CodedPrivateML: A Fast and Privacy-Preserving Framework for Distributed Machine Learning

Jinhyun So, Başak Güler, *Member, IEEE*, and A. Salman Avestimehr, *Fellow, IEEE*

***Abstract*—How to train a machine learning model while keeping the data private and secure? We present CodedPrivateML, a fast and scalable approach to this critical problem. CodedPrivateML keeps both the data and the model information-theoretically private, while allowing efficient parallelization of training across distributed workers. We characterize CodedPrivateML's privacy threshold and prove its convergence for logistic (and linear) regression. Furthermore, via extensive experiments on Amazon EC2, we demonstrate that CodedPrivateML provides significant speedup over cryptographic approaches based on multi-party computing (MPC).**

***Index Terms*—Distributed training, privacy-preserving machine learning.**

I. Introduction

MODERN machine learning models are breaking new ground by achieving unprecedented performance in various application domains [1]. Training such models, however, is a challenging task. Due to the typically large volume of data and complexity of models, training is a compute and storage intensive task. Furthermore, training should often be done on sensitive data, such as healthcare records, browsing history, or financial transactions, which raises the issues of security and privacy of the dataset. This creates a challenging dilemma. On the one hand, due to its complexity, training is often desired to be outsourced to more capable computing platforms, such as the cloud. On the other hand, the training dataset is often sensitive and particular care should be taken to protect its privacy against potential breaches in such platforms. This dilemma gives rise to the main problem that we study here: How can we offload the training task to a distributed computing platform, while maintaining the privacy of the dataset?

Manuscript received August 15, 2020; revised December 3, 2020; accepted January 14, 2021. Date of publication January 21, 2021; date of current version March 16, 2021. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract HR001117C0053; in part by the Army Research Office (ARO) under Award W911NF1810400; in part by the National Science Foundation (NSF) under Grant CCF-1703575 and Grant CCF-1763673; in part by the Office of Naval Research (ONR) under Award N00014-16-1-2189; and in part by the Intel. *(Jinhyun So and Başak Güler contributed equally to this work.) (Corresponding author: Jinhyun So.)*

Jinhyun So and A. Salman Avestimehr are with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: jinhyuns@usc.edu; avestimehr@ee.usc.edu).

Başak Güler is with the Department of Electrical and Computer Engineering, University of California Riverside, Riverside, CA 92521 USA (e-mail: bguler@ece.ucr.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/JSAIT.2021.3053220, provided by the authors.

Digital Object Identifier 10.1109/JSAIT.2021.3053220

Cloud environments often operate on a shared physical infrastructure, where multiple users share the same host machine, and are separated from each other by virtual machines that act as barriers to prevent information leakage. This shared environment provides significant benefits for scaling up cloud systems, but also introduces important security and privacy challenges that may result from potentially adversarial users. For instance, it has been shown that adversarial users can compromise the host machines by disguising themselves as regular users, and access the information of other users sharing the same host machines [2]–[7]. The focus of this article is on privacy protection against such adversaries that can access a portion of the physical host machines in the cloud, and use them to spy on other users' datasets. We focus on the semi-honest adversary setup, where the adversaries follow the protocol but may leak information in an attempt to learn the training dataset. Our goal is to develop a privacy-preserving training strategy for the honest users that will protect the privacy of their datasets even if a portion of the compute machines in the cloud are controlled by adversaries.

More specifically, we consider a scenario in which a data-owner (e.g., a hospital) wishes to train a logistic regression model by offloading the large volume of data (e.g., healthcare records) and computationally-intensive training tasks (e.g., gradient computations) to N machines over a cloud platform, while ensuring that any collusions between T out of N workers do not leak information about the training dataset. We propose a new framework, CodedPrivateML (Coded Privacy-preserving Machine Learning), towards addressing this problem. CodedPrivateML has three salient features:

1) provides strong information-theoretic privacy guarantees for both the training dataset and model parameters in the presence of colluding workers,
2) enables fast training by distributing the computation load effectively across several workers,
3) leverages a new method for encoding the dataset and model parameters based on coding and information theory principles, which significantly reduces the communication overhead and the complexity for distributed training.

At a high level, CodedPrivateML can be described as follows. It secret shares the dataset and model parameters at each round of the training in two steps. First, it employs stochastic quantization to convert the dataset and the weight vector at each round into a finite domain. It then combines (or *encodes*) the quantized values with random matrices using Lagrange

coding [8], to guarantee privacy (in an information-theoretic sense) while simultaneously distributing the workload among multiple workers. The challenge is however that Lagrange coding can only work for computations that are in the form of polynomial evaluations. The gradient computation for logistic regression, on the other hand, includes non-linearities that cannot be expressed as polynomials. CodedPrivateML handles this challenge through polynomial approximations of the non-linear sigmoid function in the training phase. Upon secret sharing of the encoded dataset and model parameters, each worker performs the gradient computations using the chosen polynomial approximation, and sends the result back to the master. The workers perform the computations over the quantized and encoded data *as if they were computing over the uncoded dataset*. That is, the structure of the computations are the same for computing over the uncoded dataset versus computing over the encoded dataset. Finally, the master collects the results from a subset of fastest workers and decodes the gradient over the finite field. It then converts the decoded gradients to the real domain, updates the weight vector, and secret shares it with the worker nodes for the next round. We note that since the computations are performed in a finite domain while the weights are updated in the real domain, the update process may lead to undesired behavior as weights may not converge. Our system guarantees convergence through a stochastic quantization technique while converting between real and finite fields.

We theoretically prove that CodedPrivateML guarantees the convergence of the model parameters, while providing information-theoretic privacy for the training dataset. Our theoretical analysis also identifies a trade-off between privacy and parallelization. More specifically, each additional worker can be utilized either for more privacy, by protecting against a larger number of collusions T, or more parallelization, by reducing the computation load at each worker. We characterize this trade-off for CodedPrivateML. Furthermore, we empirically demonstrate the impact of CodedPrivateML by comparing it with the cryptographic approach based on secure multi-party computing (MPC) [9]–[12], that can also be applied to enable privacy-preserving machine learning tasks (e.g., see [13]–[18]). In particular, we envision a master who secret shares its data and model among multiple workers who collectively perform the gradient computation using a multi-round MPC protocol. Given our focus on information-theoretic privacy, the most relevant MPC-based schemes for empirical comparison are the protocols from [10] and [11], [12] based on Shamir's secret sharing [19]. While several more recent works design MPC-based learning setups with information-theoretic privacy, their constructions are limited to three or four parties [20], [21].

We run extensive experiments over the Amazon EC2 cloud platform to empirically demonstrate the performance of CodedPrivateML. We train a logistic regression model for image classification over the CIFAR-10 [22] and GISETTE [23] datasets, while the computation workload is distributed to up to $N = 50$ machines over the cloud. We demonstrate that CodedPrivateML can provide significant speedup in the training time against the state-of-the-art MPC baseline (up to 5.2×), while guaranteeing comparable levels of accuracy. This is primarily due to conventional MPC protocols' reliance on extensive communication and coordination between the workers for private computing, and not benefiting from parallelization. They can however guarantee a higher privacy threshold (i.e., larger T) compared with CodedPrivateML.

A. Other Related Works

Apart from the MPC-based schemes, one can consider two other solutions to this problem. One is based on Homomorphic Encryption (HE) [24] which allows for computations to be performed on encrypted data, and has been used for privacy-preserving machine learning solutions [25]–[33]. The privacy guarantees of HE are based on computational assumptions, whereas CodedPrivateML provides strong information-theoretic privacy. Moreover, HE requires computations to be performed on encrypted data which leads to many orders of magnitude slow down in training. For example, for image classification on the simple MNIST dataset, HE takes 2 hours to learn a logistic regression model with 96% accuracy [33], whereas for the same training setup, CodedPrivateML takes only 37 seconds. This is due to the fact that, in CodedPrivateML there is no slow down in performing coded computations which allows for a faster implementation. As a trade-off, HE allows collusions between a larger number of workers whereas in CodedPrivateML this number is determined by other system parameters such as the number of workers and the computation load assigned per worker.

Another possible solution is based on differential privacy (DP), which is a noisy release mechanism that preserves the privacy of personally identifiable information, in that the removal of any single element from the dataset does not change the computation outcomes significantly [34]. In the context of machine learning, DP is mainly used for training when the model parameters are to be released for public use, to ensure that the individual data points from the dataset cannot be identified from the released model [35]–[41]. The main difference between these approaches and our work is that our focus is on ensuring strong information-theoretic privacy (that leaks no information about the dataset) during training, while preserving the accuracy of the model. We note, however, that if the intention is to publicly release the model after training, it is in principle possible to compose the techniques of CodedPrivateML with differential privacy to obtain the best of both worlds.

II. System Model

We study the problem of training a logistic regression model. The training dataset is represented by a matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ consisting of m data points with d features and a label vector $\mathbf{y} \in \{0, 1\}^m$. The model parameters (weights) $\mathbf{w} \in \mathbb{R}^d$ are obtained by minimizing the cross entropy function,

$$C(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \big(-y_i \log \hat{y}_i - (1 - y_i) \log\big(1 - \hat{y}_i\big)\big), \qquad (1)$$

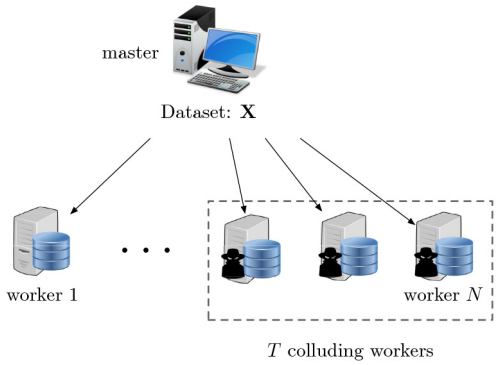


Fig. 1. The distributed training setup consisting of a master and N worker nodes.

where $\hat{y}_i = s(\mathbf{x}_i \cdot \mathbf{w}) \in (0,1)$ is the estimated probability of label i being equal to 1, $\mathbf{x}_i$ is the i^{th} row of $\mathbf{X}$, and $s(\cdot)$ is the sigmoid function $s(z) = 1/(1+e^{-z})$. The problem in (1) can be solved via gradient descent, through an iterative process that updates the model parameters in the opposite direction of the gradient. The gradient for (1) is given by $\nabla C(\mathbf{w}) = \frac{1}{m}\mathbf{X}^\top(s(\mathbf{X}\times\mathbf{w}) - \mathbf{y})$. Accordingly, model parameters are updated as,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta}{m}\mathbf{X}^\top(s(\mathbf{X}\times\mathbf{w}^{(t)}) - \mathbf{y}), \tag{2}$$

where $\mathbf{w}^{(t)}$ holds the estimated parameters from iteration t, η is the learning rate, and function $s(\cdot)$ operates element-wise over the vector given by $\mathbf{X}\times\mathbf{w}^{(t)}$.

We consider the master-worker distributed computing architecture shown in Figure 1, in which the master offloads the computationally-intensive operations to N workers. For the training problem, these operations correspond to gradient computations in (2).

In doing so, the master wishes to protect the privacy of the dataset $\mathbf{X}$ against any potential collusions between up to T workers, where T is the *privacy parameter* of the system.

In this work, we consider strong information-theoretic privacy, where any subset of T colluding workers can not learn any information about the original dataset $\mathbf{X}$. Formally, for every subset of workers $\mathcal{T} \subseteq [N]$ of size at most T, we require $I(\mathbf{X}; \mathbf{Z}_\mathcal{T}) = 0$ for any distribution on $\mathbf{X}$, where I is the mutual information, and $\mathbf{Z}_\mathcal{T}$ represents the collection of all the information received by the workers in set $\mathcal{T}$ during training. The distribution of $\mathbf{X}$ may be known to the workers. We refer to a protocol that guarantees privacy against T colluding workers as a T-private protocol. In the sequel, we present a novel protocol, CodedPrivateML, to solve (1) while preserving the information-theoretic privacy of the dataset against up to T colluding workers.

Remark 1: Although our presentation is based on logistic regression, CodedPrivateML can also be applied to the simpler linear regression model with minor modifications.

III. THECODEDPRIVATEML PROTOCOL

CodedPrivateML consists of four main components: 1) quantization, 2) encoding, 3) polynomial approximation and gradient computation, and 4) decoding the gradient and model update. Figure 2 shows the flowchart of CodedPrivateML. In the first component, the master quantizes the dataset from the real domain to the domain of integers, and then embeds it in a finite field. In the second component, the master encodes the quantized dataset and sends them to the workers. At each iteration, the master also quantizes and encodes the model parameters. In the third component, given the encoded dataset and model parameters, each worker performs the gradient computations by using polynomial approximation to substitute the sigmoid function. In the last component, the master decodes the gradient computations and converts them from the finite field to real domain, and updates the model parameters in the real domain. This process is iterated until the model parameters converge.

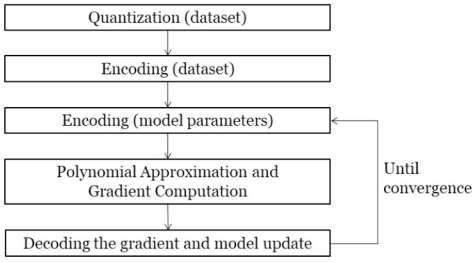


Fig. 2. Flowchart of CodedPrivateML.

We now provide the details of each component.

A. Quantization

In order to guarantee information-theoretic privacy, one has to mask the dataset and weights in a finite field[1] $\mathbb{F}$ using uniformly random matrices, so that the added randomness can make each data point appear equally likely. In contrast, the dataset and weights for the training task are defined in the domain of real numbers. Our solution to handle the conversion between the real and finite domains is through the use of stochastic quantization. Accordingly, in the first component of our system, master quantizes the dataset and weights from the real domain to the domain of integers, and then embeds them in a field $\mathbb{F}_p$ of integers modulo a prime p. The quantized version of the dataset $\mathbf{X}$ is given by $\overline{\mathbf{X}}$. The quantization of the weight vector $\mathbf{w}^{(t)}$, on the other hand, is represented by a matrix $\overline{\mathbf{W}}^{(t)}$, where each column holds an independent stochastic quantization of $\mathbf{w}^{(t)}$. This structure will be important for the convergence of the model.

We consider an element-wise lossy quantization scheme for the dataset and weights. For quantizing the dataset $\mathbf{X} \in \mathbb{R}^{m\times d}$, we use a simple deterministic rounding technique:

$$Round(x) = \begin{cases} \lfloor x \rfloor & \text{if} \quad x - \lfloor x \rfloor < 0.5 \\ \lfloor x \rfloor + 1 & \text{otherwise} \end{cases}, \tag{3}$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x. We define the quantized dataset as

$$\overline{\mathbf{X}} \triangleq \phi\Big(Round(2^{l_x}\cdot\mathbf{X})\Big), \tag{4}$$

[1]We need a finite field instead of a ring as our encoding and decoding schemes based on Lagrange coding, which we explain in Sections III-B and III-D, require division (or inverse multiplication) which the ring does not have in general.

where the rounding function from (3) is applied element-wise to the elements of matrix $\mathbf{X}$ and l_x is an integer parameter that controls the quantization loss. Function $\phi : \mathbb{R} \to \mathbb{F}_p$ is a mapping defined to represent a negative integer in the finite field by using two's complement representation,

$$\phi(x) = \begin{cases} x & \text{if } x \geq 0 \\ p + x & \text{if } x < 0 \end{cases}. \tag{5}$$

Note that the domain of (4) is $\left[-\frac{p-1}{2^{(l_x+1)}}, \frac{p-1}{2^{(l_x+1)}}\right]$. To avoid a wrap-around which may lead to an overflow error, prime p should be large enough, i.e., $p \geq 2^{l_x+1} \max\{|\mathbf{X}_{i,j}|\} + 1$. The value of p also depends on the bitwidth of the machine as well as the number of features d. For instance, in our experiments presented in Section V, we select $p = 2^{25} - 37$ in a 64-bit implementation with the GISETTE dataset whose number of features is $d = 5000$. This is the largest prime to avoid an overflow on intermediate multiplications. More specifically, we do a modular operation after the inner product of vectors instead of doing a modular operation per product of each element in order to speed up the running time of matrix-matrix multiplication. To avoid an overflow on this, p should satisfy $d(p-1)^2 \leq 2^{64} - 1$.

At each iteration t, master also quantizes the weight vector $\mathbf{w}^{(t)}$ from real domain to the finite field. This proves to be a challenging task as it should be performed in a way to ensure the convergence of the model. Our solution to this is a quantization technique inspired by [42], [43]. Initially, we define a stochastic quantization function:

$$Q(x; l_w) \triangleq \phi\Big(Round_{stoc}\Big(2^{l_w} \cdot x\Big)\Big), \tag{6}$$

where l_w is an integer parameter to control the quantization loss. $Round_{stoc} : \mathbb{R} \to \mathbb{R}$ is a stochastic rounding function:

$$Round_{stoc}(x) = \begin{cases} \lfloor x \rfloor & \text{with prob. } 1 - (x - \lfloor x \rfloor) \\ \lfloor x \rfloor + 1 & \text{with prob. } x - \lfloor x \rfloor \end{cases}.$$

The probability of rounding x to $\lfloor x \rfloor$ is proportional to the proximity of x to $\lfloor x \rfloor$ so that stochastic rounding is unbiased (i.e., $\mathbb{E}[Round_{stoc}(x)] = x$).

For quantizing the weight vector $\mathbf{w}^{(t)}$, the master creates r independent quantized vectors:

$$\overline{\mathbf{w}}^{(t),j} \triangleq Q_j\Big(\mathbf{w}^{(t)}; l_w\Big) \in \mathbb{F}_p^{d \times 1} \quad \text{for} \quad j \in [r], \tag{7}$$

where the quantization function (6) is applied element-wise to the vector $\mathbf{w}^{(t)}$ and each $Q_j(\cdot;\cdot)$ denotes an independent realization of (6). To avoid a wrap-around which may lead to an overflow error, prime p should be large enough, i.e., $p \geq 2^{l_x+1} \max\{|\mathbf{w}_i^{(t)}|\} + 1$. The number of quantized vectors r is equal to the degree of the polynomial approximation for the sigmoid function, which we will describe later in Section III-C. The intuition behind creating r independent quantizations is to ensure that the gradient computations performed using the quantized weights are unbiased estimators of the true gradients. As detailed in Section IV, this property is fundamental for the convergence analysis of our model. The specific values of parameters l_x and l_w provide a trade-off between the rounding error and overflow error. In particular, a larger value reduces the rounding error while increasing the chance of an overflow. We denote the quantization of the weight vector $\mathbf{w}^{(t)}$ as

$$\overline{\mathbf{W}}^{(t)} = \Big[\overline{\mathbf{w}}^{(t),1} \quad \cdots \quad \overline{\mathbf{w}}^{(t),r}\Big], \tag{8}$$

by arranging the quantized vectors from (7) in matrix form.

B. Encoding the Dataset and the Model

In the second component, the master partitions the quantized dataset $\overline{\mathbf{X}} \in \mathbb{F}_p^{m \times d}$ into K submatrices and encodes them using Lagrange coding [8]. It then sends to worker $i \in [N]$ a coded submatrix $\widetilde{\mathbf{X}}_i \in \mathbb{F}_p^{\frac{m}{K} \times d}$. This encoding enables two salient features of CodedPrivateML, parallelization and information-theoretic privacy guarantees. First, parameter K is related to the computation load at each worker (i.e., what fraction of the dataset is processed at each worker) because the size of encoded dataset is $1/K$-th of the size of original dataset $\mathbf{X}$. As we will show later, we can increase the parameter K as N increases, which reduces the computation overhead of each worker and communication overhead between the master and workers. This property enables our approach to scale to a significantly larger number of workers than state-of-the-art privacy preserving machine learning approaches. Second, this encoding ensures that the coded matrices do not leak any information about the original dataset $\mathbf{X}$ even if T workers collude, which will be showed in Section IV. In addition, the master has to ensure the weight estimations sent to the workers at each iteration do not leak information about the dataset. This is because the weights updated via (2) carry information about the whole training set, and sending them directly to the workers may breach privacy. In order to prevent this, at iteration t, the master also quantizes the current weight vector $\mathbf{w}^{(t)}$ to the finite field and encodes it again using Lagrange coding.

We now state the details of our second component. The master first partitions the quantized dataset $\overline{\mathbf{X}}$ into K submatrices $\overline{\mathbf{X}} = [\overline{\mathbf{X}}_1^\top \ldots \overline{\mathbf{X}}_K^\top]^\top$, where $\overline{\mathbf{X}}_i \in \mathbb{F}_p^{\frac{m}{K} \times d}$ for $i \in [K]$. We assume that m is divisible by K. Next, the master selects $K + T$ distinct elements $\beta_1, \ldots, \beta_{K+T}$ from $\mathbb{F}_p$ and employs Lagrange coding [8] to encode the dataset. To do so, the master forms a polynomial $u : \mathbb{F}_p \to \mathbb{F}_p^{\frac{m}{K} \times d}$ of degree at most $K + T - 1$ such that $u(\beta_i) = \overline{\mathbf{X}}_i$ for $i \in [K]$, and $u(\beta_i) = \mathbf{R}_i$ for $i \in \{K+1, \ldots, K+T\}$, where $\mathbf{R}_i$'s are chosen uniformly at random from $\mathbb{F}_p^{\frac{m}{K} \times d}$ (the role of $\mathbf{R}_i$'s is to mask the dataset and provide privacy against up to T colluding workers). This can be accomplished by letting u be the respective *Lagrange interpolation polynomial,*

$$u(z) \triangleq \sum_{j \in [K]} \overline{\mathbf{X}}_j \cdot \prod_{k \in [K+T] \setminus \{j\}} \frac{z - \beta_k}{\beta_j - \beta_k} + \sum_{j=K+1}^{K+T} \mathbf{R}_j \cdot \prod_{k \in [K+T] \setminus \{j\}} \frac{z - \beta_k}{\beta_j - \beta_k}. \tag{9}$$

The master then selects N distinct elements $\{\alpha_i\}_{i \in [N]}$ from $\mathbb{F}_p$ such that $\{\alpha_i\}_{i \in [N]} \cap \{\beta_j\}_{j \in [K]} = \varnothing$, and encodes the dataset by letting $\widetilde{\mathbf{X}}_i = u(\alpha_i)$ for $i \in [N]$. By defining an encoding matrix $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_N] \in \mathbb{F}_p^{(K+T) \times N}$ whose $(i,j)^{th}$ element is given by $u_{ij} = \prod_{\ell \in [K+T] \setminus \{i\}} \frac{\alpha_j - \beta_\ell}{\beta_i - \beta_\ell}$, one can also represent the

encoding of the dataset as

$$\widetilde{\mathbf{X}}_i = u(\alpha_i) = \left(\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_K, \mathbf{R}_{K+1}, \ldots, \mathbf{R}_{K+T}\right) \cdot \mathbf{u}_i. \quad (10)$$

At iteration t, the quantized weights $\overline{\mathbf{W}}^{(t)}$ are also encoded using a Lagrange interpolation polynomial,

$$v(z) \triangleq \sum_{j\in[K]} \overline{\mathbf{W}}^{(t)} \cdot \prod_{k\in[K+T]\setminus\{j\}} \frac{z-\beta_k}{\beta_j-\beta_k} + \sum_{j=K+1}^{K+T} \mathbf{V}_j \cdot \prod_{k\in[K+T]\setminus\{j\}} \frac{z-\beta_k}{\beta_j-\beta_k}, \quad (11)$$

where $\mathbf{V}_j$ for $j \in [K+1, K+T]$ are chosen uniformly at random from $\mathbb{F}_p^{d\times r}$. The coefficients $\beta_1, \ldots, \beta_{K+T}$ are the same as in (9), and we have the property $v(\beta_i) = \overline{\mathbf{W}}^{(t)}$ for $i \in [K]$. The master then encodes the quantized weight vector by using the same evaluation points $\{\alpha_i\}_{i\in[N]}$. Accordingly, the weight vector is encoded as

$$\widetilde{\mathbf{W}}_i^{(t)} = v(\alpha_i) = \left(\overline{\mathbf{W}}^{(t)}, \ldots, \overline{\mathbf{W}}^{(t)}, \mathbf{V}_{K+1}, \ldots, \mathbf{V}_{K+T}\right) \cdot \mathbf{u}_i, \quad (12)$$

for $i \in [N]$, using the encoding matrix $\mathbf{U}$ from (10). The degree of the polynomials $u(z)$ and $v(z)$ are both $K+T-1$.

C. Polynomial Approximation and Gradient Computation

Upon receiving the encoded (and quantized) dataset and weights, workers should proceed with gradient computations. However, a major challenge is that Lagrange coding is originally designed for polynomial computations, while the gradient computations are not polynomials due to the sigmoid function. Our solution is to use a polynomial approximation of the sigmoid function,

$$\hat{s}(z) = \sum_{i=0}^{r} c_i z^i, \quad (13)$$

where r and c_i denotes the degree and coefficients of the polynomial, respectively. The coefficients are obtained by fitting the sigmoid function via least squares estimation. Using this polynomial approximation we can rewrite (2) as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta}{m}\overline{\mathbf{X}}^\top \left(\hat{s}\left(\overline{\mathbf{X}} \times \mathbf{w}^{(t)}\right) - \mathbf{y}\right). \quad (14)$$

where $\overline{\mathbf{X}}$ is the quantized version of $\mathbf{X}$, and $\hat{s}(\cdot)$ operates element-wise over the vector $\overline{\mathbf{X}} \times \mathbf{w}^{(t)}$.

Another challenge is to ensure the convergence of weights. As we detail in Section IV, this necessitates the gradient estimations to be unbiased using the polynomial approximation with quantized weights. We solve this by utilizing the computation technique from [43, Sec. 4.1] using the quantized weights formed in Section III-A. Specifically, given a degree r polynomial from (13) and r independent quantizations from (8), we define a function

$$\bar{s}\left(\overline{\mathbf{X}}, \overline{\mathbf{W}}^{(t)}\right) \triangleq \sum_{i=0}^{r} c_i \prod_{j\le i} \left(\overline{\mathbf{X}} \times \overline{\mathbf{w}}^{(t),j}\right), \quad (15)$$

where the product $\prod_{j\le i}$ operates element-wise over the vectors $(\overline{\mathbf{X}} \times \overline{\mathbf{w}}^{(t),j})$ for $j \le i$. Lastly, we note that (15) is an unbiased estimator of $\hat{s}(\overline{\mathbf{X}} \times \mathbf{w}^{(t)})$,

$$E\left[\bar{s}\left(\overline{\mathbf{X}}, \overline{\mathbf{W}}^{(t)}\right)\right] = \hat{s}\left(\overline{\mathbf{X}} \times \mathbf{w}^{(t)}\right), \quad (16)$$

where $\hat{s}(\cdot)$ acts element-wise over the vector $\overline{\mathbf{X}} \times \mathbf{w}^{(t)}$, and the result follows from the independence of quantizations. Using (15), we rewrite the update equations from (14) using quantized weights,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta}{m}\overline{\mathbf{X}}^\top \left(\bar{s}\left(\overline{\mathbf{X}}, \overline{\mathbf{W}}^{(t)}\right) - \mathbf{y}\right). \quad (17)$$

CodedPrivateML guarantees the convergence to the optimal loss function $C(\mathbf{w}^*)$ where C is the cross entropy function defined in (1), even though we use the polynomial approximation to substitute the sigmoid function in the update equation (2), which will be demonstrated in Section IV.

Computations are then performed at each worker locally. At each iteration, worker $i \in [N]$ locally computes $f : \mathbb{F}_p^{\frac{m}{K}\times d} \times \mathbb{F}_p^{d\times r} \to \mathbb{F}_p^d$,

$$f\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)}\right) = \widetilde{\mathbf{X}}_i^\top \bar{s}\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)}\right), \quad (18)$$

using $\widetilde{\mathbf{X}}_i$ and $\widetilde{\mathbf{W}}_i^{(t)}$ and sends the result back to the master. This computation is a polynomial function evaluation in finite field arithmetic and the degree of f is $\deg(f) = 2r+1$.

D. Decoding the Gradient and Model Update

After receiving the evaluation results in (18) from a sufficient number of workers, master decodes $\{f(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)})\}_{k\in[K]}$ over the finite field. The minimum number of workers needed for the decoding operation to be successful, which we call the *recovery threshold* of the protocol, is equal to $(2r+1)(K+T-1)+1$ as we demonstrate in Section IV.

We now proceed to the details of decoding. By construction of the Lagrange polynomials in (9) and (11), one can define a univariate polynomial $h(z) = f(u(z), v(z))$ such that

$$h(\beta_i) = f(u(\beta_i), v(\beta_i)) = f\left(\overline{\mathbf{X}}_i, \overline{\mathbf{W}}^{(t)}\right) = \overline{\mathbf{X}}_i^\top \bar{s}\left(\overline{\mathbf{X}}_i, \overline{\mathbf{W}}^{(t)}\right), \quad (19)$$

for $i \in [K]$. On the other hand, from (18), the computation result from worker i equals to

$$h(\alpha_i) = f(u(\alpha_i), v(\alpha_i)) = f\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)}\right) = \widetilde{\mathbf{X}}_i^\top \bar{s}\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)}\right). \quad (20)$$

The main intuition behind the decoding process is to use the computations from (20) as evaluation points $h(\alpha_i)$ to interpolate the polynomial $h(z)$. Specifically, the master can obtain all coefficients of $h(z)$ from $(2r+1)(K+T-1)+1$ evaluation results as the degree of the polynomial $h(z)$ is less than or equal to $(2r+1)(K+T-1)$. After $h(z)$ is recovered, the master can recover (19) by computing $h(\beta_i)$ for $i \in [K]$. To do so, the master performs polynomial interpolation in a finite field. Upon receiving the local computation $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)})$ in (20) from at least $(2r+1)(K+T-1)+1$ workers, the master computes

$$f\left(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)}\right) \triangleq \sum_{i\in\mathcal{I}} f\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)}\right) \cdot \prod_{j\in\mathcal{I}\setminus\{i\}} \frac{\beta_k - \alpha_j}{\alpha_i - \alpha_j} \quad (21)$$

for $k \in [K]$, where $\mathcal{I} \subseteq [N]$ denotes the set of the first $(2r+1)(K+T-1)+1$ workers who send their local computations $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)})$ to the master. The master then aggregates

Algorithm 1 CodedPrivateML

input Dataset $\mathbf{X}, \mathbf{y}$
output Model parameters $\mathbf{w}^{(J)}$ where J is the number of iterations
Master:
Compute the quantized dataset $\overline{\mathbf{X}}$ using (4).
Form the encoded matrices $\{\widetilde{\mathbf{X}}_i\}_{i\in[N]}$ in (10).
Send $\widetilde{\mathbf{X}}_i$ to worker $i \in [N]$.
Initialize the weights $\mathbf{w}^{(0)} \in \mathbb{R}^{d\times 1}$.
for iteration $t = 0, \ldots, J-1$ **do**
  Find the quantized weights $\overline{\mathbf{W}}^{(t)}$ from (8).
  Encode $\overline{\mathbf{W}}^{(t)}$ into $\{\widetilde{\mathbf{W}}_i^{(t)}\}_{i\in[N]}$ using (12).
  Send $\widetilde{\mathbf{W}}_i^{(t)}$ to worker $i \in [N]$.
  Workers:
  for worker $i = 1, \ldots, N$ **do**
    Compute $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)})$ from (18) and send the result back to the master.
  end for
  Master:
  if Results received from at least $(2r+1)(K+T-1)+1$ workers **then**
    Decode $\{f(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)})\}_{k\in[K]}$ via polynomial interpolation from the received results.
  end if
  Compute $\sum_{k=1}^{K} f(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)})$ in (22) and convert it from finite field to real domain using (23).
  Update the weights via (17) to obtain $\mathbf{w}^{(t+1)}$.
end for
return $\mathbf{w}^{(J)}$

the decoded computations $f(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)})$ to compute the desired gradient as,

$$\sum_{k=1}^{K} f\Big(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)}\Big) = \sum_{k=1}^{K} \overline{\mathbf{X}}_k^\top \bar{s}\Big(\overline{\mathbf{X}}_k, \overline{\mathbf{W}}^{(t)}\Big) = \overline{\mathbf{X}}^\top \bar{s}\Big(\overline{\mathbf{X}}, \overline{\mathbf{W}}^{(t)}\Big). \quad (22)$$

Lastly, master converts (22) from the finite field to the real domain and updates the weights according to (17) in the real domain. This conversion is attained by the function

$$Q_p^{-1}(\bar{x}; l) = 2^{-l} \cdot \phi^{-1}(\bar{x}), \quad (23)$$

where we let $l = l_x + r(l_x + l_w)$, and $\phi^{-1} : \mathbb{F}_p \to \mathbb{R}$ is defined as,

$$\phi^{-1}(\bar{x}) = \begin{cases} \bar{x} & \text{if} \quad 0 \le \bar{x} < \frac{p-1}{2} \\ \bar{x} - p & \text{if} \quad \frac{p-1}{2} \le \bar{x} < p \end{cases}. \quad (24)$$

The overall procedure of CodedPrivateML is given in Algorithm 1.

IV. Theoretical Results

Consider the cost function (1) when the dataset $\mathbf{X}$ is replaced with the quantized dataset $\overline{\mathbf{X}}$. Also, denote $\mathbf{w}^*$ as the optimal weight vector that minimizes (1) when $\hat{y}_i = s(\overline{\mathbf{x}}_i \cdot \mathbf{w})$, where $\overline{\mathbf{x}}_i$ is row i of $\overline{\mathbf{X}}$. In this section, we prove that CodedPrivateML guarantees convergence to the optimal model parameters (i.e., $\mathbf{w}^*$) while maintaining the privacy of the dataset against colluding workers. Recall that the model update at the master follows from (17), which is

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta}{m} \overline{\mathbf{X}}^\top \Big(\bar{s}\Big(\overline{\mathbf{X}}, \overline{\mathbf{W}}^{(t)}\Big) - \mathbf{y} \Big). \quad (25)$$

We first state a lemma, which shows that the gradient estimation of CodedPrivateML is unbiased and variance bounded.

Lemma 1: Let $\mathbf{p}^{(t)} \triangleq \frac{1}{m} \overline{\mathbf{X}}^\top (\bar{s}(\overline{\mathbf{X}}, \overline{\mathbf{W}}^{(t)}) - \mathbf{y})$ denote the gradient computation using the quantized weights $\overline{\mathbf{W}}^{(t)}$ in CodedPrivateML. Then, we have

- (Unbiasedness) Vector $\mathbf{p}^{(t)}$ is an asymptotically unbiased estimator of the true gradient. $\mathbb{E}[\mathbf{p}^{(t)}] = \nabla C(\mathbf{w}^{(t)}) + \epsilon(r)$, and $\epsilon(r) \to \mathbf{0}$ as $r \to \infty$ where r is the degree of the polynomial in (13) and the expectation is taken with respect to the quantization errors,
- (Variance bound) $\mathbb{E}[\|\mathbf{p}^{(t)} - \mathbb{E}[\mathbf{p}^{(t)}]\|_2^2] \le \frac{1}{2^{-2l_w} m^2} \| \overline{\mathbf{X}} \|_F^2 \triangleq \sigma^2$ where $\|\cdot\|_2$ and $\|\cdot\|_F$ are the l_2 and Frobenius norms, respectively.

Proof: The proof of Lemma 1 is presented in Appendix A in the supplementary material. ∎

We also need the following basic lemma, which describes the L-Lipschitz property of the gradient of the cost function.

Lemma 2: The gradient of the cost function from (1) evaluated on the quantized dataset $\overline{\mathbf{X}}$ is L-Lipschitz with $L \triangleq \frac{1}{4}\|\overline{\mathbf{X}}\|_2^2$, that is, $\|\nabla C(\mathbf{w}) - \nabla C(\mathbf{w}')\| \le L\|\mathbf{w} - \mathbf{w}'\|$ for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$.

Proof: The proof of Lemma 2 is presented in Appendix B in the supplementary material. ∎

We now state our main result for the theoretical performance guarantees of CodedPrivateML.

Theorem 1: Consider the training of a logistic regression model in a distributed system with N workers using CodedPrivateML with the dataset $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_K)$, initial weight vector $\mathbf{w}^{(0)}$, and constant step size $\eta = 1/L$ (where L is defined in Lemma 2). Then, CodedPrivateML guarantees,

- (Convergence) $\mathbb{E}[C(\frac{1}{J}\sum_{t=0}^{J} \mathbf{w}^{(t)})] - C(\mathbf{w}^*) \le \frac{\|\mathbf{w}^{(0)} - \mathbf{w}^*\|^2}{2\eta J} + \eta\sigma^2$ in J iterations, where σ^2 is given in Lemma 1,
- (Privacy) $\mathbf{X}$ remains information-theoretically private against any T colluding workers, i.e., $I(\mathbf{X}; \widetilde{\mathbf{X}}_{\mathcal{T}}, \{\widetilde{\mathbf{W}}_{\mathcal{T}}^{(t)}\}_{t\in[J]}) = 0$ for any distribution on $\mathbf{X}$ and any set $\mathcal{T} \subset [N]$ with $|\mathcal{T}| \le T$,

for any $N \ge (2r+1)(K+T-1)+1$, where r is the degree of the polynomial from (13).

Remark 2: Theorem 1 reveals an important trade-off between privacy and parallelization in CodedPrivateML. Parameter K reflects the amount of parallelization in CodedPrivateML, since the computation load at each worker node is proportional to $1/K$-th of the dataset. Parameter T reflects the privacy threshold in CodedPrivateML. Theorem 1 shows that, in a cluster with N workers, we can achieve any K and T as long as $N \ge (2r+1)(K+T-1)+1$. This condition further implies that, as the number of workers N increases, the parallelization (K) and privacy threshold (T) of CodedPrivateML can also increase linearly, leading to a scalable solution.

Remark 3: There are two terms in the bound on the distance between the loss function to the optimum in the first equation of Theorem 1, i.e., $\mathbb{E}[C(\frac{1}{J}\sum_{t=0}^{J} \mathbf{w}^{(t)})] - C(\mathbf{w}^*) \le$

$\frac{\| \mathbf{w}^{(0)} - \mathbf{w}^* \|^2}{2\eta J} + \eta\sigma^2$. When we use a constant learning rate $\eta = 1/L$, the first term $\frac{\| \mathbf{w}^{(0)} - \mathbf{w}^* \|^2}{2\eta J} = \frac{L\| \mathbf{w}^{(0)} - \mathbf{w}^* \|^2}{2J}$ goes to zero as the number of iterations J increases, hence CodedPrivateML has the convergence rate of $O(1/J)$. The second term $\eta\sigma^2 = \frac{\sigma^2}{L}$ is a residual error in the training as it does not go to zero as J increases. By using an adaptive (decreasing) learning rate, this term can be made arbitrarily small.

Remark 4: The convergence rate of CodedPrivateML is the same as that of conventional logistic regression. This follows from Theorem 1 where the convergence rate of CodedPrivateML is found as $O(\frac{1}{J})$ where J is the iteration index, which is the same as the convergence rate of conventional logistic regression, which follows from [44, Sec. 9.3] and [44, Sec. 7.1.1].

Remark 5: Theorem 1 applies also to (simpler) linear regression. The proof follows the same steps.

Proof: The proof of Theorem 1 is presented in Appendix C in the supplementary material. ∎

A. Complexity Analysis

In this section, we analyze the asymptotic complexity of CodedPrivateML with respect to the number of workers N, parallelization parameter K, privacy parameter T, number of samples m, number of features d, and number of iterations J.

Complexity Analysis of the Master Node: Computation cost of the master node can be broken into three parts: 1) encoding the dataset by using $\widetilde{\mathbf{X}}_i = u(\alpha_i)$ from (9) for $i \in [N]$, 2) encoding the weight vector by using $\widetilde{\mathbf{W}}_i^{(t)} = v(\alpha_i)$ from (11) for $i \in [N], t \in [J]$, and 3) decoding the gradient by recovering $h(\beta_i)$ in (19) for $i \in [K]$. For the first part, the encoded dataset $\widetilde{\mathbf{X}}_i$ ($i \in [N]$) from (9) is a weighted sum of $K+T$ matrices where the size of each matrix is $\frac{m}{K} \times d$. The Lagrangian coefficients can be calculated offline since the sets of $\{\alpha_i\}_{i\in[N]}$ and $\{\beta_j\}_{j\in[K]}$ are public. Each encoding requires $O(\frac{md(K+T)}{K})$ multiplications and we must perform N encodings, resulting in a total computational cost of $O(\frac{mdN(K+T)}{K})$. Decoding the gradient computations from (21) can be performed via a weighted sum of $(2r+1)(K+T-1)+1 = O(N)$ vectors where the size of each vector is d. Each decoding requires $O(dN)$ multiplications and we require K decoded gradients, resulting in a total computational cost of $O(dJNK)$. Communication cost of the master node to send the encoded dataset $\widetilde{\mathbf{X}}_i$ and the encoded weight vector $\widetilde{\mathbf{W}}_i^{(t)}$ to worker $i \in [N]$ is $O(\frac{mdN}{K})$ and $O(drNJ)$, respectively. Communication cost of the master to receive the local computation $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)})$ from worker $i \in [N]$ for $t \in [J]$ is $O(dJN)$.

Complexity Analysis of the Workers: Computation cost of worker i to compute $\widetilde{\mathbf{X}}_i^\top \widetilde{\mathbf{X}}_i$, the dominant part of the local computation $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{w}}_i^{(t)})$ in (18), is $O(\frac{md^2}{K})$. This corresponds to $O(\frac{1}{K})^{th}$ of the computation cost of conventional logistic regression, which requires the computation of $\mathbf{X}^\top s(\mathbf{X} \times \mathbf{w}^{(t)})$ in (2). This is due to the fact that the size of the encoded dataset $\widetilde{\mathbf{X}}_i$ and original dataset $\mathbf{X}$ are $\frac{m}{K} \times d$ and $m \times d$, respectively. Communication cost of worker i to receive the encoded dataset $\widetilde{\mathbf{X}}_i$ and the encoded weight vector $\widetilde{\mathbf{W}}_i^{(t)}$ for $t \in [J]$ is $O(\frac{md}{K})$ and $O(drJ)$, respectively. Communication cost of worker i to send the local computation $f(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{W}}_i^{(t)})$ to the master for $t \in [J]$ is $O(dJ)$.

We summarize the asymptotic complexity of CodedPrivateML in Table I.

TABLE I
COMPLEXITY SUMMARY OF CODEDPRIVATEML

	Computation	Communication
Master	$O\big(\frac{mdN(K+T)}{K} + drJN(K+T)\big)$	$O(\frac{mdN}{K} + drNJ)$
Worker	$O(\frac{md^2}{K})$	$O(\frac{md}{K} + drJ)$

V. EXPERIMENTS

We now experimentally demonstrate the performance of CodedPrivateML compared to conventional MPC baselines. Our focus is on training a logistic regression model for image classification, while the computation load is distributed to multiple machines on the Amazon EC2 Cloud Platform.

Experiment Setup: We train the logistic regression model from (1) for binary image classification on the CIFAR-10 [22] and GISETTE [23] datasets to experimentally examine two things: the accuracy of CodedPrivateML and the performance gain in terms of training time. The size of the CIFAR-10 and GISETTE datasets are $(m, d) = (9019, 3073)$[2] and $(6000, 5000)$, respectively. We implement the communication phase using the `MPI4Py` [45] message passing interface on `Python`. Computations are performed in a distributed manner on Amazon EC2 clusters using `m3.xlarge` machine instances.

We then compare CodedPrivateML with two MPC-based benchmarks that we apply to our problem. In particular, we implement two MPC constructions. The first one is based on the well-known BGW protocol [10], whereas the second one is a more recent protocol from [11], [12] that trade-offs offline calculations for a more efficient implementation. Our choice of these MPC benchmarks is due to their ability to be applied to a large number of workers. While several more recent works exist that have developed MPC-based training protocols with information-theoretic privacy guarantees, their constructions are limited to three or four parties [15], [20], [21]. For instance, [15] is a two-party protocol that requires two non-colluding workers.

Both baselines utilize Shamir's secret sharing scheme [19] where the dataset is secret shared among the N workers. For the (quantized) dataset $\overline{\mathbf{X}}$, this is achieved by creating a random polynomial $\mathbf{P}(z) = \overline{\mathbf{X}} + z\mathbf{Z}_1 + \cdots + z^T\mathbf{Z}_T$, where $\mathbf{Z}_j$ for $j \in [T]$ are i.i.d. uniformly distributed random matrices. This guarantees privacy against $\lfloor \frac{N-1}{2} \rfloor$ colluding workers [10]–[12], but requires a computation load at each worker that is as large as processing the whole dataset at a single worker, leading to slow training. Hence, in order to provide a fair comparison with CodedPrivateML, we optimize (speed up) the benchmark protocols by partitioning the users into subgroups of size

[2]We select images with the label of "plane" and "car", and the number of these images in 50000 training samples is 9019. For the number of features, we added a bias term, hence, we have $3072 + 1 = 3073$ features.

$2T+1$. Then, we let each group compute the gradient over the partitioned dataset $\overline{\mathbf{X}}_i' \in \mathbb{F}_p^{\frac{m}{G}\times d}$, where $\overline{\mathbf{X}} = [\overline{\mathbf{X}}_1'^{\top} \dots \overline{\mathbf{X}}_G'^{\top}]^{\top}$ and G is the number of subgroups. For group $i \in [G]$, each worker receives a share of the partitioned dataset by using a random polynomial $\mathbf{P_i}(z) = \overline{\mathbf{X}}_i' + z\mathbf{Z}_{i1} + \cdots + z^T\mathbf{Z}_{iT}$, where $\mathbf{Z}_{ij}$ for $j \in [T]$ and $i \in [G]$ are i.i.d. uniformly distributed random matrices. Workers then proceed with a multiround protocol to compute the sub-gradient. We further incorporate our quantization and approximation techniques in our benchmark implementations as conventional MPC protocols are also bound to arithmetic operations over a finite field. In our experiments, we set $G = 3$, hence the total amount of data stored at each worker is equal to one third of the size of the dataset $\overline{\mathbf{X}}$, which significantly reduces the total training time of the two benchmarks, while providing a privacy threshold of $T = \lfloor \frac{N-3}{6} \rfloor$. The implementation details of the MPC operations are provided in Appendix D in the supplementary material.

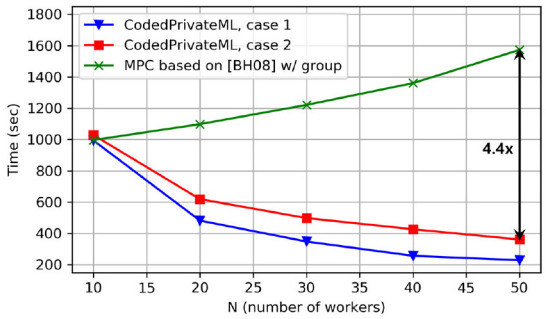
CodedPrivateML Parameters: There are several system parameters in CodedPrivateML that should be set. Given that we have a 64-bit implementation, we select the field size to be $p = 2^{25} - 37$, which is the largest prime with 25 bits to avoid an overflow on intermediate multiplications. We then optimize the quantization parameters, l_x in (4) and l_w in (7), by taking into account the trade-off between the rounding and overflow error. In particular, we choose $(l_x, l_w) = (2, 6)$ and $(2, 5)$ for the CIFAR-10 and GISETTE datasets, respectively. We also need to set the parameter r, the degree of the polynomial for approximating the sigmoid function. We consider both $r = 1$ and $r = 2$ and as shown later empirically we observe that the degree one approximation achieves good accuracy. We finally need to select T (privacy threshold) and K (amount of parallelization) in CodedPrivateML. As stated in Theorem 1, these parameters should satisfy $N \geq (2r+1)(K+T-1)+1$. Given our choice of $r = 1$, we consider two cases:

- *Case 1 (Maximum Parallelization):* All resources allocated for parallelization (faster training) by setting $K = \lfloor \frac{N-1}{3} \rfloor$, $T = 1$,
- *Case 2 (Equal Parallelization & Privacy):* Resources split almost equally between parallelization & privacy, i.e., $T = \lfloor \frac{N-3}{6} \rfloor$, $K = \lfloor \frac{N+2}{3} \rfloor - T$.

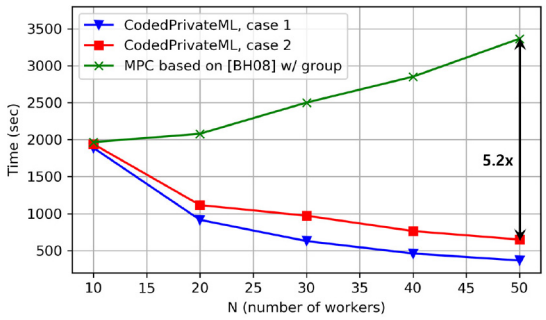
Training Time: Initially, we measure the training time while increasing the number of workers N gradually. Our results are demonstrated in Figure 3, which shows the comparison of CodedPrivateML with the [BH08] protocol from [11], as we have found it to be the faster of the two benchmarks. In particular, we make the following observations.[3]

- CodedPrivateML provides substantial speedup over the MPC baselines, in particular, up to 4.4× and 5.2× with the CIFAR-10 and GISETTE datasets, respectively, while providing the same privacy threshold as the benchmarks ($T = \lfloor \frac{N-3}{6} \rfloor$ for Case 2). Table II demonstrates the breakdown of the total runtime with the CIFAR-10 dataset for $N = 50$ workers. In this scenario, CodedPrivateML provides significant improvement in all three categories of dataset encoding and secret sharing; communication time between the workers and the master; and the computation time. Main reason for this is that, in the MPC baselines, the size of the data processed at each worker is one third of the original dataset, while in CodedPrivateML it is $1/K$-th of the dataset. This reduces the computational overhead of each worker while computing matrix multiplications as well as the communication overhead between the master and workers. We also observe that a higher amount of speedup is achieved as the dimension of the dataset becomes larger (CIFAR-10 vs. GISETTE datasets), suggesting CodedPrivateML to be well-suited for data-intensive training tasks where parallelization is essential.
- The total runtime of CodedPrivateML decreases as the number of workers increases. This is again due to the parallelization gain of CodedPrivateML (i.e., increasing K while N increases). This is not achievable in conventional MPC baselines, since the size of data processed at each worker is constant for all N.

[3]For $N = 10$, all schemes have similar performance because the total amount of data stored at each worker is one third of the size of whole dataset ($K = 3$ for CodedPrivateML and $G = 3$ for the benchmark).



(a) CIFAR-10 (for accuracy 81.35% with 50 iterations)

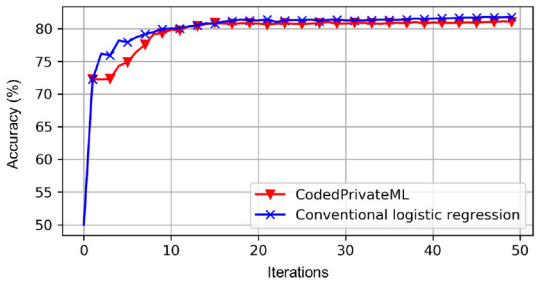


(b) GISETTE (for accuracy 97.50% with 50 iterations)

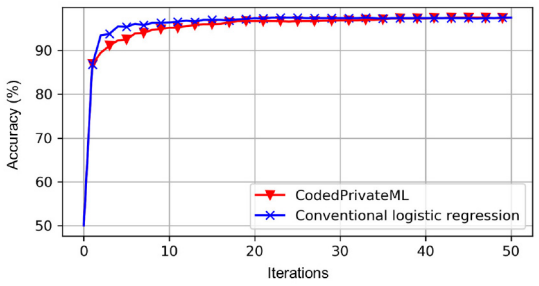
Fig. 3. Performance gain of CodedPrivateML over the MPC baseline ([BH08] from [11]). The plot shows the total training time for different number of workers N.

TABLE II
(CIFAR-10) BREAKDOWN OF TOTAL RUNTIME FOR $N = 50$

Protocol	Enc. time (s)	Comm. time (s)	Comp. time (s)	Total time (s)
MPC using [BGW88]	202.78	31.02	7892.42	8127.07
MPC using [BH08]	201.08	30.25	1326.03	1572.34
CodedPrivateML (Case 1)	59.93	4.76	141.72	229.07
CodedPrivateML (Case 2)	91.53	8.30	235.18	361.08



(a) CIFAR-10 dataset, binary classification between *car* and *plane* images (using 9019 samples for the training set and 2000 samples for the test set).



(b) GISETTE dataset, binary classification between the images of digits 4 and 9 (using 6000 samples for the training set and 1000 samples for the test set).

Fig. 4. Comparison of the accuracy of CodedPrivateML (demonstrated for Case 2 and $N = 50$ workers) vs conventional logistic regression that uses the sigmoid function without quantization.

- Increasing N in CodedPrivateML has two major impacts on the total training time. The first one is reducing the computation load per worker, as each new worker can be used to increase the parameter K. This in turn reduces the computation load per worker as the amount of work done by each worker is scaled with respect to $1/K$. The second one is that increasing the number of workers increases the encoding time at the master node. Hence, the gain from increasing the number of workers beyond a certain point may be minimal and the system may saturate. In those cases, increasing the number of workers cannot further reduce the training time, as the computation will be dominated by the encoding overhead.
- CodedPrivateML provides up to 22.5× speedup over the BGW protocol [10], as shown in Table II for the CIFAR-10 dataset with $N = 50$ workers. This is due to the fact that BGW requires additional communication between the workers to execute a degree reduction phase for every multiplication operation.

Accuracy: We also examine the accuracy and convergence of CodedPrivateML. Figure 4(a) illustrates the test accuracy of the binary classification problem between *plane* and *car* images for the CIFAR-10 dataset. With 50 iterations, the accuracy of CodedPrivateML with degree one polynomial approximation and conventional logistic regression are 81.35% and 81.75%, respectively. Figure 4(b) shows the test accuracy for binary classification between digits 4 and 9 for the GISETTE dataset. With 50 iterations, the accuracy

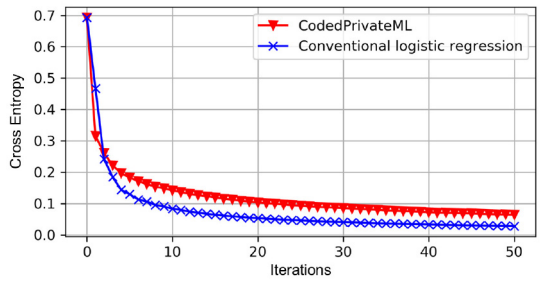


Fig. 5. Convergence of CodedPrivateML (demonstrated for Case 2 and $N = 50$ workers) vs conventional logistic regression (using the sigmoid function without polynomial approximation or quantization).

of CodedPrivateML with degree one polynomial approximation and conventional logistic regression has the same value of 97.5%. Hence, CodedPrivateML has comparable accuracy to conventional logistic regression while being privacy preserving.

Figure 5 presents the cross entropy loss for CodedPrivateML versus the conventional logistic regression model for the GISETTE dataset. The latter setup uses the sigmoid function and no polynomial approximation, in addition, no quantization is applied to the dataset or the weight vectors. We observe that CodedPrivateML achieves convergence with comparable rate to conventional logistic regression, while being privacy preserving.

VI. Conclusion and Discussion

In this article, we considered a distributed training scenario in which a data-owner wants to train a logistic regression model by off-loading the computationally-intensive gradient computations to multiple workers, while preserving the privacy of the dataset. We proposed a privacy-preserving training framework, CodedPrivateML, that distributes the computation load effectively across multiple workers, and reduces the per-worker computation load as more and more workers become available. We demonstrated the theoretical convergence guarantees and the fundamental trade-offs of our framework, in terms of the number of workers, privacy protection, and scalability. Our experiment results demonstrate significant speedup in the training time compared to conventional baseline protocols.

This work focuses on a logistic regression model mainly with the goal of demonstrating how CodedPrivateML can be utilized to scale and speed up logistic regression training under privacy and convergence guarantees, which is a first step towards more complex models. To the best of our knowledge, even for this setup, no other system has been able to efficiently scale beyond 3−4 workers while achieving information-theoretic privacy. Our work is the first privacy-preserving machine learning approach that reduces the communication and computation load per worker as the number of workers increases, which we hope will open up further research. Future directions include extending CodedPrivateML to deeper neural networks by leveraging an MPC-friendly (i.e., polynomial) activation function or extending CodedPrivateML to collaborative learning setting such as [46].

In this article, in order to provide information-theoretic privacy, we utilize quantization to convert the dataset and model to the finite field $\mathbb{F}_p$. Doing so has two inherent challenges: 1) determining a proper value for p and 2) potential performance degradation caused by quantization or overflow error. This has inspired a new line of works, such as *analog coded computing* [47], [48], which uses floating-point numbers instead of fixed-point numbers to represent the finite field and provides a fundamental trade-off between the accuracy and privacy level. Leveraging such techniques to address these challenges is another interesting future direction.

ACKNOWLEDGMENT

The authors would like to thank helpful comments and discussions with Dr. Payman Mohassel on this problem. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 38–67, 1st Quart., 2019.

[2] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. ACM Conf. Comput. Commun. Security*, 2009, pp. 199–212.

[3] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 305–316.

[4] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-tenant side-channel attacks in PaaS clouds," in *Proc. ACM Conf. Comput. Commun. Security*, 2014, pp. 990–1003.

[5] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-bandwidth and reliable covert channel attacks inside the cloud," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 603–615, Apr. 2015.

[6] V. Varadarajan, Y. Zhang, T. Ristenpart, and M. Swift, "A placement vulnerability study in multi-tenant public clouds," in *Proc. 24th USENIX Security Symp.*, 2015, pp. 913–928.

[7] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip Feng Shui: Hammering a needle in the software stack," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 1–18.

[8] Q. Yu *et al.*, "Lagrange coded computing: Optimal design for resiliency, security and privacy," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2019, pp. 1215–1225.

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

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

[11] Z. Beerliová-Trubíniová and M. Hirt, "Perfectly-secure MPC with linear communication complexity," in *Proc. Theory Cryptol. Conf.*, 2008, pp. 213–230.

[12] I. Damgård and J. B. Nielsen, "Scalable and unconditionally secure multiparty computation," in *Proc. Int. Cryptol. Conf.*, 2007, pp. 572–590.

[13] 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. Security Privacy*, 2013, pp. 334–348.

[14] A. Gascón *et al.*, "Privacy-preserving distributed linear regression on high-dimensional data," in *Privacy Enhancing Technologies*. Heidelberg, Germany: Springer, 2017, pp. 345–364.

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

[16] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Proc. Annu. Int. Cryptol. Conf.*, 2000, pp. 36–54.

[17] M. Dahl *et al.*, "Private machine learning in TensorFlow using secure computation," 2018. [Online]. Available: arXiv:1810.08130.

[18] V. Chen, V. Pastro, and M. Raykova, "Secure computation for machine learning with SPDZ," 2019. [Online]. Available: arXiv:1901.00329.

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

[20] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: Efficient and private neural network training," Cryptol. ePrint Archive, Rep. 2018/442, 2018. [Online]. Available: https://eprint.iacr.org/2018/442.

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

[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Rep., 2009.

[23] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 545–552.

[24] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*, Stanford Univ., Stanford, CA, USA, 2009.

[25] 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.

[26] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," 2017. [Online]. Available: arXiv:1711.05189.

[27] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Proc. Int. Conf. Inf. Security Cryptol.*, 2012, pp. 1–21.

[28] 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.

[29] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," in *Proc. IACR Cryptol. ePrint Arch.*, vol. 2017, 2017, p. 35.

[30] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Clust. Comput.*, vol. 21, pp. 277–286, Apr. 2017.

[31] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC Med. Genom.*, vol. 11, no. 4, pp. 23–55, Oct. 2018.

[32] 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.

[33] K. Han, S. Hong, J. H. Cheon, and D. Park, "Logistic regression on homomorphic encrypted data at scale," in *Proc. Annu. Conf. Innov. Appl. Artif. Intell.*, 2019, pp. 9466–9471.

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

[35] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 289–296.

[36] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. ACM Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.

[37] M. Abadi, "Deep learning with differential privacy," in *Proc. ACM Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[38] 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.

[39] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. on Learn. Rep.*, 2018.

[40] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2012, pp. 933–941.

[41] 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.

[42] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang, "ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Aug. 2017, pp. 4035–4043.

[43] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang, "The ZipML framework for training models with end-to-end low precision: The cans, the cannots, and a little bit of deep learning," 2016. [Online]. Available: arXiv:1611.05402.

[44] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[45] L. Dalcín, R. Paz, and M. Storti, "MPI for Python," *J. Parallel Distrib. Comput.*, vol. 65, no. 9, pp. 1108–1115, 2005.
[46] J. So, B. Guler, and A. S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.
[47] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog lagrange coded computing," 2020. [Online]. Available: arXiv:2008.08565.
[48] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," 2020. [Online]. Available: arXiv:2007.08803.

Jinhyun So received the B.S. and M.S. degrees in electrical and computer engineering from KAIST. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Southern California. His research interests include information theory, large-scale distributed machine learning, and secure and private computing. He received the Annenberg Graduate Fellowship in 2017.

Başak Güler (Member, IEEE) received the Ph.D. degree from the Pennsylvania State University in 2017, and was a Postdoctoral Scholar with the University of Southern California from 2018 to 2020. She is an Assistant Professor with the Department of Electrical and Computer Engineering, University of California Riverside, Riverside, CA, USA. Her research interests include machine learning in wireless networks, information theory, distributed computing, and signal processing.

A. Salman Avestimehr (Fellow, IEEE) received the B.S. degree in electrical engineering from the Sharif University of Technology in 2003, the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California Berkeley, Berkeley, CA, USA, in 2005 and 2008, respectively.

He is a Professor and the Director of the Information Theory and Machine Learning Research Lab with the Electrical and Computer Engineering Department, University of Southern California. His research interests include information theory and coding theory, and large-scale distributed computing and machine learning, secure and private computing, and blockchain systems. He has received a number of awards for his research, including the James L. Massey Research & Teaching Award from IEEE Information Theory Society, an Information Theory Society, and the Communication Society Joint Paper Award, the Presidential Early Career Award for Scientists and Engineers from the White House (President Obama), the Young Investigator Program Award from the U.S. Air Force Office of Scientific Research, the National Science Foundation CAREER Award, the David J. Sakrison Memorial Prize, and several Best Paper Awards at Conferences. He has been an Associate Editor for IEEE TRANSACTIONS ON INFORMATION THEORY and the General Co-Chair of the 2020 International Symposium on Information Theory.