \mathbf{b}_k\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) && (78)
\end{aligned}$$

where (76) follows from the fact that any polynomial of degree $K + T - 1$ can be determined from at least $K + T$ evaluation points, therefore there is a bijective mapping from any feasible set $\{\hat{\mathbf{a}}_i\}_{i\in[N]}$ to a set of $K + T$ coefficients $\{\sum_{j\in[N]} \mathbf{y}_{jk} - \mathbf{a}_k\}_{k\in[K]}, \{\sum_{j\in[N]} \mathbf{r}_{jk} - \mathbf{b}_k\}_{k\in\{K+1,\ldots,K+T\}}$. As such, the uncertainty in the former is equal to the latter.

For the second term in (78), we find that:

$$\begin{aligned}
&H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \{\sum_{j\in[N]} \mathbf{y}_{jk} - \mathbf{a}_k\}_{k\in[K]}, \{\sum_{j\in[N]} \mathbf{r}_{jk} - \mathbf{b}_k\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) && (79)\\
&= H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \{\mathbf{a}_k\}_{k\in[K]}, \{\sum_{j\in[N]} \mathbf{r}_{jk} - \mathbf{b}_k\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) && (80)\\
&= H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \left\{(\mathbf{M}\otimes\mathbf{I})\begin{bmatrix}\mathbf{a}_{1k}\\ \vdots \\ \mathbf{a}_{Nk}\end{bmatrix}\right\}_{k\in[K]}, \left\{\sum_{j\in[N]} \mathbf{r}_{jk} - (\mathbf{M}\otimes\mathbf{I})\begin{bmatrix}\mathbf{b}_{1k}\\ \vdots \\ \mathbf{b}_{Nk}\end{bmatrix}\right\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) && (81)\\
&= H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \left\{(\overline{\mathbf{M}}\otimes\mathbf{I})\begin{bmatrix}\mathbf{a}_{1k}\\ \vdots \\ \mathbf{a}_{(N-T)k}\end{bmatrix}\right\}_{k\in[K]}, \left\{\sum_{j\in[N-T]} \mathbf{r}_{jk} - (\overline{\mathbf{M}}\otimes\mathbf{I})\begin{bmatrix}\mathbf{b}_{1k}\\ \vdots \\ \mathbf{b}_{(N-T)k}\end{bmatrix}\right\}_{k\in\{K+1,\ldots,K+T\}},\\
&\quad \{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T}, k'\in[K], l\in[T]\\ k\in\{K+1,\ldots,K+T\}}} |\mathcal{M}_{\mathcal{T}}^1, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) && (82)\\
&= H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H}, j\in\mathcal{T}, k\in[K]}, \left\{\begin{bmatrix}\mathbf{a}_{1k}\\ \vdots \\ \mathbf{a}_{(N-T)k}\end{bmatrix}\right\}_{k\in[K]}, \left\{\sum_{j\in[N-T]} \mathbf{r}_{jk} - (\overline{\mathbf{M}}\otimes\mathbf{I})\begin{bmatrix}\mathbf{b}_{1k}\\ \vdots \\ \mathbf{b}_{(N-T)k}\end{bmatrix}\right\}_{k\in\{K+1,\ldots,K+T\}})
\end{aligned}$$

$$+ H(\{\mathbf{r}_{ik}, \mathbf{b}_{ik}, \mathbf{a}_{ik'}, \mathbf{z}_{ik'l}\}_{i\in\mathcal{T},k'\in[K],l\in[T],k\in\{K+1,\ldots,K+T\}}) \tag{83}$$

$$= H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]}, \{\mathbf{a}_{jk}\}_{j\in[N-T],k\in[K]}, \{\sum_{j\in[N-T]} \mathbf{r}_{jk} - (\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\}_{k\in\{K+1,\ldots,K+T\}})$$

$$+ Td\Big(\frac{T}{K} + \frac{T}{K(N-T)} + \frac{1}{N-T} + \frac{T}{N-T}\Big)\log q \tag{84}$$

where (80) follows from the fact that given $\{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}$, there is uncertainty in $\sum_{j\in[N]}\mathbf{y}_{jk}$ for all $k\in[K]$. In (82), we define the following square submatrix of $\mathbf{M}$ from (12),

$$\overline{\mathbf{M}} \triangleq \begin{bmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{N-T-1} \\ 1 & \lambda_2 & \ldots & \lambda_2^{N-T-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{N-T} & \ldots & \lambda_{N-T}^{N-T-1} \end{bmatrix} \tag{85}$$

which is an $(N-T)\times(N-T)$ MDS matrix (hence is invertible), from which (83) follows. Equation (84) follows from the entropy of uniform random variables, and,

$$\overline{\mathbf{b}}_k \triangleq \begin{bmatrix} \mathbf{b}_{1k} \\ \vdots \\ \mathbf{b}_{(N-T)k} \end{bmatrix} \tag{86}$$

For the first term in (84),

$$H(\{\widetilde{\mathbf{a}}_{ij}, [\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]}, \{\mathbf{a}_{jk}\}_{j\in[N-T],k\in[K]}, \{\sum_{j\in[N-T]} \mathbf{r}_{jk} - (\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\}_{k\in\{K+1,\ldots,K+T\}}) \tag{87}$$

$$= H(\{\widetilde{\mathbf{a}}_{ij}\}_{i\in\mathcal{H},j\in\mathcal{T}}, \{\sum_{j\in[N-T]} \mathbf{r}_{jk} - (\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\}_{k\in\{K+1,\ldots,K+T\}} | \{[\mathbf{a}_{ik}]_j\}_{\substack{i\in\mathcal{H},j\in\mathcal{T},\\ k\in[K]}}, \{\mathbf{a}_{jk}\}_{\substack{j\in[N-T],\\ k\in[K]}})$$

$$+ H(\{[\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]} | \{\mathbf{a}_{jk}\}_{j\in[N-T],k\in[K]}) + H(\{\mathbf{a}_{jk}\}_{j\in[N-T],k\in[K]}) \tag{88}$$

$$= H\left(\Big\{\sum_{k=K+1}^{K+T} \mathbf{b}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j - \beta_l}{\beta_k - \beta_l}\Big\}_{i\in\mathcal{H},j\in\mathcal{T}}, \Big\{\sum_{j\in[N-T]} \mathbf{r}_{jk} - (\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\Big\}_{k\in\{K+1,\ldots,K+T\}}\right)$$

$$+ H\left(\{\sum_{l\in[T]} \gamma_j^l \mathbf{z}_{ikl}\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]}\right) + H(\{\mathbf{a}_{jk}\}_{j\in[N-T],k\in[K]}) \tag{89}$$

where (88) follows from the chain rule of entropy, and (89) holds since the random vectors are generated independently.

We next introduce the following variables to simplify the analysis of (89). First, we let,

$$(\sum_{l\in[T]} \gamma_{N-T+1}^l \mathbf{z}_{ikl}, \ldots, \sum_{l\in[T]} \gamma_N^l \mathbf{z}_{ikl}) = \underbrace{(\mathbf{z}_{ik1}, \ldots, \mathbf{z}_{ikT})}_{\mathbf{z}_{ik}} \underbrace{\begin{bmatrix} \gamma_{N-T+1}^1 & \cdots & \gamma_N^1 \\ \vdots & \ddots & \vdots \\ \gamma_{N-T+1}^T & \cdots & \gamma_N^T \end{bmatrix}}_{\mathbf{A}} \tag{90}$$

where $\mathbf{A}$ is an $T\times T$ MDS matrix, hence is invertible. From (90), it then follows for the second term in (89) that,

$$H\left(\{\sum_{l\in[T]} \gamma_j^l \mathbf{z}_{ikl}\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]}\right) \tag{91}$$

$$= \sum_{i\in\mathcal{H}}\sum_{k\in[K]} H(\{\sum_{l\in[T]} \gamma_j^l \mathbf{z}_{ikl}\}_{j\in\mathcal{T}}) \tag{92}$$

$$= \sum_{i\in[N-T]}\sum_{k\in[K]} H(\{\sum_{l\in[T]} \gamma_j^l \mathbf{z}_{ikl}\}_{j\in\{N-T+1,\ldots,N\}}) \tag{93}$$

$$= \sum_{i\in[N-T]}\sum_{k\in[K]} H(\mathbf{z}_{ik}\mathbf{A}) \tag{94}$$

$$= \sum_{i\in[N-T]} \sum_{k\in[K]} H(\mathbf{z}_{ik}) \tag{95}$$

$$= (N-T)KT\frac{d}{(N-T)K}\log q \tag{96}$$

$$= Td\log q \tag{97}$$

where (92) follows from the independence of the generated random variables, (94) follows from (90), and (95) holds since matrix $\mathbf{A}$ is invertible, hence represents a bijective relationship between $\mathbf{z}_{ik}$ and $\mathbf{z}_{ik}\mathbf{A}$. Finally, (96) follows from the entropy of uniform random variables. Similarly, for the last term in (89), we find that,

$$H(\{\mathbf{a}_{jk}\}_{j\in[N-T],k\in[K]}) = (N-T)K\frac{d}{(N-T)K}\log q = d\log q \tag{98}$$

which also follows from the entropy of uniform random variables.

For the first term in (89), to rewrite $\left\{\sum_{k=K+1}^{K+T}\mathbf{b}_{ik}\prod_{l\in[K+T]\setminus\{k\}}\frac{\alpha_j-\beta_l}{\beta_k-\beta_l}\right\}_{j\in\mathcal{T}}$ as:

$$\left(\sum_{k=K+1}^{K+T}\mathbf{b}_{ik}\prod_{l\in[K+T]\setminus\{k\}}\frac{\alpha_{N-T+1}-\beta_l}{\beta_k-\beta_l},\ldots,\sum_{k=K+1}^{K+T}\mathbf{b}_{ik}\prod_{l\in[K+T]\setminus\{k\}}\frac{\alpha_N-\beta_l}{\beta_k-\beta_l}\right)$$
$$= (\mathbf{b}_{i,K+1},\ldots,\mathbf{b}_{i,K+T})\underbrace{\begin{bmatrix}\rho_{N-T+1,K+1} & \cdots & \rho_{N,K+1}\\ \vdots & \ddots & \vdots\\ \rho_{N-T+1,K+T} & \cdots & \rho_{N,K+T}\end{bmatrix}}_{\mathbf{\Gamma}} \tag{99}$$

where $\rho_{i,k}$ is as defined in (61), and $\mathbf{\Gamma}$ is an $T\times T$ MDS matrix from (62) (hence is invertible). Using (99), we rewrite the first term in (89) as:

$$H\left(\left\{\sum_{k=K+1}^{K+T}\mathbf{b}_{ik}\prod_{l\in[K+T]\setminus\{k\}}\frac{\alpha_j-\beta_l}{\beta_k-\beta_l}\right\}_{i\in\mathcal{H},j\in\mathcal{T}},\left\{\sum_{j\in[N-T]}\mathbf{r}_{jk}-(\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\right\}_{k\in\{K+1,\ldots,K+T\}}\right) \tag{100}$$

$$= H\left(\{(\mathbf{b}_{i,K+1},\ldots,\mathbf{b}_{i,K+T})\mathbf{\Gamma}\}_{i\in\mathcal{H}},\left\{\sum_{j\in[N-T]}\mathbf{r}_{jk}-(\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\right\}_{k\in\{K+1,\ldots,K+T\}}\right) \tag{101}$$

$$= H\left(\{(\mathbf{b}_{i,K+1},\ldots,\mathbf{b}_{i,K+T})\}_{i\in\mathcal{H}},\left\{\sum_{j\in[N-T]}\mathbf{r}_{jk}-(\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\right\}_{k\in\{K+1,\ldots,K+T\}}\right) \tag{102}$$

$$= H\left(\{\mathbf{b}_{i,k}\}_{i\in\mathcal{H},k\in\{K+1,\ldots,K+T\}},\left\{\sum_{j\in[N-T]}\mathbf{r}_{jk}-(\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\right\}_{k\in\{K+1,\ldots,K+T\}}\right) \tag{103}$$

$$= H\left(\left\{\sum_{j\in[N-T]}\mathbf{r}_{jk}-(\overline{\mathbf{M}}\otimes\mathbf{I})\,\overline{\mathbf{b}}_k\right\}_{k\in\{K+1,\ldots,K+T\}}\middle|\{\mathbf{b}_{i,k}\}_{i\in[N-T],k\in\{K+1,\ldots,K+T\}}\right)$$
$$+ H(\{\mathbf{b}_{i,k}\}_{i\in[N-T],k\in\{K+1,\ldots,K+T\}}) \tag{104}$$

$$= H(\{\sum_{j\in[N-T]}\mathbf{r}_{jk}\}_{k\in\{K+1,\ldots,K+T\}}|\{\mathbf{b}_{i,k}\}_{i\in[N-T],k\in\{K+1,\ldots,K+T\}})+H(\{\mathbf{b}_{i,k}\}_{i\in[N-T],k\in\{K+1,\ldots,K+T\}}) \tag{105}$$

$$= H(\{\sum_{j\in[N-T]}\mathbf{r}_{jk}\}_{k\in\{K+1,\ldots,K+T\}}) + H(\{\mathbf{b}_{i,k}\}_{i\in[N-T],k\in\{K+1,\ldots,K+T\}}) \tag{106}$$

$$= T\frac{d}{K}\log q + (N-T)T\frac{d}{(N-T)K}\log q \tag{107}$$

$$= \frac{2Td}{K}\log q \tag{108}$$

where (102) holds since $\mathbf{\Gamma}$ is invertible, hence represents a bijective mapping. Equation (104) follows from the chain rule of entropy, (105) holds since given, there is no uncertainty in $(\overline{\mathbf{M}}\otimes\mathbf{I})\ \overline{\mathbf{b}}_k$, (106) follows from the independence of the random vectors, and (107) follows from the entropy of uniform random variables.

By combining (108), (97), and (98), with (84), we can rewrite the second term in (78) as follows,

$$H(\{\widetilde{\mathbf{a}}_{ij},[\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]},\{\sum_{j\in[N]}\mathbf{y}_{jk}-\mathbf{a}_k\}_{k\in[K]},\{\sum_{j\in[N]}\mathbf{r}_{jk}-\mathbf{b}_k\}_{k\in\{K+1,\dots,K+T\}}$$
$$\{\mathbf{r}_{ik},\mathbf{b}_{ik},\mathbf{a}_{ik'},\mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T},k'\in[K],l\in[T]\\k\in\{K+1,\dots,K+T\}}}|\mathcal{M}_{\mathcal{T}}^1,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{109}$$
$$=\frac{2Td}{K}\log q+Td\log q+d\log q+Td\Big(\frac{T}{K}+\frac{T}{K(N-T)}+\frac{1}{N-T}+\frac{T}{N-T}\Big)\log q \tag{110}$$
$$=\Big(d\Big(\frac{2T}{K}+T+1\Big)+Td\Big(\frac{T}{K}+\frac{T}{K(N-T)}+\frac{1}{N-T}+\frac{T}{N-T}\Big)\Big)\log q \tag{111}$$

Next, for the first term in (78), we observe that,

$$H(\{\widetilde{\mathbf{a}}_{ij},[\mathbf{a}_{ik}]_j\}_{i\in\mathcal{H},j\in\mathcal{T},k\in[K]},\{\sum_{j\in[N]}\mathbf{y}_{jk}-\mathbf{a}_k\}_{k\in[K]},\{\sum_{j\in[N]}\mathbf{r}_{jk}-\mathbf{b}_k\}_{k\in\{K+1,\dots,K+T\}}$$
$$\{\mathbf{r}_{ik},\mathbf{b}_{ik},\mathbf{a}_{ik'},\mathbf{z}_{ik'l}\}_{\substack{i\in\mathcal{T},k'\in[K],l\in[T]\\k\in\{K+1,\dots,K+T\}}}|\mathcal{M}_{\mathcal{T}}^1,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{112}$$
$$\le\Big(\frac{(N-T)Td}{(N-T)K}+\frac{(N-T)TdK}{(N-T)K}+\frac{Kd}{K}+\frac{Td}{K}+\frac{T^2d}{K}+\frac{T^2d}{K(N-T)}+\frac{Td}{N-T}+\frac{T^2d}{N-T}\Big)\log q \tag{113}$$
$$=\Big(d\Big(\frac{2T}{K}+T+1\Big)+Td\Big(\frac{T}{K}+\frac{T}{K(N-T)}+\frac{1}{N-T}+\frac{T}{N-T}\Big)\Big)\log q \tag{114}$$

Finally, by combining (114) and (111) with (78), the mutual information condition from (74) satisfies the following:

$$0\le I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}^2|\mathcal{M}_{\mathcal{T}}^1,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{115}$$
$$\le\Big(d\Big(\frac{2T}{K}+T+1\Big)+Td\Big(\frac{T}{K}+\frac{T}{K(N-T)}+\frac{1}{N-T}+\frac{T}{N-T}\Big)\Big)\log q$$
$$-\Big(d\Big(\frac{2T}{K}+T+1\Big)+Td\Big(\frac{T}{K}+\frac{T}{K(N-T)}+\frac{1}{N-T}+\frac{T}{N-T}\Big)\Big)\log q \tag{116}$$
$$=0 \tag{117}$$

where the inequality in (115) follows from the non-negativity of mutual information. Hence,

$$I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}^2|\mathcal{M}_{\mathcal{T}}^1,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})=0 \tag{118}$$

for the second term in (44).

**Stage 3: Model Initialization.** We now consider the third term in (44), which corresponds to Stage 3 of PICO, i.e., model initialization. Without loss of generality, we represent the secret share $[\mathbf{w}_i^{(0)}]_j$ sent from user $i\in[N]$ to user $j\in[N]$ as follows:

$$[\mathbf{w}_i^{(0)}]_j\triangleq\mathbf{w}_i^{(0)}+\sum_{k\in[T]}\gamma_j^k\mathbf{z}_{ik}^{(0)} \tag{119}$$

where $\{\mathbf{z}_{i1}^{(0)}\}_{k\in[T]}$ are $T$ random vectors of size $d$ where each element is generated independently and uniformly at random from the finite field $\mathbb{F}_q$, and coefficients $\{\gamma_j\}_{j\in[N]}$ are as defined in (73). We can then rewrite the mutual information condition from the third term in (44) as follows:

$$I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}^3|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{120}$$
$$=I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}^{(0)}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{121}$$
$$=I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{122}$$
$$+I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{123}$$
$$=H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$-H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)})$$
$$+H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$-H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{124}$$

We next consider each term in (124) separately. For the first term in (124), we have that,

$$H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \leq (N-T)T\frac{d}{N-T}\log q \tag{125}$$

$$= Td\log q \tag{126}$$

which follows from the fact that uniform distribution maximizes entropy.

For the second term in (124), we first define,

$$([\mathbf{w}_i^{(0)}]_{N-T+1},\ldots,[\mathbf{w}_i^{(0)}]_N) = \mathbf{w}_i^{(0)}\underbrace{(1,1,\ldots,1)}_{\mathbf{1}} + \underbrace{(\mathbf{z}_{i1}^{(0)},\ldots,\mathbf{z}_{iT}^{(0)})}_{\mathbf{z}_i^{(0)}}\underbrace{\begin{bmatrix}\gamma_{N-T+1} & \cdots & \gamma_N\\ \vdots & \ddots & \vdots\\ \gamma_{N-T+1}^T & \cdots & \gamma_N^T\end{bmatrix}}_{\mathbf{A}} \tag{127}$$

$$= \mathbf{w}_i^{(0)}\mathbf{1} + \mathbf{z}_i^{(0)}\mathbf{A} \tag{128}$$

where $\mathbf{A}$ is an $T\times T$ MDS matrix defined in (90), and $\mathbf{1}$ is a $1\times T$ vector, where each element is equal to 1. Using (128), the second term in (124) can be written as,

$$H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{129}$$

$$\geq H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\{\mathbf{w}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{130}$$

$$= H(\{\mathbf{w}_i^{(0)}\mathbf{1}+\mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\{\mathbf{w}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{131}$$

$$= H(\{\mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\{\mathbf{w}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{132}$$

$$= H(\{\mathbf{z}_i^{(0)}\}_{i\in\mathcal{H}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\{\mathbf{w}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{133}$$

$$= H\left(\{\mathbf{z}_i^{(0)}\}_{i\in\mathcal{H}}\right) \tag{134}$$

$$= (N-T)T\frac{d}{N-T}\log q \tag{135}$$

$$= dT\log q \tag{136}$$

where (130) holds since conditioning cannot increase entropy, and matrix $\mathbf{A}$ in (131) is a $T\times T$ MDS matrix as defined in (90). Equation (133) holds since $\mathbf{A}$ is an MDS matrix, hence is invertible. Equation (134) follows from the independence of the generated random vectors, and (135) follows from the entropy of uniform random variables.

For the third term in (124), we have that,

$$H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{137}$$

$$\leq H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}) \tag{138}$$

$$\leq \Big(\frac{Td}{N-T}+\frac{T^2d}{N-T}\Big)\log q \tag{139}$$

where (138) holds since conditioning cannot increase entropy, and (139) follows from the entropy of uniform random variables.

For the last term in (124), we find that,

$$H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)})$$

$$\geq H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\mathbf{w}^{(0)}) \tag{140}$$

$$= H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathbf{w}^{(0)}) \tag{141}$$

$$= H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}) \tag{142}$$

$$= \Big(\frac{Td}{N-T}+\frac{T^2d}{N-T}\Big)\log q \tag{143}$$

where (140) follows from the fact that conditioning cannot increase entropy, and (141) holds since:

$$\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]} - \mathbf{w}^{(0)}, \{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}} - \mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \{\mathcal{X}_i, \mathbf{y}_i\}_{i\in[N]}, \mathbf{w}^{(J)} \tag{144}$$

forms a Markov chain, and (143) follows from the entropy of uniform random variables. For (142), we first observe,

$$I(\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}; \mathbf{w}^{(0)}, \{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}) \tag{145}$$

$$= I(\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}; \mathbf{w}^{(0)}) + I(\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}; \{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}). \tag{146}$$

For the first term in (146), we find that,

$$0 \le I(\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}; \mathbf{w}^{(0)}) \tag{147}$$

$$= H(\mathbf{w}^{(0)}) - H(\mathbf{w}^{(0)}|\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}) \tag{148}$$

$$\le d\log q - H\left((\mathbf{M}\otimes\mathbf{I})\begin{bmatrix}\mathbf{w}_1^{(0)}\\ \vdots \\ \mathbf{w}_N^{(0)}\end{bmatrix}\middle|\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}\right) \tag{149}$$

$$= d\log q - H\left((\overline{\mathbf{M}}\otimes\mathbf{I})\begin{bmatrix}\mathbf{w}_1^{(0)}\\ \vdots \\ \mathbf{w}_{N-T}^{(0)}\end{bmatrix}\middle|\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}\right) \tag{150}$$

$$= d\log q - H\left((\overline{\mathbf{M}}\otimes\mathbf{I})\begin{bmatrix}\mathbf{w}_1^{(0)}\\ \vdots \\ \mathbf{w}_{N-T}^{(0)}\end{bmatrix}\right) \tag{151}$$

$$= d\log q - H\left(\begin{bmatrix}\mathbf{w}_1^{(0)}\\ \vdots \\ \mathbf{w}_{N-T}^{(0)}\end{bmatrix}\right) \tag{152}$$

$$= d\log q - (N-T)\frac{d}{N-T}\log q \tag{153}$$

$$= 0 \tag{154}$$

where $\mathbf{M}$ and $\overline{\mathbf{M}}$ are as defined in (12), and (85), respectively, (147) is due to the non-negativity of mutual information, (149) holds since entropy is maximized by uniform distribution, (151) holds since the randomness generated by the honest clients is independent from the randomness generated by adversaries, (152) holds since $\overline{\mathbf{M}}$ is an $(N-T)\times(N-T)$ MDS matrix, hence is invertible, and (153) follows from the entropy of unfiormly random variables. From (154), we have that,

$$I(\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}; \mathbf{w}^{(0)}) = 0 \tag{155}$$

Next, for the second term in (146), we find that,

$$0 \le I(\{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}; \{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}) \tag{156}$$

$$= H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}) - H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}, \{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}) \tag{157}$$

where

$$H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}) \le \frac{d}{N-T}(N-T)T\log q \tag{158}$$

since uniform distribution maximizes entropy, and

$$H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}, \{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]})$$

$$= H\Big(\{\mathbf{w}_i^{(0)}\mathbf{1} + \mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|(\mathbf{M}\otimes\mathbf{I})\begin{bmatrix}(\mathbf{w}_1^{(0)})^T & \cdots & (\mathbf{w}_N^{(0)})^T\end{bmatrix}, \{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}\Big) \tag{159}$$

$$= H\Big(\{\mathbf{w}_i^{(0)}\mathbf{1} + \mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|(\overline{\mathbf{M}}\otimes\mathbf{I})\begin{bmatrix}(\mathbf{w}_1^{(0)})^T & \cdots & (\mathbf{w}_{N-T}^{(0)})^T\end{bmatrix}, \{\mathbf{w}_i^{(0)}, \mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}\Big) \tag{160}$$

$$= H\Big(\{\mathbf{w}_i^{(0)}\mathbf{1} + \mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|\begin{bmatrix}(\mathbf{w}_1^{(0)})^T & \cdots & (\mathbf{w}_{N-T}^{(0)})^T\end{bmatrix}^T\Big) \tag{161}$$

$$= H\Big(\{\mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|\begin{bmatrix}(\mathbf{w}_1^{(0)})^T & \cdots & \mathbf{w}_{N-T}^{(0)})^T\end{bmatrix}\Big) \tag{162}$$

$$= H(\{\mathbf{z}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{163}$$

$$= \frac{d}{N-T}(N-T)T\log q \tag{164}$$

which holds since $\overline{\mathbf{M}}$ and $\mathbf{A}$ are MDS matrices (invertible) and that the random vectors are generated independently. By combining (157) with (158) and (164), we find that,

$$I(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]};\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathbf{w}^{(0)}) = 0. \tag{165}$$

Then, by combining (155) and (165) with (146), we have that,

$$I(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]};\mathbf{w}^{(0)},\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}) = 0 \tag{166}$$

from which (142) follows.

Finally, by combining (126), (136), (139), and (143) with (124), we find for (120) the following,

$$0 \le I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}^3|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{167}$$

$$\begin{aligned}&= H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})\\ &\quad - H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)})\\ &\quad + H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})\\ &\quad - H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{w}^{(J)})\end{aligned} \tag{168}$$

$$\le Td\log q - Td\log q + \Big(\frac{Td}{N-T}+\frac{T^2d}{N-T}\Big)\log q - \Big(\frac{Td}{N-T}+\frac{T^2d}{N-T}\Big)\log q \tag{169}$$

$$= 0 \tag{170}$$

Hence, the third term in (44) satisfies:

$$I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}^3|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) = 0. \tag{171}$$

The steps for the remaining two stages follow along the same lines, from which one can find that:

$$I(\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}|\{\mathcal{X}_i,\mathbf{y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) = 0 \tag{172}$$

□

# E COMMUNICATION COMPLEXITY

In the following, we first analyze the per-client communication complexity of PICO.

(Online) The online communication per-client consists of the following components: 1) $O(dm)$ for dataset encoding (Stage 1), 2) $O(\frac{Nd}{K})$ for label encoding (Stage 2), 3) $O(d)$ for model encoding (Stage 4) per training round, 4) $O(d)$ for gradient computing and model update (Stage 5) per training round.

(Offline) The offline communication per-client consists of the following components: 1) $O(Nd\frac{m}{K})$ for dataset encoding (Stage 1), 2) $O(\frac{Nd}{(N-T)K})$ for label encoding (Stage 2), 3) $O(\frac{Nd}{N-T})$ for model initialization (Stage 3), 4) $O(\frac{Nd}{N-T})$ for model encoding (Stage 4) per training round, 5) $O(\frac{Nd}{N-T})$ for gradient computing and model update (Stage 5) per training round.

Hence, the communication overhead per-client is $O(dm+\frac{N}{K}d+dJ)$ in the online phase, and $O(\frac{N}{K}dm+\frac{N}{N-T}dJ)$ in the offline phase. As a result, the total communication complexity across all $N$ clients is $O(Ndm+\frac{N^2}{K}d+NdJ)$ in the online phase, and $O(\frac{N^2}{K}dm+\frac{N^2}{N-T}dJ)$ in the offline phase.

# F COMPUTATION COMPLEXITY

In the following we first analyze the per-client computational overhead of each stage of PICO, for both the offline and online phases, respectively.

**Offline Phase.** The offline phase consists of encoding the local randomness generated by the clients, and random initialization of the model as follows.

**Stage 1:** Generation of $\{\widetilde{\mathbf{R}}_{ij}\}_{j\in[N]}$ requires evaluating a Lagrange polynomial of degree $K+T-1$ at $N$ points. It is known that interpolating a polynomial of degree $\kappa$ (and evaluating it at $\kappa$ points) has a computational complexity of $O(\kappa \log^2 \kappa \log\log \kappa)$ (Kedlaya and Umans, 2011). As such, this stage has a complexity of $O(Nd\frac{m}{K}\log^2(K+T)\log\log(K+T))$ per client.

**Stage 2:** Computing $\{\widetilde{\mathbf{a}}_{ij}\}_{j\in[N]}$ requires evaluating a polynomial of degree $K+T-1$ at $N$ points, which has a computational complexity of $O(N\frac{d}{(N-T)K}\log^2(K+T)\log\log(K+T))$ per client. Computing $\widetilde{\mathbf{a}}_i$ in (10) has a complexity of $O(\frac{Nd}{K})$ per client (since only the non-zero terms should be multiplied due to the identity matrix). Computing the secret shares $\{[\mathbf{a}_{ik}]_j\}_{j\in[N]}$ for all $k\in[K]$ requires evaluating each of the $K$ polynomials of degree $T$ at $N$ points, which has complexity $O(N\frac{d}{N-T}\log^2 T\log\log T)$ for each client. Evaluating the secret shares $\{[\mathbf{a}_k]_i\}_{k\in[K]}$ has an overhead of $O(Nd)$ per client.

**Stage 3:** Computing the secret share $\{[\mathbf{w}_i^{(0)}]_j\}_{j\in[N]}$ requires evaluating a polynomial of degree $T$ at $N$ points, which has complexity $O(N\frac{d}{N-T}\log^2 T\log\log T)$ for each client. Finally, computation of the final secret share, $\mathbf{w}^{(0)}$ from (16) has complexity $O(Nd)$ per client.

**Stage 4:** Computation of $\widetilde{\mathbf{r}}_{ij}^{(t)}$ requires evaluating a Lagrange polynomial of degree $K+T-1$ at $N$ points, which has a complexity of $O(N\frac{d}{N-T}\log^2(K+T)\log\log(K+T))$ per client. Given $\{\widetilde{\mathbf{r}}_{ji}^{(t)}\}_{j\in[N]}$, the computation of $\widetilde{\mathbf{r}}_i^{(t)}$ from (20) has an overhead of $O(Nd)$ per client. Constructing the secret share $[\mathbf{r}_i^{(t)}]_j$ requires evaluating a polynomial of degree $T$ at $N$ points, which has complexity $O(N\frac{d}{N-T}\log^2 T\log\log T)$ for each client. Afterwards, creating the secret share $[\mathbf{r}^{(t)}]_i$ has a complexity of $O(Nd)$ per client. Overall, this stage has a per client computational overhead of $O(N\frac{d}{N-T}\log^2(K+T)\log\log(K+T)+Nd)$ per training round. For $J$ rounds, this leads to an overhead of $O(JN\frac{d}{N-T}\log^2(K+T)\log\log(K+T)+JNd)$ per client.

**Stage 5:** Computing $\{\widetilde{\mathbf{u}}_{ij}^{(t)}\}_{j\in[N]}$ requires evaluating a Lagrange polynomial of degree $(2r+1)(K+T-1)$ at $N$ points, which has a complexity of $O(N\frac{d}{N-T}\log^2 r(K+T)\log\log r(K+T))$ per client per training round. Given $\{\widetilde{\mathbf{u}}_{ji}^{(t)}\}_{j\in[N]}$, computation of $\widetilde{\mathbf{u}}_i^{(t)}$ in (29) has complexity of $O(Nd)$ per client per training round. Next, computing $\sum_{k\in[K]}\mathbf{u}_{ik}^{(t)}$ has a computational overhead of $O(K\frac{d}{N-T})$ per client per training round. Computing the secret shares $\{[\sum_{k\in[K]}\mathbf{u}_{ik}^{(t)}]_j\}_{j\in[N]}$ requires evaluating a polynomial of degree $T$ at $N$ points, which incurs a complexity of $O(N\frac{d}{N-T}\log^2 T\log\log T)$ per client per training round. Finally, given $\{[\sum_{k\in[K]}\mathbf{u}_{jk}^{(t)}]_i\}_{j\in[N]}$, the computation of $[\sum_{k\in[K]}\mathbf{u}_k^{(t)}]_i$ has complexity of $O(Nd)$ per client per training round. For $J$ iterations, the computational complexity is $O(JN\frac{d}{N-T}\log^2 r(K+T)\log\log r(K+T)+JNd)$ per client.

Overall, the computation complexity of the offline phase is $O(Nd\frac{m}{K}\log^2(K+T)\log\log(K+T)+JN\frac{d}{N-T}\log^2 r(K+T)\log\log r(K+T)+JNd)$ per client.

**Online Phase.** The online phase consists of encoding the dataset and the model, gradient computations, and model update.

**Stage 1:** Computing $\hat{\mathbf{X}}_i$ has an overhead of $O(md)$ per client, as each client holds a local dataset of size $m$ locally. Computing $\widetilde{\mathbf{X}}_i$ has an overhead of $O(Nmd)$ per client.

**Stage 2:** Computation of $\mathbf{y}_i=\sum_{l\in\mathcal{X}_i}\mathbf{x}_l^T y_l$ has complexity of $O(md)$ per client. Computation of $\{\widetilde{\mathbf{y}}_{ij}\}_{j\in[N]}$ requires evaluation of Lagrange polynomial of degree $K+T-1$ at $N$ points, which has complexity of $O(N\frac{d}{K}\log^2(K+T)\log\log(K+T))$ per client. Given $\{\widetilde{\mathbf{y}}_{ji}\}_{j\in[N]}$ and $\widetilde{\mathbf{a}}_i$ (from offline computation), computation of $\hat{\mathbf{a}}_i$ incurs a complexity of $O(N\frac{d}{K})$ per client. Next, upon receiving $\{\hat{\mathbf{a}}_j\}_{j\in[N]}$ from at least $K+T$ clients, client $i$ recovers $\sum_{j\in[N]}\mathbf{y}_{jk}-\mathbf{a}_k$ for all $k\in[K]$, which has complexity of $O(\frac{d}{K}(K+T)\log^2(K+T)\log\log(K+T))$. Next, computation of $[\mathbf{X}^T\mathbf{y}]_i$ from (15) has complexity of $O(d)$ per client.

Table 2: Comparison of the total computation overhead (per client) for PICO with respect to COPML with $m_i = m$ for all $i \in [N]$.

| | COPML | PICO | |
|---|---|---|---|
| 1. Dataset encoding | $O(N^2 \frac{m}{K} d \log^2(K+T) \log\log(K+T))$ | (online) | $O(Ndm)$ |
| | | (offline) | $O(N \frac{m}{K} d \log^2(K+T) \log\log(K+T))$ |
| 2. Label encoding | $O(N(m+d) \log^2 T \log\log T)$ | (online) | $O(N \frac{d}{K} \log^2(K+T) \log\log(K+T))$ |
| | | (offline) | $O(N \frac{d}{(N-T)K} \log^2(K+T) \log\log(K+T) + \frac{Nd}{K} + N \frac{d}{N-T} \log^2 T \log\log T)$ |
| 3. Model initialization | $O(Nd \log^2(K+T) \log\log(K+T))$ | (online) | $-$ |
| | | (offline) | $O(N \frac{d}{N-T} \log^2 T \log\log T + Nd)$ |
| 4. Model encoding | $O(JNd \log^2(K+T) \log\log(K+T))$ | (online) | $O(KdJ + TdJ \log^2 T \log\log T)$ |
| | | (offline) | $O(JN \frac{d}{N-T} \log^2(K+T) \log\log(K+T) + JNd)$ |
| 5. Gradient comp./ model update | $O(J \frac{Nm}{K}(d+r) + Jdr(K+T) \times \log^2 r(K+T) \log\log r(K+T))$ | (online) | $O(J \frac{Nm}{K}(d+r) + Jdr(K+T) \times \log^2 r(K+T) \log\log r(K+T))$ |
| | | (offline) | $O(JN \frac{d}{N-T} \log^2 r(K+T) \times \log\log r(K+T) + JNd)$ |

Table 3: Comparison of the total computation overhead (per client) for PICO with respect to COPML with $m_i = m$ for all $i \in [N]$, when $K, T = O(N)$

| | COPML | PICO (online+offline) |
|---|---|---|
| 1. Dataset encoding | $O(Ndm \log^2 N \log\log N)$ | $O(Ndm + dm \log^2 N \log\log N)$ |
| 2. Label encoding | $O(N(m+d) \log^2 N \log\log N)$ | $O(d \log^2 N \log\log N)$ |
| 3. Model initialization | $O(Nd \log^2 N \log\log N)$ | $O(d \log^2 N \log\log N + Nd)$ |
| 4. Model encoding | $O(JNd \log^2 N \log\log N)$ | $O(JNd \log^2 N \log\log N)$ |
| 5. Gradient comp./ model update | $O(Jm(d+r) + JNd \log^2 N \log\log N)$ | $O(Jm(d+r) + JNd \log^2 N \log\log N))$ |

**Stage 4:** Computing $\widehat{\mathbf{w}}^{(t)}$ requires interpolating a polynomial of degree $T$, which has a complexity of $O(Td \log^2 T \log\log T)$ per client per training round. Computing the encoded model $\widetilde{\mathbf{w}}_i$ has a computation overhead of $O(Kd)$ per client. As the above computation steps should be repeated at every training round, for a total number of $J$ training iterations, the computational overhead is $O(KdJ + TdJ \log^2 T \log\log T)$ per client.

**Stage 5:** Computation of the gradient $\widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)})$ has an overhead of $O(\frac{Nm}{K}(d+r))$ per client, at each training round. Next, the computation of $\hat{\mathbf{u}}_i$ has an overhead of $O(d)$ per client. Then, each client recovers $\{\mathbf{X}_k^T \hat{g}(\mathbf{X}_k \times \mathbf{w}^{(t)}) - \mathbf{u}_k^{(t)}\}_{k \in [K]}$, which requires interpolating a polynomial of degree $(2r+1)(K+T-1)$, which has complexity $O(dr(K+T) \log^2 r(K+T) \log\log r(K+T))$ per client. Finally, the summation to obtain $[\mathbf{X}^T \hat{g}(\mathbf{X} \times \mathbf{w}^{(t)})]_i$ has a computational cost $O(Kd)$ per client. The computation overhead of model update is $O(d)$. The above computation steps are operated over $J$ training iterations. For $J$ training rounds, the computation complexity is $O(J \frac{Nm}{K}(d+r) + Jdr(K+T) \log^2 r(K+T) \log\log r(K+T))$ per client.

Overall, the computation complexity of the online phase is $O(Nmd + N \frac{d}{K} \log^2(K+T) \log\log(K+T) + J \frac{Nm}{K}(d+r) + Jdr(K+T) \log^2 r(K+T) \log\log r(K+T))$ per client.

**Computation complexity of PICO vs COPML.** In Table 2, we demonstrate the per-client computational complexity of PICO versus COPML (So et al., 2020) for each stage. For a fair comparison, we also considered the utilization of fast polynomial interpolation mechanisms (Kedlaya and Umans, 2011) for evaluating and interpolating polynomials in COPML (hence the complexity we report below is even lower than the one originally reported in So et al. (2020)).

In Table 3, we demonstrate the per-client computational complexity for PICO (offline+online) and COPML, with $T = O(N)$ and $K = O(N)$. We observe that the overall per-client complexity (across all algorithm steps) is $O(Ndm + dm \log^2 N \log\log N + JNd \log^2 N \log\log N + Jm(d+r))$ for PICO and $O(Ndm \log^2 N \log\log N + JNd \log^2 N \log\log N + Jm(d+r))$ for COPML, respectively. Hence, PICO achieves the same computation complexity as COPML. This is due to the fact that PICO reduces the overall number of variables encoded, hence the additional operations due to the matrix transformations with MDS matrices do not increase the overall computation complexity.

# G CORRECTNESS

We now demonstrate the correctness of PICO. The correctness of the encoding and decoding process is a result of the decodability of the Lagrange interpolation polynomial, in particular, any polynomial $f$ of degree $\deg(f)$ can be uniquely reconstructed from any set of at least $\deg(f)+1$. As such, as long as the total number of clients $N$ satisfy the minimum number identified by the recovery threshold, i.e., $N-D \geq (2r+1)(K+T-1)+1$, then all polynomials, including both the encoded computations using the Lagrange interpolation polynomial, as well as Shamir's secret sharing (which corresponds to a degree $T$ polynomial), is decodable. As such, one can correctly recover $\mathbf{w}^{(J)}$ from the computations performed on the encoded datasets and models.

# H FINITE FIELD REPRESENTATION

To represent the real-valued data points in a finite field $\mathbb{F}_q$ of integers modulo a prime $q$, each client initially quantizes its local dataset from the real domain to the domain of integers, and then embeds it in a field $\mathbb{F}_q$ of integers modulo a large prime $q$. Parameter $q$ is selected sufficiently large to avoid wrap-around in finite field computations. The dataset $\mathcal{X}_i$ of client $i \in [N]$ is quantized using a simple scalar quantization function $\varphi\left(round(2^c \cdot \mathbf{x}_l)\right)$ for all $l \in \mathcal{X}_i$, with the stochastic rounding operation,

$$round(x) = \begin{cases} \lfloor x \rfloor & \text{if} \quad x - \lfloor x \rfloor < 0.5 \\ \lfloor x \rfloor + 1 & \text{otherwise} \end{cases} \tag{173}$$

applied element-wise to the features $x$ of data point $\mathbf{x}_l$ from dataset $\mathcal{X}_i$ of client $i$, where $c$ is an integer parameter to control the quantization loss. $\lfloor x \rfloor$ is the largest integer less than or equal to $x$, and function $\varphi : \mathbb{Z} \to \mathbb{F}_q$ is a mapping used to represent a negative integer in the finite field, also known as *two's complement representation*,

$$\varphi(x) = \begin{cases} x & \text{if } x \geq 0 \\ q + x & \text{if } x < 0 \end{cases} \tag{174}$$

which maps the positive/negative numbers to the first/second half of the finite field, respectively. All operations are then carried out within the finite field $\mathbb{F}_q$. After the completion of training (i.e., after $J$ training iterations), the final model $\mathbf{w}^{(J)}$ is mapped back from the finite field to the real domain, by inverting (174), i.e., $\mathbf{w}^{(J)} \leftarrow (\varphi^{-1}(\mathbf{w}^{(J)}))$.

# I ADDITIONAL EXPERIMENTAL DETAILS

**Details on the communication protocol.** In all our experiments, the inter-client communication is implemented using the `MPI4Py` Message Passing Interface (MPI) for Python. The `broadcast` functionality of the MPI protocol communicates messages through a tree topology, as opposed to an ideal broadcast (e.g., a cellular network). As such, the communication overhead of PICO scales with respect to $O(N \log N)$ in the experiments, slightly higher than $O(N)$. This suggests PICO could in principle achieve even higher gains in an ideal broadcasting setting, such as a cellular network among devices within the same coverage area.

**Details on the datasets.** For the CIFAR-10 dataset, $9019$ samples are used in the training set, and an additional $1000$ samples are reserved for test set. Then, each local training set is complemented with simple random crop augmentation (to avoid having too few samples per client as the number of clients increase), leading to a total number of $18038$ training samples. Similarly, for the MNIST dataset, $11432$ samples are used for training, and additional $2115$ samples are reserved for testing. Then, each local training set is again complemented with random crop augmentation, leading to $22864$ training samples. Model accuracy is then evaluated on the test set, using the model trained jointly across the $N$ clients.

For binary classification, we consider the following two classes, the *plane* and *car* classes on CIFAR-10 and digits $0$ and $1$ for MNIST. Note that the size of the MNIST dataset for the two classes is larger than that of CIFAR-10. In our experiments, we have observed that as the dataset size grows, the performance gain of PICO over the baseline increases, for both communication overhead and online wall-clock training time. Moreover, the gain of PICO over the baseline increases as the number of clients increase, for both the communication overhead and wall-clock training time, suggesting PICO to be a promising candidate for large-scale collaborative learning tasks.

**Secure truncation.** In the experiments, for both PICO and COPML, we leverage the secure truncation protocol from Catrina and Saxena (2010), as suggested by So et al. (2020), to reduce the size of the finite field required for model update operations. The secure truncation protocol takes the secret shares $\{[x]_i\}_{i\in[N]}$ of a secret variable $x$ (where client $i$ holds a

share $[s]_i$), along with two public integer parameters $\kappa_1$ and $\kappa_2$ such that $0 < \kappa_1 < \kappa_2$, and $x \in \mathbb{F}_{2^{\kappa_2}}$. Then, the secure truncation protocol returns the secret shares $\{[z]_i\}_{i\in[N]}$ of a variable $z$ such that $z = \lceil \frac{x}{2^{\kappa_1}} \rceil + b$ where $b$ is a Bernoulli random variable (random bit) with probability $P[b = 1] = (x \mod 2^{\kappa_1})/2^{\kappa_1}$. In other words, the secure truncation protocol quantizes the secret $x$, by rounding $x/(2^{\kappa_1})$ to the nearest integer with probability $1 - \rho$, where $\rho$ is the distance between the two.

During training, secure truncation is applied for the model updating stage, to ensure that the range of the updated model stays within the range of the finite field. In particular, after obtaining $[\mathbf{X}^T \hat{g}(\mathbf{X} \times \mathbf{w}^{(t)})]_i$ from (33), client $i$ computes a secret share,

$$[\mathbf{X}^T \hat{g}(\mathbf{X} \times \mathbf{w}^{(t)})]_i - [\mathbf{X}^T \mathbf{y}]_i = [\mathbf{X}^T (\hat{g}(\mathbf{X} \times \mathbf{w}^{(t)}) - \mathbf{y})]_i, \tag{175}$$

then, clients carry out a secure truncation operation to multiply $\mathbf{X}^T(\hat{g}(\mathbf{X} \times \mathbf{w}^{(t)}) - \mathbf{y}$ with parameter $\frac{\eta}{|\mathcal{X}|}$, where $\frac{\eta}{|\mathcal{X}|} < 1$. Then, the model is updated at each client $i$ as follows,

$$[\mathbf{w}^{(t+1)}]_i = [\mathbf{w}^{(t)}]_i - \frac{\eta}{|\mathcal{X}|}[\mathbf{X}^T(\hat{g}(\mathbf{X} \times \mathbf{w}^{(t)}) - \mathbf{y})]_i \tag{176}$$

$$= [\mathbf{w}^{(t)} - \frac{\eta}{|\mathcal{X}|}\mathbf{X}^T(\hat{g}(\mathbf{X} \times \mathbf{w}^{(t)}) - \mathbf{y})]_i \tag{177}$$

after which each client learns the secret share $[\mathbf{w}^{(t+1)}]_i$ of the model $\mathbf{w}^{(t+1)}$ for the next training round. In the experiments, we use $(\kappa_1, \kappa_2) = (21, 24)$ for both datasets.