SCALR: Communication-Efficient Secure Multi-Party Logistic Regression

Xingyu Lu, Hasin Us Sami, *Graduate Student Member, IEEE*, and Başak Güler, *Member, IEEE*

***Abstract*— Privacy-preserving coded computing is a popular framework for multiple data-owners to jointly train machine learning models, with strong end-to-end information-theoretic privacy guarantees for the local data. A major challenge against the scalability of current approaches is their communication overhead, which is quadratic in the number of users. Towards addressing this challenge, we present SCALR, a communication-efficient collaborative learning framework for training logistic regression models. To do so, we introduce a novel coded computing mechanism, by decoupling the communication-intensive encoding operations from real-time training, and offloading the former to a *data-independent offline phase*, where the communicated variables are independent from training data. As such, the offline phase can be executed proactively during periods of low network activity. Communication complexity of the *data-dependent* (online) training operations is only *linear* in the number of users, greatly reducing the *quadratic* state-of-the-art. Our theoretical analysis presents the information-theoretic privacy guarantees, and shows that SCALR achieves the same performance guarantees as the state-of-the-art, in terms of adversary resilience, robustness to user dropouts, and model convergence. Through extensive experiments, we demonstrate up to $80\times$ reduction in online communication overhead, and $6\times$ speed-up in the wall-clock training time compared to the state-of-the-art.**

***Index Terms*— Privacy-preserving distributed learning, information-theory, decentralized training.**

I. Introduction

MACHINE learning has led to recent breakthroughs in a variety of fields. In many modern applications, the data is privacy-sensitive (such as healthcare records or geolocation data), and distributed across a large number of data-owners. Information and coding theory offers a promising approach to the design of privacy-preserving machine learning (PPML) frameworks, called *privacy-preserving coded computing*, as initiated by the recent works [1], [2], [3], [4], [5]. These approaches build on Lagrange coded computing (LCC), a popular framework for function computation over data distributed across multiple users [1]. LCC allows computing any multi-variate polynomial function $f(\mathbf{X}_1),\ldots,f(\mathbf{X}_K)$ of degree $\deg f$ on a dataset $\mathbf{X}=(\mathbf{X}_1,\ldots,\mathbf{X}_K)$, while providing strong information-theoretic privacy guarantees against T adversarial users, and robustness against S dropout (or straggling) users, as long as $N\geq(T+K-1)\deg f+S+1$. To do so, the dataset $\mathbf{X}$ is first *encoded* by combining the K parts $\mathbf{X}_1,\ldots,\mathbf{X}_K$ along with T random matrices $\mathbf{R}_1,\ldots,\mathbf{R}_T$ using the well-known Lagrange interpolation polynomial. Then, each user $i\in[N]$ receives an encoded dataset $\widetilde{\mathbf{X}}_i$, and computes the function $f(\widetilde{\mathbf{X}}_i)$ over the encoded dataset. At the end, the true function values $f(\mathbf{X}_1),\ldots,f(\mathbf{X}_N)$ can be decoded by collecting the computations performed on the encoded datasets, via polynomial interpolation.

In the context of PPML, adversarial users are parties who try to gain information on the privacy-sensitive data of other users. In this work, our focus is on the honest but curious adversary model, as is the most common threat model in PPML [5]. Such adversaries follow the protocol truthfully, but try to gain additional information on the private data of honest users using the information exchanged during the protocol. In a network of N users, we assume that up to T users are adversarial, who may collude with each other. Dropout and straggling users, on the other hand, are parties who fail to send their messages successfully during training, due to various reasons such as poor connectivity, low battery, or device unavailability. Out of N users, it is assumed that up to S users may drop out at any given training round.

Collaborative privacy-preserving machine learning (COPML) is a recent application of LCC to logistic regression (a binary classification framework widely-used due to its practicality and interpretability). In this approach, the dataset $\mathbf{X}$ corresponds to the *combination of N local datasets*, each held by a different user. Then, COPML encodes $\mathbf{X}$ by first partitioning it into K parts, and then combining the K parts along with T random matrices using a Lagrange interpolation polynomial. The additional randomness protects the privacy of sensitive local datasets against any collusions between up to T adversarial users, such that no information (in an information-theoretic sense) about the datasets are revealed beyond the final model. At the end, user i learns an encoded dataset $\widetilde{\mathbf{X}}_i$ whose size is only $(1/K)^{th}$ of the original dataset $\mathbf{X}$. The training computations are then performed on the encoded datasets, *as if they were computed on the original datasets*. As the network size N grows, one can select a larger K, reducing the training load per user, speeding up training. At the end, the final model

Manuscript received 1 November 2022; revised 11 May 2023; accepted 15 August 2023. Date of publication 28 August 2023; date of current version 17 January 2024. This research was sponsored in part by the NSF CAREER Award CCF-2144927, OUSD (R&E)/RT&L under Cooperative Agreement Number W911NF-20-2-0267, and the UCR OASIS Funding Award. The views and conclusions contained in this document are those of the authors. The associate editor coordinating the review of this article and approving it for publication was M. Ji. *(Corresponding author: Başak Güler.)*

The authors are with the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA 92521 USA (e-mail: xlu065@ucr.edu; hsami003@ucr.edu; bguler@ece.ucr.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2023.3308954.

Digital Object Identifier 10.1109/TCOMM.2023.3308954

is recovered using polynomial interpolation, by collecting the local computations from a sufficient number of users, as long as $N \geq (T + K - 1)\deg f + S + 1$. COPML has shown significant (an order of magnitude) speed-up in training compared to conventional cryptographic PPML frameworks [5].

The major challenge of such privacy-preserving coded machine learning approaches is the *communication overhead*. This is due to the fact that the degree of the polynomial $f(\cdot)$ doubles with every multiplication operation (associated with gradient computations), which leads to an exponential increase in the minimum number of users required for successful decoding of the final model, since the number of users must satisfy $N \geq (T+K-1)\deg f + S + 1$. After several training rounds, the degree $\deg f$ can grow to a level that the number of users is no longer sufficient for correct decoding of the final model. To prevent such a *degree explosion*, a communication-intensive *degree reduction* step is carried out, to reduce the polynomial degree after each training round. This process has a communication complexity of $O(N^2)$ per training iteration, creating a major bottleneck against scalability to large networks.

To address this challenge, in this work we propose SCALR, a **S**ecure **C**ommunication-efficient Multi-p**a**rty **L**ogistic **R**egression framework. The key intuition behind SCALR is an online-offline communication trade-off for Lagrange coding. Online communication depends on the local datasets, hence should be carried out after training starts. Offline communication is independent of the data and/or the model, hence can be carried out in advance when the network traffic is low, or paralellized with other components of training. In doing so, SCALR introduces a novel encoding and degree reduction strategy for Lagrange coding, which achieves a highly-efficient (linear) online communication overhead, as opposed to the quadratic online communication overhead of the state-of-the-art, while providing equal adversary-resilience, robustness to user dropouts, and training performance. Through extensive distributed experiments for various image classification tasks, we observe that SCALR reduces the online communication overhead by up to $80\times$, and achieves $6\times$ speed up in the wall-clock training time compared to the state-of-the-art while achieving comparable model accuracy. Our contributions can be summarized as follows:

- We propose an online-offline communication trade-off for Lagrange coding, where communication is decoupled into data-dependent online and data-agnostic offline phases.
- We introduce SCALR, a communication-efficient logistic regression framework with *linear* online communication overhead, as opposed to the quadratic overhead of the state-of-the-art.
- We present formal information-theoretic privacy guarantees for SCALR, while achieving the adversary-resilience, robustness to user dropouts, and model accuracy of the state-of-the-art.
- Through extensive numerical experiments, we demonstrate an order of magnitude reduction in the online communication overhead compared to the state-of-the-art.

II. Related Work

In addition to the coding-theoretic approaches, there are three complementary approaches to PPML. *Secure multi-party computing (MPC)* protocols are based on a cryptographic primitive known as *secret sharing*, where parties inject randomness to sensitive data, and computations are then performed on the secret shared data [6], [7], [8], [9], [10], [11], [12]. Secure MPC can provide information-theoretic privacy, however, requires extensive communication and interaction between the parties. As such, current constructions are limited to 3-4 parties [13], [14], [15]. Recently, MPC has also been utilized for gradient aggregation in federated learning, also known as *secure aggregation*, where the aggregated gradient/model is revealed after each training round [16], [17], [18], [19]. It has been shown, however, that the aggregated models can still reveal private information over multiple training rounds [20], [21]. In contrast, our focus is on *end-to-end* training, where no intermediate model/gradient can be revealed (even in aggregated form) beyond the final model.

Differential Privacy (DP) is a noisy release mechanism that aims to protect the privacy of personally identifiable information (PII) by injecting *permanent* noise (unlike MPC or HE) to the computations, so that an adversary observing the released model cannot backtrack whether a certain individual's information was used in the computations [22], [23], [24], [25], [26], [27], [28]. Privacy in DP is quantified by the amount of noise injected in the training computations; stronger privacy requires higher noise, leading to a privacy-accuracy trade-off. In contrast, our focus is on ensuring information-theoretic privacy throughout training, while preserving the accuracy of the final model. Although beyond our current focus, we note that the two can in principle be combined and benefit DP, as recent works have shown that information-theoretic techniques can boost DP accuracy [29].

Homomorphic encryption (HE) is a cryptographic framework which allows computations to be performed on encrypted data [30], [31]. The privacy guarantees of HE are based on computational assumptions (adversaries have bounded computational power), as opposed to information-theoretic privacy (where adversaries may have unlimited computational power). Computing in the encrypted domain is computationally-intensive, and stronger privacy guarantees require a larger encrypted data size, which limits scalability in larger networks. As such, HE is primarily utilized for the inference task in ML, as opposed to the more computationally-intensive training (which is the focus of current work) [32], [33], [34], [35], [36], [37].

III. System Model

We consider a network of N users illustrated in Fig. 1, where user i holds a local dataset $\mathcal{D}_i$ consisting of $|\mathcal{D}_i|$ data points, represented by a $|\mathcal{D}_i| \times d$ matrix $\overline{\mathbf{X}}_i$, along with the labels $\mathcal{Y}_i \in \{0,1\}^{|\mathcal{D}_i|}$. The collection of all local datasets $\mathcal{D} \triangleq \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_N$ is represented by a $|\mathcal{D}| \times d$ matrix $\mathbf{X} \triangleq (\overline{\mathbf{X}}_i^T, \ldots, \overline{\mathbf{X}}_N^T)^T$ where the i^{th} row $\mathbf{x}_i$ holds the features of data point $i \in \mathcal{D}$, and d is the number of features. The corresponding labels are represented by a vector $\mathbf{y} \in \{0,1\}^{|\mathcal{D}|}$,

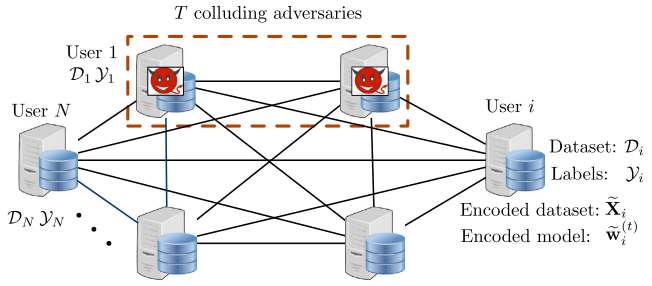


Fig. 1. **System model.** The collaborative learning architecture with N users. User $i \in [N]$ holds a dataset $\mathcal{D}_i$ with labels $\mathcal{Y}_i$.

where the i^{th} element $y_i \in \{0,1\}$ denotes the label of data point $i \in |\mathcal{D}|$. The goal is to train a logistic regression model $\mathbf{w}$ over $\mathbf{X}$, by minimizing a cross entropy loss function:

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

where $\hat{y}_i = g(\mathbf{x}_i \times \mathbf{w}) \in (0,1)$ is the probability of label i being equal to 1, and $g(x) \triangleq 1/(1+e^{-x})$ is the sigmoid function. The model is trained via gradient descent with a learning rate η,

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

where $\nabla\mathcal{F}(\mathbf{w}) \triangleq \frac{1}{|\mathcal{D}|}\mathbf{X}^T(g(\mathbf{X}\times\mathbf{w}) - \mathbf{y})$ is the gradient, $\mathbf{w}^{(t)}$ is the state of the model at training iteration t, and function $g(\cdot)$ is applied element-wise over $\mathbf{X}\times\mathbf{w}^{(t)}$. At each training round, up to S users may drop out from the system (e.g., due to poor channel quality, low battery etc).

A. Threat Model

We consider an *honest-but curious* adversary model (most common threat model in PPML [2], [3], [4], [5]), where adversaries follow the protocol, but try to reveal the datasets of honest users using the messages exchanged during training. From N users, up to T are adversarial (who may collude with each other) denoted by a set $\mathcal{T}$. Honest users are denoted by a set $\mathcal{H} = [N]\backslash\mathcal{T}$.

B. Information-Theoretic Privacy

Our goal is to ensure that adversaries learn no information about the local datasets of honest users, beyond the final model. Formally, for all $\mathcal{T}$ such that $|\mathcal{T}| \leq T$, and J being the total the number of training rounds, this condition can be stated as,

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

where I denotes the mutual information, and $\mathcal{M}_\mathcal{T}$ is the collection of all messages received by adversaries. Similar to [2], [4], [5], and [3], our framework is bound to finite field operations, which requires the representation of datasets in a finite field. As such, in the sequel, we assume that all datasets are represented in a finite field $\mathbb{F}_q$ of integers modulo a large prime q, and all operations are carried out in $\mathbb{F}_q$. For space considerations, we refer to [2], [4], [5], and [3] for the details of this mapping.

C. Background and Challenges

In order to solve (1) under the constraint (3), the state-of-the-art is the COPML framework from [5]. In this setup, user $i \in [N]$ first secret shares its local dataset $\overline{\mathbf{X}}_i \in \mathbb{F}_q^{D\times d}$ using Shamir's T-out-of-N secret sharing (detailed in Appendix A), by sending a secret share $[\overline{\mathbf{X}}_i]_j \in \mathbb{F}_q^{D\times d}$ to each user $j \in [N]$ where $D = |\mathcal{D}_i|$ for $i \in [N]$. This has a (quadratic) total communication overhead of $O(N^2Dd)$ across the N users. Then, Lagrange encoding is performed using the secret shares. To do so, user $i \in [N]$ concatenates the received shares $\{[\overline{\mathbf{X}}_j]_i\}_{j\in[N]}$, partitions it into K equal-sized shards $\{[\mathbf{X}_k]_i\}_{k\in[K]}$, forms a Lagrange interpolation polynomial of degree $K+T-1$,

$$[\phi(z)]_i \triangleq \sum_{k\in[K]}[\mathbf{X}_k]_i \prod_{l\in[K+T]\backslash\{k\}}\frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{l=K+1}^{K+T}[\mathbf{V}_k]_i \prod_{l\in[K+T]\backslash\{k\}}\frac{z-\beta_l}{\beta_k-\beta_l} \quad (4)$$

where $[\phi(\beta_k)]_i = [\mathbf{X}_k]_i$ for $k \in [K]$ and $i \in [N]$, and sends an evaluation $[\widetilde{\mathbf{X}}_j]_i \triangleq [\phi(\alpha_j)]_i$ to user $j \in [N]$. Here, $[\mathbf{V}_k]_i$ denotes the secret share of a random matrix $\mathbf{V}_k \in \mathbb{F}_q^{\frac{D}{K}\times d}$ for $k \in [T]$ generated by a crypto-service provider [5]. Upon receiving $\{[\widetilde{\mathbf{X}}_i]_j\}_{j\in[N]}$, client $i \in [N]$ recovers its encoded matrix $\widetilde{\mathbf{X}}_i$ through polynomial interpolation. The total communication overhead of this stage across the N users is $O(\frac{N^2}{K}Dd)$. The model $\mathbf{w}^{(t)}$ is encoded similarly; at each round $t \in [J]$, user i learns an encoded model $\widetilde{\mathbf{w}}_i^{(t)}$. The encoded dataset $\widetilde{\mathbf{X}}_i$ and model $\widetilde{\mathbf{w}}_i^{(t)}$ correspond to evaluation points of the following degree $K+T-1$ Lagrange polynomials,

$$\phi(z) = \sum_{k\in[K]}\mathbf{X}_k \prod_{l\in[K+T]\backslash\{k\}}\frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{l=K+1}^{K+T}\mathbf{V}_k \prod_{l\in[K+T]\backslash\{k\}}\frac{z-\beta_l}{\beta_k-\beta_l} \quad (5)$$

$$\psi^{(t)}(z) \triangleq \sum_{k\in[K]}\mathbf{w}^{(t)} \prod_{l\in[K+T]\backslash\{k\}}\frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T}\mathbf{v}_k^{(t)} \prod_{l\in[K+T]\backslash\{k\}}\frac{z-\beta_l}{\beta_k-\beta_l} \quad (6)$$

where $\widetilde{\mathbf{X}}_i = \phi(\alpha_i)$ and $\widetilde{\mathbf{w}}_i^{(t)} = \psi^{(t)}(\alpha_i)$, respectively, and $\{\mathbf{V}_k, \mathbf{v}_k^{(t)}\}_{k\in\{K+1,\ldots,K+T\}}$ are random masks that hide the true dataset $(\mathbf{X}_1,\ldots,\mathbf{X}_K)$ and model $\mathbf{w}^{(t)}$ against up to T adversaries.

D. Degree Explosion

Using the encoded dataset and model, user $i \in [N]$ computes a local gradient $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}^{(t)})$, which is then used to update the model. After multiple training rounds, the final model $\mathbf{w}_i^{(J)}$ is decoded via polynomial interpolation, by collecting the computations from at least $\deg(f)+1$ users. On the other hand, the degree $\deg f$ grows *exponentially* over the iterations, leading to a *degree-explosion* after multiple

rounds, preventing correct recovery of the final model. We next demonstrate degree explosion with an illustrative example. The gradient $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}^{(0)}) \triangleq \widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(0)})$ computed by user i at round $t = 0$ corresponds to a degree $(2r + 1)(K + T - 1)$ polynomial, where $\hat{g}(\cdot)$ is a degree r polynomial approximation of the sigmoid function (as detailed in Section IV). Then, from (2), the encoded model $\widetilde{\mathbf{w}}^{(1)}$ at round $t = 1$ will have degree $(2r + 1)(K + T - 1)$, and the corresponding gradient $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}^{(1)}) \triangleq \widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(1)})$ will have degree $(K + T - 1) + r((K + T - 1) + (2r + 1)(K + T - 1)) = (2r^2 + 2r + 1)(K + T - 1)$. After J training rounds, the degree will grow to $\deg(f) = (2r^J + 2r^{J-1} + \ldots + 2r + 1)(K + T - 1)$, requiring the local computations from at least $\deg(f) + 1$ users to decode the final model. As the number of iterations grow, the total number of users will no longer be sufficient to decode the model. This necessitates a communication-intensive *degree reduction* step (described in Appendix A) after each training round, where user $i \in [N]$ secret shares its local gradient $\widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)}) \in \mathbb{F}_q^{d \times 1}$ using Shamir's T-out-of-N secret sharing, by sending a secret share $[\widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)})]_j$ to each user $j \in [N]$. This incurs a quadratic $O(N^2 d)$ total communication overhead (across N users) per round. Unlike dataset encoding, the degree reduction operation is repeated at each training round, hence the overhead increases as the number of training rounds increase. The data-reliant nature of this communication requires online communication during training, leading to limited scalability in low-bandwidth environments.

E. Main Problem

In this work, our goal is to address this challenge. In particular, we seek to develop an efficient framework that decouples the communication-intensive operations from real-time training. To this end, we separate the overall communication into two phases:

1) *Online phase* depends on the training data, hence can only take place after training starts. For fast training, it is critical to have a highly efficient online communication protocol.

2) *Offline phase* is independent from the training data, such as randomness generation. Hence, offline phase can take place way in advance before training starts, when the network traffic is low, or can be parallelized with other components of training, such as gradient computations.

Our goal is to develop a PPML framework with highly efficient online communication, whose overhead is no greater than $O(N)$ (linear in the number of users), along with an offline communication component whose overhead is no greater than $O(N^2)$. We ask the following:

- How can we solve (1) under the information-theoretic guarantees from (3), with linear online communication complexity?

To address this challenge, in this work we introduce SCALR, a communication-efficient logistic regression framework with *linear* online communication overhead. The key contribution is a novel encoding and degree reduction strategy with a linear online communication overhead of $O(N)$ (broadcast), as opposed to the former $O(N^2)$ (point-to-point) online overhead of the state-of-the-art. To do so, SCALR decouples the communication-intensive operations for coding and model updating, and offloads them to a data-agnostic offline phase, which can be performed in advance during low network traffic. We next describe the details of SCALR.

TABLE I
ONLINE COMMUNICATION OVERHEAD OF SCALR WITH RESPECT TO COPML, WITH $|\mathcal{D}_i| = D$ FOR ALL $i \in [N]$

	COPML	SCALR
Stage 1. Dataset encoding	$O(N^2 dD)$	$O(NdD)$
Stage 2. Label secret sharing	$O(N^2 d)$	$O(Nd)$
Stage 3. Model initialization	-	-
Stage 4. Model encoding	$O(N^2 dJ)$	$O(NdJ)$
Stage 5. Gradient comp. and model update	$O(N^2 dJ)$	$O(NdJ)$

IV. THE SCALR FRAMEWORK

SCALR consists of five main components shown in Table I, where the online communication overhead of each component is compared with respect to COPML [5]. The offline phases do not depend on online/offline phases from previous stages, hence can be fully carried out in advance and in parallel, independently from other components. As such, user dropouts are assumed to occur only during the online phase, and after the dataset encoding and label secret sharing stages. For the latter, if any users drop out during dataset encoding/label secret sharing, training can proceed by removing such dropout users from the protocol. We next describe the details of each component. Table II provides the list of notations used in the sequel.

Stage 1: Dataset Encoding. The first stage of SCALR is Lagrange encoding of the datasets, which enables the training computations to be handled efficiently, while hiding the contents of the local datasets. The encoding process consists of the following online and offline phases.

(Offline): Initially, users agree on $N + K + T$ distinct public parameters $\{\alpha_j\}_{j \in [N]}$ and $\{\beta_j\}_{j \in [K+T]}$ from $\mathbb{F}_q$. User $i \in [N]$ then generates $K + T$ random matrices $\mathbf{R}_{ik} \in \mathbb{F}_q^{\frac{|\mathcal{D}_i|}{K} \times d}$ for $k \in [K]$, $\mathbf{V}_{ik} \in \mathbb{F}_q^{\frac{|\mathcal{D}_i|}{K} \times d}$ for $k \in \{K + 1, \ldots, K + T\}$, and forms a Lagrange polynomial of degree $K + T - 1$:

$$\phi_i(z) = \sum_{k \in [K]} \mathbf{R}_{ik} \prod_{l \in [K+T] \setminus \{k\}} \frac{z - \beta_l}{\beta_k - \beta_l} + \sum_{k=K+1}^{K+T} \mathbf{V}_{ik} \prod_{l \in [K+T] \setminus \{k\}} \frac{z - \beta_l}{\beta_k - \beta_l} \quad (7)$$

where the additional randomness $\{\mathbf{V}_{ik}\}_{k \in \{K+1, \ldots, K+T\}}$ is to hide the true contents of $\{\mathbf{R}_{ik}\}_{k \in [K]}$ against up to T adversaries, and sends to each user $j \in [N]$ an *encoded random matrix*,

$$\widetilde{\mathbf{R}}_{ij} \triangleq \phi_i(\alpha_j). \quad (8)$$

The key intuition behind (7) is to perform Lagrange coding on the random matrices, which can be handled offline, as opposed to directly on the data (which should be handled online during training). The Lagrange coded random matrices $\{\widetilde{\mathbf{R}}_{ji}\}_{j \in [N]}$ will later be used to construct a *Lagrange coded dataset*

TABLE II
LIST OF NOTATIONS AND DEFINITIONS

N	Total number of users.
$\mathcal{T}$	Set of adversarial users.
$\mathcal{H} = [N]\backslash\mathcal{T}$	Set of honest users.
T	Maximum number of adversarial users.
S	Maximum number of dropout users per round.
$\mathcal{D}_i$	Local dataset of user $i \in [N]$.
$\mathcal{D} = \mathcal{D}_1 \cup \ldots \mathcal{D}_N$	Collection of the local datasets of all users.
$\mathcal{Y}_i$	Local labels of user $i \in [N]$.
d	Number of features for each data point.
$\mathbf{X}$	$\lvert\mathcal{D}\rvert \times d$ matrix denoting $\mathcal{D}$.
K	Parallelization degree.
$g(x) = 1/(1+e^{-x})$	Sigmoid function.
r	Degree of polynomial approximation.
$\hat{g}(x) = \sum_{i=1}^{r} \theta_i x_i$	Polynomial approximation of the sigmoid.
$\mathbf{w}^{(t)}$	True model parameters at training round t.
$\widetilde{\mathbf{X}}_i$	Encoded dataset for user $i \in [N]$.
$\overline{\mathbf{X}}_i = (\mathbf{X}_{i1}, \ldots, \mathbf{X}_{iK})$	$\lvert\mathcal{D}_i\rvert \times d$ matrix denoting the local dataset $\mathcal{D}_i$.
$\hat{\mathbf{X}}_{ik} = \mathbf{X}_{ik} - \mathbf{R}_{ik}$	Masked dataset of user $i \in [N]$ and $k \in [K]$ with random mask $\mathbf{R}_{ik}$.
$\widetilde{\mathbf{w}}_i^{(t)}$	Encoded model for user $i \in [N]$ at round t.
$\widehat{\mathbf{w}}^{(t)} = \mathbf{w}^{(t)} - \mathbf{r}^{(t)}$	Masked model at training round t with a uniformly random mask $\mathbf{r}^{(t)}$.
$[x]_i$	Shamir's T-out-of-N secret sharing of a secret x at user $i \in [N]$.
$[\mathbf{w}^{(t)}]_i$	Secret share of the true model at user $i \in [N]$ at training round t.
$\mathbb{F}_q$	Finite field of integers modulo a large prime q.
η	Learning rate.
J	Total number of training rounds.

$\widetilde{\mathbf{X}}_i$ at each user $i \in [N]$ as in (5), but with a linear communication overhead. This allows us to decouple the communication-intensive encoding operations from real-time training, by moving the quadratic communication overhead to the offline phase.

(Online): In the online phase, each user $i \in [N]$ partitions its local dataset $\mathcal{D}_i$ into K submatrices $(\mathbf{X}_{i1}, \ldots, \mathbf{X}_{iK})$ each of size $\frac{|\mathcal{D}_i|}{K} \times d$, and broadcasts,

$$\hat{\mathbf{X}}_{ik} = \mathbf{X}_{ik} - \mathbf{R}_{ik} \quad \forall k \in [K]. \tag{9}$$

After receiving $\{\hat{\mathbf{X}}_{1k}, \ldots, \hat{\mathbf{X}}_{Nk}\}_{k\in[K]}$, user i generates an encoded dataset:

$$\widetilde{\mathbf{X}}_i \triangleq \sum_{k\in[K]} \left[\hat{\mathbf{X}}_{1k}^T \cdots \hat{\mathbf{X}}_{Nk}^T\right]^T \prod_{l\in[K+T]\backslash\{k\}} \frac{\alpha_i - \beta_l}{\beta_k - \beta_l} + \left[\widetilde{\mathbf{R}}_{1i}^T \cdots \widetilde{\mathbf{R}}_{Ni}^T\right]^T \tag{10}$$

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

where $\{\widetilde{\mathbf{R}}_{ji}\}_{j\in[N]}$ are defined in (8). The key intuition behind (10) is to simultaneously cancel the additive randomness $\{\mathbf{R}_{jk}\}_{j\in[N],k\in[K]}$, and form a Lagrange polynomial to encode the datasets,

$$\phi(z) \triangleq \sum_{k\in[K]} \mathbf{X}_k \prod_{l\in[K+T]\backslash\{k\}} \frac{z - \beta_l}{\beta_k - \beta_l} + \sum_{k=K+1}^{K+T} \mathbf{V}_k \prod_{l\in[K+T]\backslash\{k\}} \frac{z - \beta_l}{\beta_k - \beta_l} \tag{12}$$

where $\mathbf{X}_k \triangleq \left[\mathbf{X}_{1k}^T \cdots \mathbf{X}_{Nk}^T\right]^T$ and $\mathbf{V}_k \triangleq \left[\mathbf{V}_{1k}^T \cdots \mathbf{V}_{Nk}^T\right]^T$, such that $\phi(\beta_k) = \mathbf{X}_k$ for all $k \in [K]$. The encoded dataset at user $i \in [N]$ then corresponds to $\widetilde{\mathbf{X}}_i = \phi(\alpha_i)$. The additional randomness $\{\mathbf{V}_k\}_{k\in\{K+1,\ldots,K+T\}}$ hides the contents of the datasets against up to T adversaries.

Stage 2: Label Secret Sharing. In order to update the model as in (2), users also need to compute $\mathbf{X}^T\mathbf{y} = \sum_{i\in[N]} \sum_{l\in\mathcal{D}_i} \mathbf{x}_l^T y_l$. In doing so, the computation should not reveal the true content of the labels. In SCALR, this is handled by the following offline and online phases.

(Offline): User $i \in [N]$ generates a random vector $\mathbf{a}_i \in \mathbb{F}_q^{d\times 1}$, and secret shares it with other users, using Shamir's T-out-of-N secret sharing (details of Shamir's secret sharing is available in Appendix A). The secret share sent from user i to user j is denoted by $[\mathbf{a}_i]_j \in \mathbb{F}_q^{d\times 1}$.

(Online): User $i \in [N]$ locally computes $\sum_{l\in\mathcal{D}_i} \mathbf{x}_l^T y_l$, and broadcasts,

$$\hat{\mathbf{a}}_i \triangleq \Big(\sum_{l\in\mathcal{D}_i} \mathbf{x}_l^T y_l\Big) - \mathbf{a}_i. \tag{13}$$

After receiving $\{\hat{\mathbf{a}}_j\}_{j\in[N]}$, user $i \in [N]$ can compute a secret share of $\mathbf{X}^T\mathbf{y}$ as follows:

$$\begin{aligned}[\mathbf{X}^T\mathbf{y}]_i &\triangleq \sum_{j\in[N]} (\hat{\mathbf{a}}_j + [\mathbf{a}_j]_i) \\ &= \sum_{j\in[N]} \Big(\Big(\sum_{l\in\mathcal{D}_j} \mathbf{x}_l^T y_l\Big) - \mathbf{a}_j + [\mathbf{a}_j]_i\Big) \\ &= \Big[\sum_{j\in[N]} \sum_{l\in\mathcal{D}_j} \mathbf{x}_l^T y_l\Big]_i \end{aligned} \tag{14}$$

since summing the shares of multiple variables leads to a secret share of the sum (Appendix A).

Stage 3: Model Initialization. The model at time $t = 0$ (i.e., $\mathbf{w}^{(0)}$) is initialized randomly within $\mathbb{F}_q$. In doing so, to preserve the privacy of intermediate computations, its true value should not be revealed to any user. To do so, each user $i \in [N]$ generates a random vector $\mathbf{w}_i^{(0)} \in \mathbb{F}_q^d$, and secret shares it using Shamir's T-out-of-N secret sharing, where the secret share sent from user i to user j is denoted by $[\mathbf{w}_i^{(0)}]_j$. After receiving $\{[\mathbf{w}_j^{(0)}]_i\}_{j\in[N]}$ user $i \in [N]$ computes,

$$[\mathbf{w}^{(0)}]_i \triangleq \sum_{j\in[N]} [\mathbf{w}_j^{(0)}]_i = [\sum_{j\in[N]} \mathbf{w}_j^{(0)}]_i \tag{15}$$

where $\mathbf{w}^{(0)} = \sum_{i\in[N]} \mathbf{w}_i^{(0)}$ denotes the initialized model at round $t = 0$. At the end of (15), user i obtains a secret share $[\mathbf{w}^{(0)}]_i$ of the initial model $\mathbf{w}^{(0)} = \sum_{i\in[N]} \mathbf{w}_i^{(0)}$, but the real value of $\mathbf{w}_i^{(0)}$ cannot be recovered by any group of up to T users. This stage can be fully carried out offline.

Stage 4: Model Encoding. At each training round, users encode the current state of the model $\mathbf{w}^{(t)}$, to preserve its privacy and enable gradient computations to be performed on encoded data. At the beginning of each round t, user i holds a secret share $[\mathbf{w}^{(t)}]_i$ of $\mathbf{w}^{(t)}$. Initially, at round $t = 0$, $[\mathbf{w}^{(0)}]_i$ is

generated as in (15). For all other rounds (i.e., $t > 0$), $[\mathbf{w}^{(t)}]_i$ is obtained at the end of the model update in (33), which will be described later. Using the secret shares $[\mathbf{w}^{(t)}]_i$, the model $\mathbf{w}^{(t)}$ is encoded as we describe next, where user i learns an encoded model $\widetilde{\mathbf{w}}_i^{(t)}$.

(Offline): User $i \in [N]$ generates $T+1$ random vectors $\mathbf{r}_i^{(t)} \in \mathbb{F}_q^{d\times 1}$ and $\mathbf{v}_{ik}^{(t)} \in \mathbb{F}_q^{d\times 1}$ for $k \in \{K+1, \ldots, K+T\}$, and then secret shares $\mathbf{r}_i^{(t)}$ using Shamir's T-out-of-N secret sharing. The secret share sent from user i to user j is denoted by $[\mathbf{r}_i^{(t)}]_j$. In addition, user $i \in [N]$ *encodes* $\mathbf{r}_i^{(t)}$ by constructing a Lagrange polynomial of degree $K+T-1$,

$$\psi_i^{(t)}(z) \triangleq \sum_{k\in[K]} \mathbf{r}_i^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T} \mathbf{v}_{ik}^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{z-\beta_l}{\beta_k-\beta_l} \tag{16}$$

where $\mathbf{v}_{ik}^{(t)} \in \mathbb{F}_q^d$ are generated uniformly at random, and sends an encoded vector,

$$\widetilde{\mathbf{r}}_{ij}^{(t)} \triangleq \psi_i^{(t)}(\alpha_j) \tag{17}$$

to each user $j \in [N]$. After receiving $\{\widetilde{\mathbf{r}}_{ji}\}_{j\in[N]}$, user i aggregates them,

$$\widetilde{\mathbf{r}}_i^{(t)} = \sum_{j\in[N]} \widetilde{\mathbf{r}}_{ji}^{(t)} = \sum_{k\in[K]} \mathbf{r}^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T} \mathbf{v}_k^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i-\beta_l}{\beta_k-\beta_l} \tag{18}$$

where $\mathbf{r}^{(t)} \triangleq \sum_{j\in[N]} \mathbf{r}_j^{(t)}$ and $\mathbf{v}_k^{(t)} \triangleq \sum_{j\in[N]} \mathbf{v}_{jk}^{(t)}$. Finally, user i aggregates the secret shares $\{[\mathbf{r}_j^{(t)}]_i\}_{j\in[N]}$ received from users $j \in [N]$, to generate a secret share of $\mathbf{r}^{(t)}$ as follows,

$$\sum_{j\in[N]} [\mathbf{r}_j^{(t)}]_i = [\sum_{j\in[N]} \mathbf{r}_j^{(t)}]_i = [\mathbf{r}^{(t)}]_i \tag{19}$$

(Online): In this phase, user i broadcasts a secret share $[\widehat{\mathbf{w}}^{(t)}]_i$ of $\widehat{\mathbf{w}}^{(t)} \triangleq \mathbf{w}^{(t)} - \mathbf{r}^{(t)}$, defined as,

$$[\widehat{\mathbf{w}}^{(t)}]_i \triangleq [\mathbf{w}^{(t)}]_i - [\mathbf{r}^{(t)}]_i = [\mathbf{w}^{(t)} - \mathbf{r}^{(t)}]_i \tag{20}$$

where the last equality follows from the additivity property of Shamir's secret sharing from Appendix A, i.e., summing the shares of two secrets leads to a secret share of their sum. Specifically, by denoting the two secret shares as $[\mathbf{w}^{(t)}]_i = \mathbf{w}^{(t)} + \sum_{l\in[T]} \alpha_i^l \mathbf{n}_l$ and $[\mathbf{r}^{(t)}]_i = \mathbf{r}^{(t)} + \sum_{l\in[T]} \alpha_i^l \mathbf{n}'_l$, where $\mathbf{n}_l \in \mathbb{F}_q^d$ and $\mathbf{n}'_l \in \mathbb{F}_q^d$ are uniformly random vectors, we observe that,

$$[\mathbf{w}^{(t)}]_i - [\mathbf{r}^{(t)}]_i = (\mathbf{w}^{(t)} - \mathbf{r}^{(t)}) + \sum_{l\in[T]} \alpha_i^l (\mathbf{n}_l - \mathbf{n}'_l) = [\mathbf{w}^{(t)} - \mathbf{r}^{(t)}]_i \tag{21}$$

is a share of the secret $\mathbf{w}^{(t)} - \mathbf{r}^{(t)}$, where the secret is hidden by T uniformly random masks $\mathbf{n}_l - \mathbf{n}'_l$ for $l \in [T]$. Accordingly, by broadcasting $[\widehat{\mathbf{w}}^{(t)}]_i = [\mathbf{w}^{(t)}]_i - [\mathbf{r}^{(t)}]_i$, user i broadcasts a secret share of $\mathbf{w}^{(t)} - \mathbf{r}^{(t)}$. Let $\mathcal{U}_1 \subseteq [N]$ denote the set of surviving users at this stage. After receiving $[\widehat{\mathbf{w}}^{(t)}]_i$ from any set $\mathcal{U}_1$ of $|\mathcal{U}_1| \geq T+1$ users, each user can decode:

$$\widehat{\mathbf{w}}^{(t)} = \mathbf{w}^{(t)} - \mathbf{r}^{(t)}. \tag{22}$$

via polynomial interpolation, where the true model $\mathbf{w}^{(t)}$ is hidden by the random vector $\mathbf{r}^{(t)}$. Finally, each user $i \in [N]$ constructs an encoded model:

$$\widetilde{\mathbf{w}}_i^{(t)} \triangleq \sum_{k\in[K]} \widehat{\mathbf{w}}^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i-\beta_l}{\beta_k-\beta_l} + \widetilde{\mathbf{r}}_i^{(t)} \tag{23}$$

$$= \sum_{k\in[K]} \mathbf{w}^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i-\beta_l}{\beta_k-\beta_l} + \sum_{k=K+1}^{K+T} \mathbf{v}_k^{(t)} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_i-\beta_l}{\beta_k-\beta_l} \tag{24}$$

where the T random vectors $\{\mathbf{v}_k^{(t)}\}_{k\in\{K+1,\ldots,K+T\}}$ hide the contents of $\mathbf{w}^{(t)}$ against up to T adversaries. Intuitively, (24) embeds the model $\mathbf{w}^{(t)}$ in a Lagrange polynomial of degree $K+T-1$,

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

and at the end user i obtains an encoded model $\widetilde{\mathbf{w}}_i^{(t)} = \psi^{(t)}(\alpha_i)$.

Stage 5: Gradient Computing and Model Update. The last component of SCALR is to compute the gradients and update the model. As detailed in Section III, Lagrange coding is bound to polynomial computations. On the other hand, the sigmoid function from (1) is not a polynomial, hence is often approximated using a polynomial function $\hat{g}(x) = \sum_{i=0}^{r} \theta_i x^i$, where coefficients $\{\theta_i\}_{i\in[r]}$ are public parameters fitted via least squares [5]. The degree r quantifies the accuracy of approximation [38]. Given $\hat{g}(\cdot)$, the model update from (2) can be rewritten as:

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

Then, gradient computing and model updates consist of the following online and offline phases.

(Offline): User $i \in [N]$ generates a uniformly random vector $\mathbf{u}_i^{(t)} \in \mathbb{F}_q^d$, and secret shares it via Shamir's T-out-of-N secret sharing, where the secret share sent to user j is denoted by $[\mathbf{u}_i^{(t)}]_j$.

(Online): In the online phase, user $i \in [N]$ locally computes a gradient,

$$f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)}) \triangleq \widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)}) \tag{27}$$

using the coded dataset $\widetilde{\mathbf{X}}_i$ and model $\widetilde{\mathbf{w}}_i^{(t)}$, and broadcasts,

$$\hat{\mathbf{u}}_i^{(t)} \triangleq f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)}) - \mathbf{u}_i^{(t)} = \widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)}) - \mathbf{u}_i^{(t)} \tag{28}$$

where the true content of $\widetilde{\mathbf{X}}_i^T \hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)})$ is hidden by the random vector $\mathbf{u}_i^{(t)}$. Next, users decode the gradients and

Algorithm 1 SCALR - Offline Phase

Input: Number of users N, polynomial coefficients $(\alpha_1,\dots,\alpha_N),(\beta_1,\dots,\beta_K)$.
Output: Random vectors $\{\widetilde{\mathbf{R}}_{ij},[\mathbf{a}_i]_j\}_{i,j\in[N]}$, $\{\widetilde{\mathbf{r}}_i^{(t)},[\mathbf{r}^t]_i,[\mathbf{u}_i^{(t)}]_j\}_{i\in[N],t\in\{0,\dots,J-1\}}$, and $\{[\mathbf{w}^{(0)}]_i\}_{i\in[N]}$.

for *user* $i=1,\dots,N$ **do**
  Encode random matrices $\{\mathbf{R}_{ik}\}_{k\in[K]},\{\mathbf{V}_{ik}\}_{k\in\{K+1,\dots,K+T\}}$ from (7), send the encoded matrix $\widetilde{\mathbf{R}}_{ij}$ to user $j\in[N]$.
for *user* $i=1,\dots,N$ **do**
  Generate a random vector $\mathbf{a}_i$ from $\mathbb{F}_q$, send a secret share $[\mathbf{a}_i]_j$ to user $j\in[N]$.
for *user* $i=1,\dots,N$ **do**
  Generate a random vector $\mathbf{w}_i^{(0)}$ from $\mathbb{F}_q$, send a secret share $[\mathbf{w}_i^{(0)}]_j$ to user $j\in[N]$
for *user* $i=1,\dots,N$ **do**
  Initialize the model $[\mathbf{w}^{(0)}]_i=\sum_{j\in[N]}[\mathbf{w}_j^{(0)}]_i$ as given in (15).
for *iteration* $t=0,\dots,J-1$ **do**
  for *user* $i=1,\dots,N$ **do**
    Encode the random vectors $\{\mathbf{r}_{ik}^{(t)}\}_{k\in[K]}$, $\{\mathbf{v}_{ik}^{(t)}\}_{k\in\{K+1,\dots,K+T\}}$ from (17).
    for $j=1,\dots,N$ **do**
      Send the encoded vector $\widetilde{\mathbf{r}}_{ij}^{(t)}$ and secret share $[\mathbf{r}_i^{(t)}]_j$ to user j.
  for *user* $i=1,\dots,N$ **do**
    Aggregate the coded vectors $\widetilde{\mathbf{r}}_i^{(t)}=\sum_{j\in[N]}\widetilde{\mathbf{r}}_{ji}^{(t)}$ and secret shares $[\mathbf{r}^{(t)}]_i=\sum_{j\in[N]}[\mathbf{r}_j^{(t)}]_i$ as in (18), (19).
  for *user* $i=1,\dots,N$ **do**
    Generate a random vector $\mathbf{u}_i^{(t)}$ from $\mathbb{F}_q$, and send a secret share $[\mathbf{u}_i^{(t)}]_j$ to user $j\in[N]$.

Algorithm 2 SCALR - Online Phase

Input: Dataset $(\mathcal{D},\mathbf{y})=((\mathcal{D}_1,\mathbf{y}_1),\dots,(\mathcal{D}_N,\mathbf{y}_N))$ distributed across N users.
Output: Model parameters $\mathbf{w}^{(J)}$ after J training rounds.

for *user* $i=1,\dots,N$ **do**
  Partition $\mathcal{D}_i$ into K shards $(\mathbf{X}_{i1},\dots,\mathbf{X}_{iK})$, broadcast the masked dataset $\hat{\mathbf{X}}_{ik}=\mathbf{X}_{ik}-\mathbf{R}_{ik}$ for $k\in[K]$.
for *user* $i=1,\dots,N$ **do**
  Generate the coded dataset $\widetilde{\mathbf{X}}_i$ as given in (10).
for *user* $i=1,\dots,N$ **do**
  Broadcast $\left(\sum_{l\in\mathcal{D}_i}\mathbf{x}_l^T y_l\right)-\mathbf{a}_i$.
for *user* $i=1,\dots,N$ **do**
  Compute a secret share $[\mathbf{X}^T\mathbf{y}]_i=\sum_{j\in[N]}(\hat{\mathbf{a}}_j+[\mathbf{a}_j^{(t)}]_i)$ of the labels as given in (14).
for *iteration* $t=0,\dots,J-1$ **do**
  for $i=1,\dots,N$ **do**
    Broadcast $[\widehat{\mathbf{w}}^{(t)}]_i$ from (20).
  for $i=1,\dots,N$ **do**
    Decode $\widehat{\mathbf{w}}^{(t)}\triangleq\mathbf{w}^{(t)}-\mathbf{r}^{(t)}$ using polynomial interpolation, compute the encoded model $\widetilde{\mathbf{w}}_i^{(t)}$.
  for *user* $i=1,\dots,N$ **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 (27), broadcast $\hat{\mathbf{u}}_i^{(t)}=\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i\times\widetilde{\mathbf{w}}_i^{(t)})-\mathbf{u}_i^{(t)}$ in (28).
  for *user* $i=1,\dots,N$ **do**
    Decode the gradient $[\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})]_i$ as given in (32), update the model as given in (33).

update the model, but without learning their true content. This is achieved by polynomial interpolation, where we define a polynomial $h^{(t)}(z)=f(\phi(z),\psi^{(t)}(z))$ such that,

$$\begin{aligned}h^{(t)}(\alpha_i)&=f(\phi(\alpha_i),\psi^{(t)}(\alpha_i))\\&=f(\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{w}}_i^{(t)})=\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i\times\widetilde{\mathbf{w}}^{(t)})\end{aligned}\tag{29}$$

$$\begin{aligned}h^{(t)}(\beta_k)&=f(\phi(\beta_k),\psi^{(t)}(\beta_k))\\&=f(\mathbf{X}_k,\mathbf{w}^{(t)})=\mathbf{X}_k^T\hat{g}(\mathbf{X}_k\times\mathbf{w}^{(t)})\text{ for }k\in[K]\end{aligned}\tag{30}$$

Let $\mathcal{U}_2\subseteq\mathcal{U}_1\subseteq[N]$ denote the set of surviving users at this stage. Then, after receiving $\hat{\mathbf{u}}_j^{(t)}$ from any set of users $j\in\mathcal{U}_2$ of size at least $|\mathcal{U}_2|\geq\deg(h)+1=(2r+1)(K+T-1)+1$, user i can compute a secret share of $f(\mathbf{X}_k,\mathbf{w}^{(t)})=\mathbf{X}_k^T\hat{g}(\mathbf{X}_k\times\mathbf{w}^{(t)})$ as follows,

$$[f(\mathbf{X}_k,\mathbf{w}^{(t)})]_i\triangleq\sum_{j\in\mathcal{I}}(\hat{\mathbf{u}}_j^{(t)}+[\mathbf{u}_j^{(t)}]_i)\prod_{l\in\mathcal{I}\setminus\{j\}}\frac{\beta_k-\alpha_l}{\alpha_j-\alpha_l}\forall k\in[K],\tag{31}$$

and then sum them up to obtain a secret share of $\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})=\sum_{i\in[N]}\mathbf{X}_i^T\hat{g}(\mathbf{X}_i\times\mathbf{w}^{(t)})$,

$$\begin{aligned}\sum_{k\in[K]}[f(\mathbf{X}_k,\mathbf{w}^{(t)})]_i&=[\sum_{k\in[K]}f(\mathbf{X}_k,\mathbf{w}^{(t)})]_i\\&=[\sum_{i\in[N]}\mathbf{X}_i^T\hat{g}(\mathbf{X}_i\times\mathbf{w}^{(t)})]_i\\&=[\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})]_i\end{aligned}\tag{32}$$

After computing the gradient, users update the model. Note that SCALR relies on finite field polynomial operations, bound to finite field addition and multiplications. Conversely, the model update in (2) requires a division. To handle this, one approach is to treat model updating as an integer operation (assuming a large field size), as detailed in Appendix B. In practice, one can also apply the secure truncation protocol from [39] and [5] to update the model according to (26),

$$[\mathbf{w}^{(t+1)}]_i=[\mathbf{w}^{(t)}]_i-\frac{\eta}{|\mathcal{D}|}([\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})]_i-[\mathbf{X}^T\mathbf{y}]_i),\tag{33}$$

where η is selected such that $\frac{|\mathcal{D}|}{\eta}\in\mathbb{F}_q$, at the end of which user i obtains a secret share $[\mathbf{w}^{(t+1)}]_i$ of the updated model $\mathbf{w}^{(t+1)}$ for the next training round. This protocol takes as input the secret shares $\{[z]_i\}_{i\in[N]}$ of a variable z (client i holds a share $[z]_i$), and two public integer parameters p_1 and

p_2 such that $0 < p_1 < p_2$, and $z \in \mathbb{F}_{2^{p_2}}$. Then, the protocol returns the secret shares of the variable $\lfloor \frac{z}{2^{p_1}} \rfloor + u$ where u is a binary random variable with probability $P[u = 1] = (z \mod 2^{p_1})/2^{p_1}$. This quantization is unbiased, ensuring the convergence of training [5].

Final Model. After J training rounds, the final model $\mathbf{w}^{(J)}$ is decoded by collecting the secret shares $[\mathbf{w}^{(J)}]_i$ from any set of $T+1$ users, and using polynomial interpolation. The offline and online components of SCALR are presented in Algorithms 1 and 2, respectively.

Remark 1: The use of Shamir's secret sharing in model encoding/update builds on two key intuitions: 1) In contrast to dataset encoding where the individual users can observe their local datasets in the clear, the true model should be kept private from all users, during both model encoding and update stages throughout the training. 2) During the model update, gradients corresponding to different data points, including those that are mapped to different Lagrange coefficients in the coded dataset, should be accumulated. In our framework, this is handled by using Shamir's secret sharing, to decode and sum up the gradients evaluated at different Lagrange coefficients, without revealing their true value to any user.

V. Theoretical Analysis

A. Privacy

We first present the information-theoretic privacy guarantees of SCALR from (3).

Theorem 1: (Information-theoretic privacy) SCALR guarantees information theoretic-privacy:

$$I(\{\mathcal{X}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_\mathcal{T} | \{\mathcal{X}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) = 0 \qquad (34)$$

against any set of adversaries $\mathcal{T} \subseteq [N]$ such that $|\mathcal{T}| \leq T$, where $\mathcal{M}_\mathcal{T}$ denotes the collection of all messages held (received or generated) by the adversaries.

Proof: The proof is provided in Appendix B. □

B. Communication and Computation Complexity

We next analyze the communication and computation complexity of SCALR. For clarity, we let $|\mathcal{D}_i| = D$ for all $i \in [N]$, to explicitly demonstrate the complexity with respect to the number of users. The total communication and computation complexity of SCALR is given in Table III.

Theorem 2: (Communication complexity) SCALR incurs a per-user communication overhead of $O(dD + dJ))$ in the online phase, and $O(Nd\frac{D}{K} + NdJ)$ in the offline phase.

Proof: Offline per-user overhead consists of: 1) $O(Nd\frac{D}{K})$ for dataset encoding, 2) $O(Nd)$ for label secret sharing, 3) $O(Nd)$ for model initialization, 4) $O(Nd)$ for model encoding per training round, 5) $O(Nd)$ for gradient computing and model update per round. All communications are point-to-point. Online per-user overhead includes: 1) $O(dD)$ for dataset encoding, 2) $O(d)$ for label secret sharing, 3) $O(d)$ for model initialization, 4) $O(d)$ for model encoding per round, 5) $O(d)$ for gradient computing and model update per round. All communications are broadcast. □

Theorem 3: (Computation complexity) The per-user computation complexity of SCALR is given by $O(NDd + J\frac{ND}{K}(d + r) + Jdr(K+T)\log^2 r(K+T)\log\log r(K+T))$ for the online phase, and $O(N^2 d(\frac{D}{K} + J)\log^2(K+T)\log\log(K+T))$ for the offline phase.

Proof: (Offline) Interpolating a polynomial of degree κ (and evaluating it at κ points) has a computational complexity of $O(\kappa \log^2 \kappa \log\log \kappa)$ [40]. Then, the per-user complexity of the offline phase is: 1) $O(Nd\frac{D}{K}\log^2(K+T)\log\log(K+T))$ to generate $\{\widetilde{\mathbf{R}}_{ij}\}_{j\in[N]}$, 2) $O(Nd\log^2 T\log\log T)$ to compute the secret share $\{[\mathbf{a}_i]_j\}_{j\in[N]}$, 3) $O(Nd\log^2 T\log\log T)$ to compute the secret share $\{[\mathbf{w}_i^{(0)}]_j\}_{j\in[N]}$; and $O(Nd)$ to compute $[\mathbf{w}^{(0)}]_i$, 4) $O(Nd\log^2(K+T)\log\log(K+T))$ to compute $\widetilde{\mathbf{r}}_{ij}^{(t)}$; $O(Nd)$ to compute $\widetilde{\mathbf{r}}_i^{(t)} = \sum_{j\in[N]}\widetilde{\mathbf{r}}_{ji}^{(t)}$; $O(Nd\log^2 T\log\log T)$ for constructing the secret shares $\{[\mathbf{r}_i^{(t)}]_j\}_{j\in[N]}$; $O(Nd)$ to sum the secret shares $\{[\mathbf{r}_j^{(t)}]_i\}_{j\in[N]}$ (per training round), 5) $O(Nd\log^2 T\log\log T)$ to compute the secret share $\{[\mathbf{u}_i^{(t)}]_j\}_{j\in[N]}$ (per training round).

(Online) The per-user complexity of the online phase is: 1) $O(Dd)$ for computing $\hat{\mathbf{X}}_i$; $O(NDd)$ for computing $\widetilde{\mathbf{X}}_i$, 2) $O(Dd)$ for computing $\hat{\mathbf{a}}_i$; $O(Nd)$ for computing $[\mathbf{X}^T\mathbf{y}]_i = \sum_{j\in[N]}(\hat{\mathbf{a}}_j + [\mathbf{a}_j]_i)$, 4) $O(Td\log^2 T\log\log T)$ for computing $\hat{\mathbf{w}}^{(t)}$; $O(Kd)$ for computing $\widetilde{\mathbf{w}}_i$ (per round), 5) $O(\frac{ND}{K}(d + r))$ to compute $\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)})$; $O(d)$ for computing $\hat{\mathbf{u}}_i$; $O(dr(K+T)\log^2 r(K+T)\log\log r(K+T))$ for decoding $[f(\mathbf{X}_k, \mathbf{w}^{(t)})]_i$; $O(Kd)$ to compute $[\mathbf{X}^T\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})]_i$; $O(d)$ to update the model (per round). □

C. Recovery Threshold

The recovery threshold is defined as the minimum number of surviving users that are needed for correct recovery of the final model. The recovery threshold of SCALR is $N - S \geq (2r+1)(K+T-1)+1$ (equal to COPML), as model update requires local computations to be collected from at least $|\mathcal{U}_2| \geq (2r+1)(K+T-1)+1$ surviving users.

Remark 2: SCALR can also be applied to simpler linear regression, following the same steps.

D. Relation to Secure Aggregation (SA)

Both SCALR and SA [19] leverage offline generation/encoding of random masks. The key difference is that, in SA each user knows the true mask that hides their local model/gradient, and learns the updated model (and the aggregate of the gradients) after each training round, while in SCALR the updated model should stay private throughout the training, and the true masks that hide the model can not be known by any user. The two frameworks provide different benefits and trade-offs; SA can be applied to highly complex training tasks, but as the intermediate model parameters are revealed after each training iteration, privacy degrades as the number of rounds increase [21], and can be breached through *multi-round privacy attacks* [20]. In contrast, SCALR reveals no intermediate model or gradient parameters during training, preventing such privacy degradation throughout training.

TABLE III
TOTAL COMMUNICATION AND COMPUTATION COMPLEXITY (ACROSS ALL USERS) OF SCALR

	Communication	Computation
Online	$O(N(dD+dJ))$	$O(N^2Dd+N^2J\frac{D}{K}(d+r)+NJdr(K+T)\log^2 r(K+T)\log\log r(K+T))$
Offline	$O(N^2(d\frac{D}{K}+dJ))$	$O(N^2d(\frac{D}{K}+J)\log^2(K+T)\log\log(K+T))$

E. Randomness Generation

We next present the volume of randomness generated per user for SCALR. For dataset encoding (Stage 1), each user $i \in [N]$ generates $O(\frac{|\mathcal{D}_i|d(K+T)}{K})$ random parameters. For label secret sharing (Stage 2) and model initialization (Stage 3), each user generates $O(dT)$ random parameters. Then, at each training round, each user generates $O(dT)$ random parameters for model encoding (Stage 4) and model update (Stage 5). In comparison, the randomness generated per user for secure aggregation [19] is $O(\frac{d(N-S)}{N-S-T})$ per training round.

VI. EXPERIMENTS

A. Setup

We consider logistic regression for binary classification on two image datasets chosen in accordance with [5]: CIFAR-10 [41] and GISETTE [42], which are of size $(|\mathcal{D}|, d) = (9019, 3073)$ and $(6000, 5000)$ respectively. The datasets are distributed evenly across the users. We implement a multi-user network where communication between the users are carried out through a Message Passing Interface (MPI) using the `MPI4Py` Python programming tool [43].

B. 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 (beyond 3-4 users), which is the COPML framework from [5]. We measure both the communication volume, and the wall-clock training time, including both communication and computation. The communication volume includes *all protocol stages*. Note that the experimental results in [5] do not include the one-time operations, i.e., secret sharing the datasets and labels. As they are also data-dependent, we include them here. For both frameworks, we leverage the secure quantization operation to avoid overlap errors during model updating, with $(p_1, p_2) = (21, 24)$ [5]. We further optimize (speed-up) COPML using the grouping strategy suggested in [5], which partitions users into groups of size $T+1$, and communicates secret shares only between users within the same group.

C. Hyperparameters

To demonstrate the performance under the same experimental settings with prior work, the average communication bandwidth is set to 40Mbps, finite field size is set to $q = 2^{26} - 5$, along with $r = 1$ and $J = 50$. [5].

D. Performance Evaluation

We first consider the scenario where the degree of privacy (T) and parallelism (K) are (almost) equal, by letting $N = 3(K+T-1)+1$ with $T = \lfloor \frac{N-3}{6} \rfloor$ and $K = \lfloor \frac{N+2}{3} \rfloor - T$. In Figs. 2(a)-2(b), we compare the total communication overhead during training (i.e., online communication overhead) of SCALR with COPML (where all communication is online). We observe that SCALR significantly decreases the online communication overhead, by up to $80\times$ and $70\times$ on the CIFAR-10 and GISETTE datasets, respectively. We also note that the broadcast functionality of the MPI protocol communicates messages through a tree topology. As such, the communication overhead observed for SCALR scales with respect to $O(N \log N)$. In an ideal broadcasting scenario (e.g., a cellular network), one can expect further gains (approaching $O(N)$). We next compare the wall-clock online training time (per user) of the two frameworks in Figs. 2(c)-2(d). We observe that SCALR reduces the training time by up to $6.0\times$ and $5.8\times$ on the CIFAR-10 and GISETTE datasets, respectively. In Table IV we also present the wall-clock time for communication and computation in the offline and online phases per user, respectively.

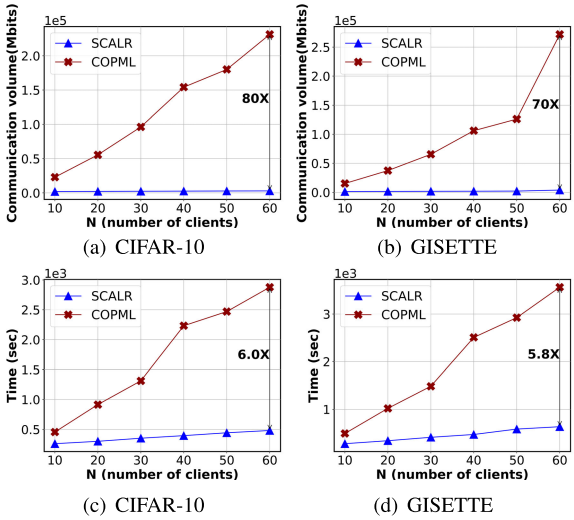


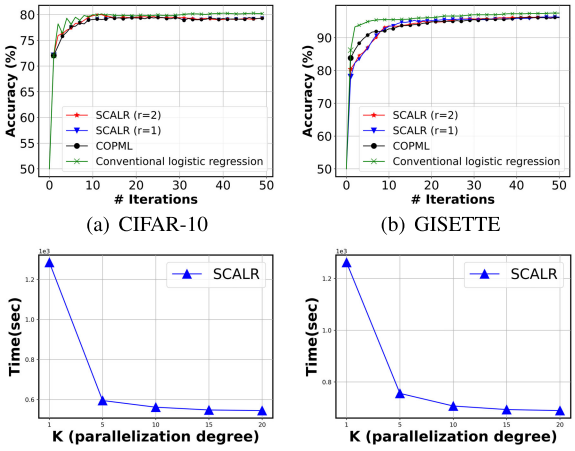
Fig. 2. Online communication volume (Fig. 2(a)-2(b) and wall-clock training time (Fig. 2(c)-2(d)) for SCALR and COPML.

TABLE IV
COMMUNICATION AND COMPUTATION TIME (IN SECONDS) PER USER FOR N=60

CIFAR-10	**Offline**	**Online**
Communication	3458.43	623.92
Computation	20.36	61.59
GISETTE	**Offline**	**Online**
Communication	2190.25	463.79
Computation	17.83	51.74

E. Accuracy

We next demonstrate the model convergence for SCALR, COPML, and conventional logistic regression (which



(a) CIFAR-10 (b) GISETTE

(c) CIFAR-10 (d) GISETTE

Fig. 3. Model convergence (Figs. 3(a)-3(b)), and online training time of SCALR for varying K (Figs. 3(c)-3(d)) with $N = 60$.

represents the target accuracy, without any privacy constraints) in Figs. 3(a)-3(b) for $N = 60$. We observe that SCALR achieves comparable model accuracy to COPML and conventional centralized logistic regression.

F. Varying the Parallelism Degree

One of the key system parameters is K (degree of parallelism), as the training load per user scales with respect to $1/K$. In Figs. 3(c)-3(d), we present the role of K on the wall-clock training time, by fixing the number of users to $N = 60$, and varying K. We observe that increasing K decreases the training time, demonstrating the trade-off between parallelism (accordingly, the training time) and adversary resilience, as increasing K decreases the maximum number of adversaries (i.e., T) that the protocol is resilient against.

G. Varying the Degree of Polynomial Approximation for the Sigmoid

In Figs. 3(a) and 3(b), we present the model accuracy for SCALR with both $r = 1$ and $r = 2$, where we observe that the two settings have comparable accuracy. To determine the range of r needed for more complex scenarios, one approach is to let each user initially select a local value that meets their minimum performance requirement on their local dataset, and then agree on a common parameter r by selecting the maximum of the locally selected parameters.

VII. Conclusion and Future Work

In this work, we introduce SCALR, a fast and scalable framework for logistic regression with end-to-end information-theoretic privacy. SCALR builds on an offline-online communication trade-off, where the online communication overhead is only linear in the number of users, and the offline communication is data-agnostic, and can take place anytime when the network load is low. Our experiments demonstrate an order of magnitude reduction in the communication overhead, while achieving the same performance guarantees with the state-of-the-art.

Future directions include extending our framework to complex learning tasks, such as neural networks, by using polynomial approximations for non-linear functions such as ReLU and softmax. Different privacy trade-offs can be provided by leveraging relaxed privacy notions, such as statistical security [39], where the leakage probability is quantified by a security parameter. Another interesting direction is to integrate our framework with complementary differential privacy (DP) techniques [22], [25], [27], to prevent potential information leakage from the final model [44], [45], [46], [47], [48]. Interestingly, doing so can further improve the model accuracy for DP in distributed settings [47], [48]. Another important direction is reducing the computational load of decoding for low-powered devices, by integrating our approach with gradient/model quantization, compression, and pruning. In doing so, one can potentially reduce the dimensionality of the gradients/model parameters, as well as the required field size for computational efficiency.

Appendix A
Shamir's Secret Sharing Protocol and Degree Reduction

Shamir's T-out-of-N secret sharing [8] embeds a secret s to a degree T polynomial $f(\xi) = s + \xi n_1 + \ldots + \xi^T n_T$, where $\{n_k\}_{k\in[T]} \in \mathbb{F}_q$ are generated uniformly random, and sends a share $[s]_i \triangleq f(\alpha_i)$ to each user $i \in [N]$, where $\{\alpha_i\}_{i\in[N]}$ are distinct parameters in $\mathbb{F}_q$. The secret s can be recovered from any collection of $T+1$ shares, but any collection of T or fewer shares reveals no information about s. Shamir's secret sharing supports addition and multiplication operations.

Addition: To compute the sum $s + s'$ of two secrets s and s', user i sums the secret shares:

$$\begin{aligned}[s]_i + [s']_i &= (s + \alpha_i n_1 + \ldots + \alpha_i^T n_T) \\ &\quad + (s' + \alpha_i n'_1 + \ldots + \alpha_i^T n'_T) \qquad (35)\\ &= (s + s') + \alpha_i(n_1 + n'_1) + \ldots + \alpha_i^T(n_T + n'_T) \\ &= [s + s']_i \qquad (36)\end{aligned}$$

where the result $[s + s']_i$ is a secret share of $s + s'$. This operation requires no communication.

Multiplication-by-a-constant: To multiply s with a public constant c, user i locally computes $c[s]_i = [cs]_i$, which results in a secret share of cs.

Multiplication and degree reduction: For computing the product ss' of two secrets, user i initially multiplies the secret shares $g_i \triangleq [s]_i \times [s']_i = ss' + \ldots + \alpha_i^{2T}(n_T n'_T)$, where the resulting polynomial has degree $2T$. As such, recovering xx' with polynomial interpolation requires collecting the shares from at least $2T + 1$ users. Each successive multiplication further increases the degree, hence the minimum number of users required, necessitating a degree reduction step to avoid a *degree explosion* [7]. Note that ss' can be written as a linear function of $2T+1$ evaluation points, hence $ss' = \sum_{i=1}^{2T+1} \lambda_i g_i$ for some $\{\lambda_i\}_{i\in[2T+1]}$. To perform degree reduction, user i

then secret shares $g_i = [s]_i[s']_i$ using Shamir's T-out-of-N secret sharing. After receiving $\{[g_i]_j\}_{i\in[N]}$, user j can compute a new secret share $[ss']_j = \sum_{i=1}^{2T+1}\lambda_i[g_i]_j = [\sum_{i=1}^{2T+1}\lambda_i g_i]_j$ of ss', where the new share $[ss']_j$ corresponds to a polynomial of degree T (as opposed to $2T$). On the other hand, this has a quadratic communication complexity, which is $O(N^2)$ over N users.

Appendix B
Information-Theoretic Privacy

Proof: For tractability of the theoretical analysis, here we consider a sufficiently large field size, and treat all model updates as integer operations. This can be achieved by letting $\bar{\eta} \triangleq |\mathcal{D}|/\eta$ be an integer, and re-defining the local computation from (27) performed by user i as,

$$f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)}) = \sum_{j=0}^{r} \theta_j \bar{\eta}^{(r-j)c_t} \widetilde{\mathbf{X}}_i^T (\widetilde{\mathbf{X}}_i \times \widetilde{\mathbf{w}}_i^{(t)})^j \tag{37}$$

where the operation $(\cdot)^j$ stands for element-wise exponentiation, and,

$$c_t \triangleq \begin{cases} 0 & \text{for } t = 0 \\ rc_{t-1} + 1 & \text{for } t \geq 1 \end{cases} \tag{38}$$

Equation (37) represents evaluations of a univariate polynomial $h(z) = f(\phi(z), \psi(z))$ such that, $h(\alpha_i) = f(\phi(\alpha_i), \psi(\alpha_i)) = f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)})$ and

$$\begin{aligned} h(\beta_k) &= f(\phi(\beta_k), \psi(\beta_k)) = f(\mathbf{X}_k, \mathbf{w}^{(t)}) \\ &= \sum_{j=0}^{r} \theta_j \bar{\eta}^{(r-j)c_t} \mathbf{X}_k^T (\mathbf{X}_k \times \mathbf{w}^{(t)})^j \quad \forall k \in [K]. \end{aligned} \tag{39}$$

After receiving $\hat{\mathbf{u}}_i = f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)}) - \widetilde{\mathbf{u}}_i$ from (28), user i computes (31) to obtain a secret share $[f(\mathbf{X}, \mathbf{w}^{(t)})]_i \triangleq \sum_{k\in[K]}[f(\mathbf{X}_k, \mathbf{w}^{(t)})]_i$ of the true gradient $f(\mathbf{X}, \mathbf{w}^{(t)}) = \sum_{k\in[K]} f(\mathbf{X}_k, \mathbf{w}^{(t)}) = \sum_{k\in[K]} h(\beta_k)$. Finally, the model update operation from (33) can be re-defined as,

$$[\mathbf{w}^{(t+1)}]_i = \bar{\eta}^{(r-1)c_t+1}[\mathbf{w}^{(t)}]_i - ([f(\mathbf{X}, \mathbf{w}^{(t)})]_i - \bar{\eta}^{rc_t}[\mathbf{X}^T\mathbf{y}]_i). \tag{40}$$

After J rounds, users can decode $\mathbf{w}^{(J)}$ by collecting $\{[\mathbf{w}^{(J)}]_i\}_{i\in[N]}$, and recover the final model as $\mathbf{w}^{(J)} \leftarrow \mathbf{w}^{(J)}/\bar{\eta}^{c_J}$. The correctness of this update process is presented in Appendix C.

Privacy. We now proceed with the privacy analysis. Consider an arbitrary set of adversaries $\mathcal{T} \subseteq N$. For ease of exposition, we focus on the worst case scenario $|\mathcal{T}| = T$, while the same analysis holds for all $|\mathcal{T}| < T$. Let $\mathcal{M}_\mathcal{T}^j$ denote the messages collected by the adversaries in Stages $j \in \{1, 2, 3\}$, and $\mathcal{M}_\mathcal{T}^{j,t}$ denote the messages received by the adversaries in Stages $j \in \{4, 5\}$ at training round $t \in \{0, \ldots, J-1\}$, respectively. Then, from the chain rule of mutual information [49], one can rewrite the mutual information condition from (34) as follows:

$$\begin{aligned} &I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_\mathcal{T} | \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \\ &= I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_\mathcal{T}^1, \mathcal{M}_\mathcal{T}^2, \mathcal{M}_\mathcal{T}^3, \cup_{t\in[J]}\mathcal{M}_\mathcal{T}^{4,t}, \cup_{t\in[J]}\mathcal{M}_\mathcal{T}^{5,t} | \\ &\qquad \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \end{aligned} \tag{41}$$

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

We next investigate each term in the summation (43).

Stage 1: Dataset Encoding. First, we start with the first term in (43), which corresponds to the first stage, dataset encoding of SCALR. This term can be written as:

$$\begin{aligned} &I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_\mathcal{T}^1 | \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \\ &= I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \\ &\quad \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}} | \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \\ &= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \\ &\qquad \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}} | \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \\ &\quad - H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \\ &\qquad \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}} | \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) \end{aligned} \tag{44, 45}$$

$$\begin{aligned} &= H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \\ &\quad \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in\mathcal{H}\\ k\in[K]}} | \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \\ &\quad - H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}) \end{aligned} \tag{46}$$

$$\begin{aligned} &\leq H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}}, \{\mathbf{R}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[K]}}, \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \\ &\quad \{\hat{\mathbf{X}}_{ik}\}_{\substack{i\in\mathcal{H}\\ k\in[K]}}) - H(\{\widetilde{\mathbf{R}}_{ij}\}_{\substack{j\in\mathcal{T}\\ i\in\mathcal{H}}} | \{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}, \\ &\quad \{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}) - H(\{\mathbf{V}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}) \\ &\qquad - H(\{\mathbf{R}_{ik}\}_{\substack{i\in[N]\\ k\in[K]}}) \end{aligned} \tag{47}$$

$$\begin{aligned} &\leq d\Big(\frac{T}{K} + 1\Big)\Big(\sum_{i\in[N]} |\mathcal{D}_i|\Big)\log q - \sum_{i\in\mathcal{H}} H(\{\mathbf{Z}_{ij}\}_{j\in\mathcal{T}}) \\ &\quad - \frac{Td}{K}(\sum_{i\in\mathcal{T}} |\mathcal{D}_i|)\log q - d(\sum_{i\in[N]} |\mathcal{D}_i|)\log q \end{aligned} \tag{48}$$

$$\leq 0 \tag{49}$$

where (46) holds since given $\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in[N]}$, there is no uncertainty in $\{\mathbf{X}_{ik}\}_{i\in[N],k\in[K]}$, and that the random matrices are independent; (47) is from the chain rule of entropy, and the fact that conditioning cannot increase entropy; (48) holds since uniform distribution maximizes entropy, which is equal to

$\log|\mathcal{A}|$ over an alphabet $\mathcal{A}$ [49]. In (48), we define $\mathbf{Z}_{ij} \triangleq \sum_{k=K+1}^{K+T} \mathbf{V}_{ik} \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j-\beta_l}{\beta_k-\beta_l}$ for all $i \in \mathcal{H}, j \in \mathcal{T}$. For simplicity, in the following we let $\mathcal{T} = [T]$ and $\mathcal{H} = \{T+1,\ldots,N\}$, hence the first T users are adversarial, while noting that the same analysis holds for any arbitrary set of adversaries. Then, we let:

$$\lambda_{jk} \triangleq \prod_{l\in[K+T]\setminus\{k\}} \frac{\alpha_j-\beta_l}{\beta_k-\beta_l} \tag{50}$$

for all $j \in [N]$ and $k \in [K+T]$, from which one can write,

$$(\mathbf{Z}_{i1},\ldots,\mathbf{Z}_{iT}) = (\mathbf{V}_{i,K+1},\ldots,\mathbf{V}_{i,K+T}) \underbrace{\begin{bmatrix} \lambda_{1,K+1} & \cdots & \lambda_{T,K+1} \\ \vdots & \ddots & \vdots \\ \lambda_{1,K+T} & \cdots & \lambda_{T,K+T} \end{bmatrix}}_{\mathbf{M}} \tag{51}$$

where $\mathbf{M}$ is a $T\times T$ MDS matrix, hence is invertible [1]. As a result,

$$H(\{\mathbf{Z}_{ij}\}_{j\in\mathcal{T}}) = H(\mathbf{Z}_{i1},\ldots,\mathbf{Z}_{iT}) = H(\mathbf{V}_{i,K+1},\ldots,\mathbf{V}_{i,K+T}) = \frac{Td|\mathcal{D}_i|}{K}\log q \tag{52}$$

where (52) follows from (51) and that $\mathbf{M}$ is an MDS matrix. Finally, from (49) we observe:

$$0 \le I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^1 | \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \le 0 \tag{53}$$

where the first inequality follows from the non-negativity of mutual information. Therefore, the first term in (43) satisfies $I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^1 | \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) = 0$.

Stage 2: Label Secret Sharing. We next consider the second term in (43), which corresponds to the secret sharing of the labels. Denote the secret share of $\mathbf{a}_i$ from user i to user j as:

$$[\mathbf{a}_i]_j \triangleq \mathbf{a}_i + \sum_{k\in[T]} \gamma_j^k \mathbf{b}_{ik} \tag{54}$$

where $\mathbf{b}_{ik} \in \mathbb{F}_q^{d\times 1}$ are random vectors for $k \in [T]$. Coefficients $\{\gamma_i\}_{i\in[N]}$ are distinct public parameters in $\mathbb{F}_q$ agreed between the users. Then, the second term in (43) can be written as:

$$I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^2 | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \tag{55}$$

$$= I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}}; \{\hat{\mathbf{a}}_i\}_{i\in\mathcal{H}}, \{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}} | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \tag{56}$$

$$\le H(\{\hat{\mathbf{a}}_i\}_{i\in\mathcal{H}}, \{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}) - H(\{\hat{\mathbf{a}}_i\}_{i\in\mathcal{H}}, \{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}} | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) \tag{57}$$

where (57) holds since conditioning cannot increase entropy. For the first term in (57),

$$H(\{\hat{\mathbf{a}}_i\}_{i\in\mathcal{H}}, \{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}) \le ((T+1)Nd)\log q \tag{58}$$

which follows from $|\mathcal{H}| = N - |\mathcal{T}|$ with $|\mathcal{T}| = T$. For the second term in (57), we have:

$$H(\{\hat{\mathbf{a}}_i\}_{i\in\mathcal{H}}, \{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}} | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]}, \mathbf{w}^{(J)})$$

$$= H(\{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}} | \{\mathbf{a}_i\}_{i\in\mathcal{H}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}) + H(\{\mathbf{a}_i\}_{i\in\mathcal{H}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}) \tag{59}$$

$$= H\Big(\Big\{\sum_{k\in[T]} \gamma_j^k \mathbf{b}_{ik}\Big\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}\Big) + ((N-T)d + Td + T^2d)\log q \tag{60}$$

$$= \sum_{i\in\mathcal{H}} H(\{\mathbf{s}_{ij}\}_{j\in\mathcal{T}}) + (Nd + T^2d)\log q \tag{61}$$

where (59) is from the chain rule of entropy, and that given $\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]}$, the only uncertainty in $\hat{\mathbf{a}}_i$ is due to $\mathbf{a}_i$; (60) is from the entropy of uniform random variables. In (61), we define:

$$\mathbf{s}_{ij} \triangleq \sum_{k\in[T]} \gamma_j^k \mathbf{b}_{ik} \text{ for all } j \in \mathcal{T} \tag{62}$$

from which one can write,

$$(\mathbf{s}_{i1},\ldots,\mathbf{s}_{iT}) = (\mathbf{b}_{i1},\ldots,\mathbf{b}_{iT}) \underbrace{\begin{bmatrix} \gamma_1 & \cdots & \gamma_T \\ \vdots & \ddots & \vdots \\ \gamma_1^T & \cdots & \gamma_T^T \end{bmatrix}}_{\mathbf{A}} \tag{63}$$

where $\mathbf{A}$ is a $T\times T$ MDS matrix (invertible), which represents a bijective mapping. Hence,

$$H(\{\mathbf{s}_{ij}\}_{j\in\mathcal{T}}) = H(\mathbf{s}_{i1},\ldots,\mathbf{s}_{iT}) = H(\mathbf{b}_{i1},\ldots,\mathbf{b}_{iT}) = Td\log q \tag{64}$$

By combining (64) with (61), we have:

$$H(\{\hat{\mathbf{a}}_i\}_{i\in\mathcal{H}}, \{[\mathbf{a}_i]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{a}_i\}_{i\in\mathcal{T}}, \{\mathbf{b}_{ik}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}} | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]}, \mathbf{w}^{(J)}) = ((T+1)Nd)\log q \tag{65}$$

Finally, by combining (65) and (58) with (57), we find that:

$$0 \le I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^2 | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \le ((T+1)Nd)\log q - ((T+1)Nd)\log q \le 0 \tag{66}$$

where the first inequality follows from the non-negativity of mutual information. Hence, the second term in (43) also satisfies $I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^2 | \mathcal{M}_{\mathcal{T}}^1, \{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) = 0$.

Stage 3: Model Initialization. We now consider the third term in (43), which corresponds to model initialization. We denote the secret share $[\mathbf{w}_i^{(0)}]_j$ sent from user i to user j as:

$$[\mathbf{w}_i^{(0)}]_j \triangleq \mathbf{w}_i^{(0)} + \sum_{k\in[T]} \gamma_j^k \mathbf{z}_{ik}^{(0)} \tag{67}$$

where $\{\mathbf{z}_{ik}^{(0)}\}_{k\in[T]}$ are uniformly random vectors of size d, and $\{\gamma_j\}_{j\in[N]}$ are as defined in (54). We can then rewrite the mutual information condition from the third term in (43) as

follows:

$$I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\mathcal{M}_T^3|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$=I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}},\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{68}$$

$$=I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$+I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{69}$$

$$=H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$-H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)})$$
$$+H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$-H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{70}$$

We next consider each term in (70) separately. For the first term in (70), we have that,

$$H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$\leq H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}) \tag{71}$$
$$\leq (dT+dT^2)\log q \tag{72}$$

which holds since uniform distribution maximizes entropy. For the second term in (70), we have,

$$H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)})$$
$$\geq H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathbf{w}^{(0)}) \tag{73}$$
$$=H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}) \tag{74}$$
$$=(dT+dT^2)\log q \tag{75}$$

where (73) follows from the data processing inequality, since $\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}-\mathbf{w}^{(0)}-\mathbf{w}^{(J)},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]}$ forms a Markov chain, and (74) holds since,

$$H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]})-H(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]}|\mathbf{w}^{(0)}) \tag{76}$$
$$=I(\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]};\mathbf{w}^{(0)}) \tag{77}$$
$$=H(\mathbf{w}^{(0)})-H(\mathbf{w}^{(0)}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]})$$
$$=d\log q-d\log q=0 \tag{78}$$

where (78) holds since $\mathbf{w}_i^{(0)}$ are uniformly random for all $i\in[N]$. For the third term in (70),

$$H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$\leq H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}})$$
$$\leq (N-T)T\log q \tag{79}$$

Finally, for the last term in (70), we rewrite the condition from (67) as follows:

$$([\mathbf{w}_i^{(0)}]_1,\ldots,[\mathbf{w}_i^{(0)}]_T)=\mathbf{w}_i^{(0)}\underbrace{(1,\ldots,1)}_{\mathbf{1}}+\underbrace{(\mathbf{z}_{i1}^{(0)},\ldots,\mathbf{z}_{iT}^{(0)})}_{\mathbf{z}_i^{(0)}}\mathbf{A}$$
$$=\mathbf{w}_i^{(0)}\mathbf{1}+\mathbf{z}_i^{(0)}\mathbf{A} \tag{80}$$

where $\mathbf{A}$ is an $T\times T$ MDS matrix as defined in (63). Then, the last term in (70) becomes:

$$H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)})$$
$$\geq H(\{[\mathbf{w}_i^{(0)}]_j\}_{i\in\mathcal{H},j\in\mathcal{T}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\{\mathbf{w}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{81}$$
$$=H(\{\mathbf{w}_i^{(0)}\mathbf{1}+\mathbf{z}_i^{(0)}\mathbf{A}\}_{i\in\mathcal{H}}|\{\mathbf{w}_i^{(0)},\mathbf{z}_{ik}\}_{i\in\mathcal{T},k\in[T]},\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\{\mathbf{w}_i^{(0)}\}_{i\in\mathcal{H}})$$
$$=H(\{\mathbf{z}_i^{(0)}\}_{i\in\mathcal{H}}) \tag{82}$$
$$=(N-T)Td\log q \tag{83}$$

where (82) holds since $\mathbf{A}$ is an MDS matrix, hence is invertible, and that random vectors generated by the honest users are independent from the random vectors generated by adversarial users, and (83) holds since $\mathbf{z}_i^{(0)}$ are generated uniformly random for all $i\in[N]$.

Finally, by combining (72), (75), (79), (83) with (70), we have for the third term in (43):

$$0\leq I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\mathcal{M}_T^3|\mathcal{M}_T^1,\mathcal{M}_T^2,\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{84}$$
$$\leq (dT+dT^2)\log q-(dT+dT^2)\log q+(N-T)Td\log q-(N-T)Td\log q=0 \tag{85}$$

Stage 4: Model Encoding. We next consider the fourth term in (43), which corresponds to model encoding (Stage 4) of SCALR. We represent the secret share of $\mathbf{r}_i^{(t)}$ at user $j\in[N]$ as:

$$[\mathbf{r}_i^{(t)}]_j=\mathbf{r}_i^{(t)}+\sum_{k\in[T]}\gamma_j^k\mathbf{g}_{ik}^{(t)}\ \text{for all}\ i\in[N], \tag{86}$$

where $\mathbf{g}_{ik}^{(t)}\in\mathbb{F}_q^d$ are uniformly random vectors, and γ_j for $j\in[N]$ are as given in (54). Without loss of generality, we represent the secret share of the model $\mathbf{w}^{(t)}$ at user $i\in[N]$ as:

$$[\mathbf{w}^{(t)}]_i=\mathbf{w}^{(t)}+\sum_{k\in[T]}\gamma_i^k\mathbf{z}_k^{(t)}\quad\text{for all}\ i\in[N], \tag{87}$$

where $\mathbf{z}_k^{(t)}\in\mathbb{F}_q^d$, and $\{\gamma_i\}_{i\in[N]}$ are defined in (54). Then, for the third term in (43), we observe,

$$I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\mathcal{M}_T^{4,t}|\mathcal{M}_T^1,\mathcal{M}_T^2,\mathcal{M}_T^3,\cup_{l=0}^{t-1}\mathcal{M}_T^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_T^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$=I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\{[\mathbf{r}_i^{(t)}]_j,\tilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{[\hat{\mathbf{w}}^{(t)}]_i\}_{i\in[N]}|\mathcal{M}_T^1,\mathcal{M}_T^2,\mathcal{M}_T^3,$$

$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)}) \tag{88}$$

$$= H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{[\hat{\mathbf{w}}^{(t)}]_i\}_{i\in[N]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$- H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\{[\hat{\mathbf{w}}^{(t)}]_i\}_{i\in[N]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{89}$$

$$= H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\mathbf{w}^{(t)}-\sum_{j\in[N]}\mathbf{r}_j^{(t)},$$
$$\{\mathbf{z}_k^{(t)}-\sum_{j\in[N]}\mathbf{g}_{jk}^{(t)}\}_{k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$- H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\mathbf{w}^{(t)}-\sum_{j\in[N]}\mathbf{r}_j^{(t)},$$
$$\{\mathbf{z}_k^{(t)}-\sum_{j\in[N]}\mathbf{g}_{jk}^{(t)}\}_{k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{90}$$

$$= H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\mathbf{w}^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{r}_j^{(t)},$$
$$\{\mathbf{z}_k^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{g}_{jk}^{(t)}\}_{k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})$$
$$- H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},$$
$$\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}},\mathbf{w}^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{r}_j^{(t)},$$
$$\{\mathbf{z}_k^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{g}_{jk}^{(t)}\}_{k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{91}$$

where (90) holds since, from (87) and (20), one can write,

$$[\hat{\mathbf{w}}^{(t)}]_i = [\mathbf{w}^{(t)}]_i - [\mathbf{r}^{(t)}]_i$$
$$= \Big(\mathbf{w}^{(t)}-\sum_{j\in[N]}\mathbf{r}_j^{(t)}\Big)+\sum_{k\in[T]}\gamma_i^k\Big(\mathbf{z}_k^{(t)}-\sum_{j\in[N]}\mathbf{g}_{jk}^{(t)}\Big) \tag{92}$$

which is a polynomial of degree T. Since the coefficients of any degree T polynomial can be uniquely determined from $T+1$ evaluation points, there is a bijective (one-to-one) mapping from any sequence of $T+1$ coefficients $(\mathbf{w}^{(t)}-\sum_{j\in[N]}\mathbf{r}_j^{(t)},\mathbf{z}_1^{(t)}-\sum_{j\in[N]}\mathbf{g}_{j1}^{(t)},\ldots,\mathbf{z}_T^{(t)}-\sum_{j\in[N]}\mathbf{g}_{jT}^{(t)})$ to a valid $\{[\hat{\mathbf{w}}^{(t)}]_i\}_{i\in[N]}$. We next define a few variables to simplify the analysis of (91),

$$([\mathbf{w}^{(t)}]_1,\ldots,[\mathbf{w}^{(t)}]_T) = \mathbf{w}^{(t)}\underbrace{(1,\ldots,1)}_{\mathbf{1}}+\underbrace{(\mathbf{z}_1^{(t)},\ldots,\mathbf{z}_T^{(t)})}_{\mathbf{z}^{(t)}}\mathbf{A}$$
$$= \mathbf{w}^{(t)}\mathbf{1}+\mathbf{z}^{(t)}\mathbf{A} \tag{93}$$

where $\mathbf{A}$ is an MDS matrix as defined in (63). Similarly, we let:

$$([\mathbf{r}_i^{(t)}]_1,\ldots,[\mathbf{r}_i^{(t)}]_T) = \mathbf{r}_i^{(t)}\underbrace{(1,\ldots,1)}_{\mathbf{1}}+\underbrace{(\mathbf{g}_{i1}^{(t)},\ldots,\mathbf{g}_{iT}^{(t)})}_{\mathbf{g}_i^{(t)}}\mathbf{A}$$
$$= \mathbf{r}_i^{(t)}\mathbf{1}+\mathbf{g}_i^{(t)}\mathbf{A} \tag{94}$$

Finally, by using the coefficients defined in (50), we can write:

$$(\widetilde{\mathbf{r}}_{i1}^{(t)},\ldots,\widetilde{\mathbf{r}}_{iT}^{(t)}) = \mathbf{r}_i^{(t)}\underbrace{\sum_{k=1}^{K}(\lambda_{k1},\ldots,\lambda_{kT})}_{\lambda}$$
$$+\underbrace{(\mathbf{v}_{i(K+1)}^{(t)},\ldots,\mathbf{v}_{i(K+T)}^{(t)})}_{\mathbf{v}_i^{(t)}}\mathbf{M} = \mathbf{r}_i^{(t)}\lambda+\mathbf{v}_i^{(t)}\mathbf{M} \tag{95}$$

where $\mathbf{M}$ is as defined in (51). From (93)-(95), chain rule of entropy, and the independence of the random vectors generated, the second term in (91) can be rewritten as:

$$H(\{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}},\{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}},\{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}})$$
$$+H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\mathbf{w}^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{r}_j^{(t)},$$
$$\{\mathbf{z}_k^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{g}_{jk}^{(t)}\}_{k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{96}$$

$$= dT(2T+1)\log q+H(\{[\mathbf{r}_i^{(t)}]_j,\widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}},\mathbf{w}^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{r}_j^{(t)},$$
$$\{\mathbf{z}_k^{(t)}-\sum_{j\in\mathcal{H}}\mathbf{g}_{jk}^{(t)}\}_{k\in[T]}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{97}$$

$$= dT(2T+1)\log q+H(\{\mathbf{r}_i^{(t)}\mathbf{1}+\mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}},$$
$$\{\mathbf{r}_i^{(t)}\lambda+\mathbf{v}_i^{(t)}\mathbf{M}\}_{i\in\mathcal{H}},\mathbf{w}^{(t)}-\sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)},\mathbf{z}^{(t)}-\sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}|\mathcal{M}_{\mathcal{T}}^1,$$
$$\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{98}$$

$$\geq dT(2T+1)\log q+H(\{\mathbf{r}_i^{(t)}\mathbf{1}+\mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}},$$
$$\{\mathbf{r}_i^{(t)}\lambda+\mathbf{v}_i^{(t)}\mathbf{M}\}_{i\in\mathcal{H}},\mathbf{w}^{(t)}-\sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)},\mathbf{z}^{(t)}-\sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}|$$
$$\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},$$
$$\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)},\mathbf{w}^{(t)},\mathbf{z}^{(t)}) \tag{99}$$

$$= dT(2T+1)\log q+H(\{\mathbf{r}_i^{(t)}\mathbf{1}+\mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}},$$
$$\{\mathbf{r}_i^{(t)}\lambda+\mathbf{v}_i^{(t)}\mathbf{M}\}_{i\in\mathcal{H}},\sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)},\sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}|\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,$$
$$\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\mathbf{w}^{(J)}) \tag{100}$$

$$= dT(2T+1)\log q + H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \{\mathbf{r}_i^{(t)}\lambda + \mathbf{v}_i^{(t)}\mathbf{M}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) \quad (101)$$

$$= dT(2T+1)\log q + H(\{\mathbf{r}_i^{(t)}\lambda + \mathbf{v}_i^{(t)}\mathbf{M}\}_{i\in\mathcal{H}}| \{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) + H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) \quad (102)$$

$$\geq dT(2T+1)\log q + H(\{\mathbf{r}_i^{(t)}\lambda + \mathbf{v}_i^{(t)}\mathbf{M}\}_{i\in\mathcal{H}}| \{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}, \{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{H}}) + H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) \quad (103)$$

$$= dT(2T+1)\log q + H(\{\mathbf{v}_i^{(t)}\}_{i\in\mathcal{H}}) + H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) \quad (104)$$

$$= dT(2T+1)\log q + (N-T)dT\log q + H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) \quad (105)$$

$$\geq (T+2TN+1)d\log q \quad (106)$$

where (99) and (103) holds since conditioning cannot increase entropy, (101) and (104) holds from the independence of random vectors, and that $\mathbf{M}$ is an MDS matrix (invertible); (105) follows from the entropy of uniform random variables. Finally, (106) follows from,

$$H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)})$$
$$= H(\sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}|\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}) + H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}) \quad (107)$$

$$= H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}) \quad (108)$$

$$\geq H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}|\sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{H}}) + H(\sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}) \quad (109)$$

$$= (N-T)Td\log q + d\log q \quad (110)$$

where (108) is from $\sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)} = (\sum_{i\in\mathcal{H}}(\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}) - \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}\mathbf{1})\mathbf{A}^{-1}$; (110) holds since $\mathbf{A}$ is an MDS matrix. We next analyze the first term in (91). For this term, we have that:

$$H(\{[\mathbf{r}_i^{(t)}]_j, \widetilde{\mathbf{r}}_{ij}^{(t)}\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}}, \{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}, \{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \mathbf{w}^{(t)} - \sum_{j\in\mathcal{H}}\mathbf{r}_j^{(t)}, \{\mathbf{z}_k^{(t)} - \sum_{j\in\mathcal{H}}\mathbf{g}_{jk}^{(t)}\}_{k\in[T}|$$
$$\mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \mathcal{M}_{\mathcal{T}}^3, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l}, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l}, \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)}) \quad (111)$$

$$\leq H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \{\mathbf{r}_i^{(t)}\lambda + \mathbf{v}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}}, \{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}, \{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \mathbf{w}^{(t)} - \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}, \mathbf{z}^{(t)} - \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)}) \quad (112)$$

$$= H(\{\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \{\mathbf{r}_i^{(t)}\lambda + \mathbf{v}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}}, \{\mathbf{r}_i^{(t)}\}_{i\in\mathcal{T}}, \{\mathbf{g}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}, \{\mathbf{v}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in\{K+1,\ldots,K+T\}}}, \mathbf{w}^{(t)} - \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)}) \quad (113)$$

$$\leq (T+2TN+1)d\log q \quad (114)$$

where (112) holds since conditioning cannot increase entropy, and (113) holds since:

$$\mathbf{z}^{(t)} - \sum_{i\in\mathcal{H}}\mathbf{g}_i^{(t)} = \Big((\mathbf{w}^{(t)}\mathbf{1} + \mathbf{z}^{(t)}\mathbf{A}) - (\mathbf{w}^{(t)} - \sum_{i\in\mathcal{H}}\mathbf{r}_i^{(t)})\mathbf{1} - \sum_{i\in\mathcal{H}}(\mathbf{r}_i^{(t)}\mathbf{1} + \mathbf{g}_i^{(t)}\mathbf{A})\Big)\mathbf{A}^{-1} \quad (115)$$

and (114) holds since entropy is maximized by the uniform distribution. Finally, by combining (106) and (114) with (91) and (89), we find for the fourth term in (43) that:

$$0 \leq I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^{4,t}|\mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \mathcal{M}_{\mathcal{T}}^3, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{4,l}, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l}, \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})$$
$$\leq (T+2TN+1)d\log q - (T+2TN+1)d\log q = 0 \quad (116)$$

Stage 5: Gradient Computing and Model Update. We next consider the last term in (43), which corresponds to Stage 5 of the proposed framework, i.e., local gradient computation and model update. Without loss of generality, we denote the secret share of $\mathbf{u}_i^{(t)}$ at user $j \in [N]$ as:

$$[\mathbf{u}_i^{(t)}]_j \triangleq \mathbf{u}_i^{(t)} + \sum_{k\in[T]}\gamma_j^k\mathbf{n}_{ik}^{(t)} \quad \text{for all } i \in [N], \quad (117)$$

where $\mathbf{u}_{ik}^{(t)} \in \mathbb{F}_q^d$ are uniformly random vectors for all $i \in [N], k \in [T]$, and γ_j is as defined in (54). Similar to (95), we also represent (117) in matrix notation as:

$$([\mathbf{u}_i^{(t)}]_1, \ldots, [\mathbf{u}_i^{(t)}]_T) = \mathbf{u}_i^{(t)}\underbrace{(1,\ldots,1)}_{\mathbf{1}} + \underbrace{(\mathbf{n}_{i1},\ldots,\mathbf{n}_{iT})}_{\mathbf{n}_i^{(t)}}\mathbf{A} = \mathbf{u}_i^{(t)}\mathbf{1} + \mathbf{n}_i^{(t)}\mathbf{A} \quad (118)$$

where $\mathbf{A}$ is a $T \times T$ MDS matrix as given in (63). Then, the last term in (43) can be written as:

$$I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \mathcal{M}_{\mathcal{T}}^{5,t}|\mathcal{M}_{\mathcal{T}}^1, \mathcal{M}_{\mathcal{T}}^2, \mathcal{M}_{\mathcal{T}}^3, \cup_{l=0}^{t}\mathcal{M}_{\mathcal{T}}^{4,l}, \cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l}, \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}, \mathbf{w}^{(J)})$$
$$= I(\{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{H}}; \{[\mathbf{u}_i^{(t)}]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\hat{\mathbf{u}}_i^{(t)}\}_{i\in[N]}, \{\mathbf{u}_i^{(t)}\}_{i\in\mathcal{T}}, \{\mathbf{n}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\ k\in[T]}}|\mathcal{M}, \{\mathcal{D}_i, \mathcal{Y}_i\}_{i\in\mathcal{T}}) \quad (119)$$
$$\leq H(\{[\mathbf{u}_i^{(t)}]_j\}_{\substack{i\in\mathcal{H}\\ j\in\mathcal{T}}}, \{\hat{\mathbf{u}}_i^{(t)}\}_{i\in[N]}, \{\mathbf{u}_i^{(t)}\}_{i\in\mathcal{T}},$$

$$\{\mathbf{n}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\k\in[T]}}|\mathcal{M},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}})$$
$$-H(\{[\mathbf{u}_i^{(t)}]_j\}_{\substack{i\in\mathcal{H}\\j\in\mathcal{T}}},\{\hat{\mathbf{u}}_i^{(t)}\}_{i\in[N]},\{\mathbf{u}_i^{(t)}\}_{i\in\mathcal{T}},$$
$$\{\mathbf{n}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\k\in[T]}}|\mathcal{M},\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in[N]},\{\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{w}}_i^{(t)})\}_{i\in[N]})$$
$$\le H(\{[\mathbf{u}_i^{(t)}]_j\}_{\substack{i\in\mathcal{H}\\j\in\mathcal{T}}},\{\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{w}}_i^{(t)})-\mathbf{u}_i^{(t)}\}_{i\in\mathcal{H}},\{\mathbf{u}_i^{(t)}\}_{i\in\mathcal{T}},$$
$$\{\mathbf{n}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\k\in[T]}})-H(\{\mathbf{n}_i^{(t)}\mathbf{A}\}_{i\in\mathcal{H}},\{\mathbf{u}_i^{(t)}\}_{i\in[N]},\{\mathbf{n}_{ik}^{(t)}\}_{\substack{i\in\mathcal{T}\\k\in[T]}}) \tag{120}$$
$$\le N(T+1)d\log q-N(T+1)d\log q \tag{121}$$
$$=0 \tag{122}$$

where $\mathcal{M}\triangleq\{\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^2,\mathcal{M}_{\mathcal{T}}^3,\cup_{l=0}^t\mathcal{M}_{\mathcal{T}}^{4,l},\cup_{l=0}^{t-1}\mathcal{M}_{\mathcal{T}}^{5,l},\mathbf{w}^{(J)}\}$; (121) holds since random vectors are generated independently, $\mathbf{A}$ is an MDS matrix (invertible), conditioning cannot increase entropy, and that given $\mathcal{M}_{\mathcal{T}}^1,\mathcal{M}_{\mathcal{T}}^{4,l}$, there is no uncertainty in $\{\widetilde{\mathbf{X}}_i^T\hat{g}(\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{w}}_i^{(t)})\}_{i\in\mathcal{T}}$.

Combining Stages 1-5. Finally, by combining (53), (66), (116), and (122) with (43), we have $I(\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{H}};\mathcal{M}_{\mathcal{T}}|\{\mathcal{D}_i,\mathcal{Y}_i\}_{i\in\mathcal{T}},\mathbf{w}^{(J)})=0$, which completes the proof. □

Appendix C
Correctness

The correctness of Lagrange coding and Shamir's secret sharing follows from [1] and [8]. We next show that the model update operations from (40) correctly recover the target model in (26). From (40), at the end of each round t, client i holds a secret share $[\mathbf{w}^{(t+1)}]_i$ of $\mathbf{w}^{(t+1)}$, where

$$\mathbf{w}^{(t+1)}=\bar{\eta}^{(r-1)c_t+1}\mathbf{w}^{(t)}-(f(\mathbf{X},\mathbf{w}^{(t)})-\bar{\eta}^{rc_t}\mathbf{X}^T\mathbf{y}). \tag{123}$$

Next, define $\overline{\mathbf{w}}^{(t)}$ as the target model from (26), where $\overline{\mathbf{w}}^{(0)}\triangleq\mathbf{w}^{(0)}$,

$$\overline{\mathbf{w}}^{(t+1)}\triangleq\overline{\mathbf{w}}^{(t)}-\frac{1}{\bar{\eta}}\mathbf{X}^T(\hat{g}(\mathbf{X}\times\mathbf{w}^{(t)})-\mathbf{y})$$
$$\text{for } t\in\{0,\dots,J-1\}, \tag{124}$$

and show that the model updates from (40) satisfy $\frac{\mathbf{w}^{(t)}}{\bar{\eta}^{c_t}}=\overline{\mathbf{w}}^{(t)}$ for all training rounds $t\ge 0$. The proof follows by induction, from the following two steps: *1) Base case:* For the base case $t=0$, the result follows directly from (40) and that $c_0\triangleq 0$, *2) Induction step:* Next, assume that $\frac{\mathbf{w}^{(t)}}{\bar{\eta}^{c_t}}=\overline{\mathbf{w}}^{(t)}$ holds for an arbitrary round t, and show that it also holds for round $t+1$,

$$\mathbf{w}^{(t+1)}$$
$$=\bar{\eta}^{(r-1)c_t+1}\mathbf{w}^{(t)}-(f(\mathbf{X},\mathbf{w}^{(t)})-\bar{\eta}^{rc_t}\mathbf{X}^T\mathbf{y}) \tag{125}$$
$$=\bar{\eta}^{(r-1)c_t+1}\bar{\eta}^{c_t}\overline{\mathbf{w}}^{(t)}-(f(\mathbf{X},\bar{\eta}^{c_t}\overline{\mathbf{w}}^{(t)})-\bar{\eta}^{rc_t}\mathbf{X}^T\mathbf{y}) \tag{126}$$
$$=\bar{\eta}^{rc_t+1}\overline{\mathbf{w}}^{(t)}-(\mathbf{X}^T\sum_{j=0}^{r}\bar{\eta}^{(r-j)c_t}\theta_j(\mathbf{X}\times\bar{\eta}^{c_t}\overline{\mathbf{w}}^{(t)})^j$$
$$-\bar{\eta}^{rc_t}\mathbf{X}^T\mathbf{y}) \tag{127}$$
$$=\bar{\eta}^{rc_t+1}\overline{\mathbf{w}}^{(t)}-\bar{\eta}^{rc_t}(\mathbf{X}^T\sum_{j=0}^{r}\theta_j(\mathbf{X}\times\overline{\mathbf{w}}^{(t)})^j-\mathbf{X}^T\mathbf{y}) \tag{128}$$
$$=\bar{\eta}^{rc_t+1}\Big(\overline{\mathbf{w}}^{(t)}-\frac{1}{\bar{\eta}}(\mathbf{X}^T\hat{g}(\mathbf{X}\times\overline{\mathbf{w}}^{(t)})-\mathbf{X}^T\mathbf{y})\Big)$$
$$=\bar{\eta}^{c_{t+1}}\overline{\mathbf{w}}^{(t+1)} \tag{129}$$

where (126) holds since $\mathbf{w}^{(t)}=\bar{\eta}^{c_t}\overline{\mathbf{w}}^{(t)}$ for round t holds by assumption, (129) follows from (124) and that $c_{t+1}=rc_t+1$, which completes the proof.

References

[1] Q. Yu et al., "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2019, pp. 1–11.

[2] J. So, B. Güler, and A. S. Avestimehr, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 441–451, Mar. 2021.

[3] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.

[4] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 479–489, Mar. 2021.

[5] J. So, B. Güler, and A. S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2020, pp. 1–13.

[6] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, Nov. 1982, pp. 160–164.

[7] M. Ben-Or and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 1–10.

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

[9] I. Damgård and J. B. Nielsen, "Scalable and unconditionally secure multiparty computation," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2007, pp. 572–590.

[10] Z. Beerliová-Trubìniová and M. Hirt, "Perfectly-secure MPC with linear communication complexity," in *Proc. Theory Cryptography Conf.* Cham, Switzerland: Springer, 2008, pp. 213–230.

[11] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 334–348.

[12] A. Gascón et al., "Privacy-preserving distributed linear regression on high-dimensional data," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 345–364, Oct. 2017.

[13] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[14] P. Mohassel and P. Rindal, "ABY 3: A mixed protocol framework for machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 35–52.

[15] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-party secure computation for neural network training," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, Jul. 2019.

[16] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1–14.

[17] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1–10.

[18] Y. Zhao and H. Sun, "Information theoretic secure aggregation with user dropouts," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1124–1129.

[19] J. So et al., "LightSecAgg: A lightweight and versatile design for secure aggregation in federated learning," in *Proc. Mach. Learn. Syst.*, vol. 4, 2022, pp. 1–27.

[20] J. So, R. E. Ali, B. Guler, J. Jiao, and S. Avestimehr, "Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 1–9.

[21] A. R. Elkordy, J. Zhang, Y. H. Ezzeldin, K. Psounis, and S. Avestimehr, "How much privacy does federated learning with secure aggregation guarantee?" in *Proc. Priv. Enhancing Technol. (PETS)*, 2023, pp. 1–16.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptography Conf.* Cham, Switzerland: Springer, 2006, pp. 265–284.
[23] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Proc. Adv. Neural Inf. Proc. Sys.*, 2009, pp. 1–8.
[24] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.
[25] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comp. Commun. Secur.*, 2016, pp. 308–318.
[26] M. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1876–1884.
[27] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
[28] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2012, pp. 933–941.
[29] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Distributed learning without distress: Privacy-preserving empirical risk minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6346–6357.
[30] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*, vol. 20, no. 9. Stanford, CA, USA: Stanford University, 2009.
[31] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, May 2009, pp. 169–178.
[32] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[33] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Cham, Switzerland: Springer, 2012, pp. 1–21.
[34] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014.
[35] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Comput.*, vol. 21, pp. 277–286, Apr. 2017.
[36] Q. Wang et al., "Privacy-preserving collaborative model learning: The case of word vector training," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2381–2393, Dec. 2018.
[37] K. Han, S. Hong, J. H. Cheon, and D. Park, "Logistic regression on homomorphic encrypted data at scale," in *Proc. Annu. Conf. Innovative App. Artif. Intell. (IAAI)*, 2019, pp. 1–6.
[38] J. Brinkhuis and V. Tikhomirov, *Optimization: Insights and Applications*. Princeton, NJ, USA: Princeton Univ. Press, 2005.
[39] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2010, pp. 35–50.
[40] K. S. Kedlaya and C. Umans, "Fast polynomial factorization and modular composition," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1767–1802, Jan. 2011.
[41] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, 2009.
[42] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, vol. 17, 2004, pp. 1–8.
[43] L. Dalcín, R. Paz, and M. Storti, "MPI for Python," *J. Parallel Distrib. Comput.*, vol. 65, no. 9, pp. 1108–1115, Sep. 2005.
[44] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1–12.
[45] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated Learning*. Cham, Switzerland: Springer, 2020, pp. 17–31.
[46] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients—How easy is it to break privacy in federated learning?" in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1–11.
[47] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete Gaussian mechanism for federated learning with secure aggregation," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1–54.
[48] W. Chen, C. A. Choquette-Choo, P. Kairouz, and A. T. Suresh, "The fundamental price of secure aggregation in differentially private federated learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 1–34.
[49] M. C. Thomas and A. T. Joy, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley-Interscience, 2006.

Xingyu Lu received the B.E. degree from the Department of Computer Science and Information Technology, Zhejiang Gongshang University, China, in 2019, and the M.Sc. degree in robotics (computer science) from the Khoury College of Computer Science and the College of Engineering, Northeastern University, Boston, MA, USA. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California, Riverside. His research interests include private machine learning, distributed learning, and federated learning.

Hasin Us Sami (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California, Riverside. His research interests include federated and distributed machine learning, information theory, secure and private computing, and wireless networks.

Başak Güler (Member, IEEE) received the B.Sc. degree in electrical and electronics engineering from Middle East Technical University (METU), Ankara, Turkey, and the Ph.D. degree from the Wireless Communications and Networking Laboratory, The Pennsylvania State University, in 2017. From 2018 to 2020, she was a Post-Doctoral Scholar with the University of Southern California. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of California, Riverside. Her research interests include information theory, distributed computing, machine learning, and wireless networks. She is a recipient of the 2022 NSF CAREER Award.