Analog Lagrange Coded Computing

Mahdi Soleymani, Hessam Mahdavifar, and A. Salman Avestimehr

***Abstract*—A distributed computing scenario is considered, where the computational power of a set of worker nodes is used to perform a certain computation task over a dataset that is dispersed among the workers. Lagrange coded computing (LCC), proposed by Yu et al., leverages the well-known Lagrange polynomial to perform polynomial evaluation of the dataset in such a scenario in an efficient parallel fashion while keeping the privacy of data amidst possible collusion of workers. This solution relies on quantizing the data into a finite field, so that Shamir's secret sharing, as one of its main building blocks, can be employed. Such a solution, however, is not properly scalable with the size of dataset, mainly due to computation overflows. To address such a critical issue, we propose a novel extension of LCC to the analog domain, referred to as analog LCC (ALCC). All the operations in the proposed ALCC protocol are done over the infinite fields of $\mathbb{R}/\mathbb{C}$ but for practical implementations floating-point numbers are used. We characterize the *privacy* of data in ALCC, against any subset of colluding workers up to a certain size, in terms of the distinguishing security (DS) and the mutual information security (MIS) metrics. Also, the *accuracy* of outcome is characterized in a practical setting assuming operations are performed using floating-point numbers. Consequently, a fundamental trade-off between the accuracy of the outcome of ALCC and its privacy level is observed and is numerically evaluated. Moreover, we implement the proposed scheme to perform matrix-matrix multiplication over a batch of matrices. It is observed that ALCC is superior compared to the state-of-the-art LCC, implemented using fixed-point numbers, assuming both schemes use an equal number of bits to represent data symbols.**

***Index Terms*—Coded computing, privacy-preserving computing, analog coding**

I. Introduction

There has been a growing interest in recent years towards performing computational tasks across networks of computational worker nodes by utilizing their computational power in a parallel fashion [1]–[5]. Computations over massive datasets need to be carried out at an unprecedented scale that entails solutions scalable with the size of datasets associated with a wide range of problems including machine learning [6], optimization [7], etc. A well-established network architecture to perform such tasks in a distributed fashion consists of a *master* node together with a set of worker nodes having communication links only with the master node [3], [4]. In such systems, a dataset is dispersed among the servers across the network to perform a certain computational task over the dataset. The master node then aggregates the results in order to recover the desired outcome, e.g., the output of a certain function over the dataset.

This work was supported by the National Science Foundation under grants CCF–1763348, CCF–1909771, CCF–1941633.

M. Soleymani and H. Mahdavifar are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48104 (email: mahdy@umich.edu and hessam@umich.edu).

A. Salman Avestimehr is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: avestimehr@ee.usc.edu).

Dispersing data across a network gives rise to several fundamental challenges in practice. One of the major concerns in such systems is to keep the data *private* as the computational tasks often involve sensitive data such as patients recordings, financial transactions, etc [8]–[10]. The worker nodes are often assumed to be *honest-but-curious*, i.e., they do not deviate from the protocol but may accumulate the shares of data they receive and try to deduce information about the data. In such settings, the challenge is to utilize the computational power of the nodes while ensuring that *almost* no information about the dataset is revealed to them. Furthermore, this restriction is often extended to preserving the privacy of data against any subset of colluding nodes up to a certain size.

Several security metrics are considered in different contexts to measure privacy/security of data. This includes semantic security (SS) and distinguishing security (DS) in the cryptography literature [11], mutual information security (MIS) in communication settings [12], differential security in machine learning [13], etc. From the information-theoretic perspective, the *perfect* privacy condition in a distributed computation setting is that *no information* is leaked about the dataset to any of the worker nodes/subsets of colluding worker nodes up to a certain size. To this end, Shamir's seminal secret sharing scheme is the main building block in protocols providing *perfect* privacy in these settings [14]. In such protocols, the data symbols are always assumed to be elements of a finite field $\mathbb{F}_p$ leading to perfect privacy guarantees. However, this often comes at the expense of substantial accuracy losses due to fixed-point representation of the data and computation overflows. Especially, this becomes a major barrier in scalability of such protocols with respect to the dataset size.

The seminal Shamir's secret sharing scheme and its various versions are often used to provide information-theoretic security for data, referred to as a secret, while distributing it among a set of servers/users [14]. Also, Shamir's scheme serves as the backbone of most of the existing schemes on privacy-preserving multiparty computing such as the celebrated BGW scheme [15]. In (n, k) Shamir's secret sharing scheme, the secret, which is regarded as an element of a finite field, is encoded to a polynomial of degree $k-1$ whose constant coefficient is the secret and all other coefficients are picked uniformly at random from the field. The shares are then evaluations of the polynomial at n distinct points. The secret can be uniquely decoded when at least k number of shares are available while no information is revealed about the secret otherwise. In order to employ Shamir-based distributed computing protocols the data is quantized and then mapped to a finite field at the beginning. This leads to a possibly substantial loss in the accuracy of the computation results mainly due to computation overflows when the dataset is *large*. In order to overcome this issue, an analog counterpart of Shamir's scheme is recently proposed in [16] and is then utilized to perform a learning task when the data is provided using floating-point numbers. Lagrange

coded computing (LCC) [4] provides a framework to efficiently perform distributed computation over a batch of data in a parallel fashion. It can be utilized to provide privacy-preserving machine learning schemes. Similar to Shamir's scheme, in LCC, the data is assumed to be an element of a finite field and the secret/data is encoded to a certain polynomial, called Lagrange interpolation polynomial. Hence, the loss in accuracy due to overflow in computations is inherited to LCC as well.

A. Our contribution

In this paper, we propose a framework to extend the privacy-preserving LCC scheme to the analog domain and refer to it as analog LCC (ALCC). It is assumed that all the worker nodes are honest-but-curious. All the operations in the proposed scheme are done over the infinite fields of $\mathbb{R}/\mathbb{C}$ but for practical implementations floating-point numbers are used. The proposed ALCC protocol enables *privately* evaluating a polynomial function over a batch of real/complex-valued dataset in parallel. We characterize the performance of the scheme in terms of the *accuracy* of its outcome, when operations are performed using standard floating-point numbers, and the *privacy* of data in terms of the DS and MIS metrics when any subset of worker nodes up to a certain size can collude. It is shown how various parameters of the ALCC protocol, including parameters associated with Lagrange monomials as well as evaluation points in the complex plane, can be carefully picked in order to provide closed-form bounds for the performance of the protocol from both the privacy and the accuracy perspectives. Furthermore, a fundamental trade-off between the accuracy of the outcome of ALCC and its privacy level is observed and is numerically evaluated, in terms of various parameters of the scheme, when the scheme is implemented using the floating-point numbers. In a related work, we show that accuracy-privacy trade-offs arise in distributed computing in the analog domain by tuning the noise variance in the underlying protocol [16]. However, the main distinction of the current paper is to illustrate that, even for a fixed noise variance, the choice of certain parameters of Lagrange monomials in ALCC leads to a new trade-off between accuracy and privacy which is specific to ALCC. Hence, one has to carefully pick these parameters apart from the noise variance in order to avoid unnecessarily compromising accuracy/privacy in practice. This is a new fundamental trade-off that does not have a counterpart in either analog adaptations of Shamir's scheme, e.g., [16], or LCC with fixed-point implementation over finite fields [4]. Also, it is numerically illustrated that ALCC scales better with the number of representation bits considered in the floating-point implementation compared to LCC. Moreover, experiments are shown in which the proposed protocol is implemented to perform matrix-matrix multiplication over a batch of matrices. The results indicate the superiority of the proposed ALCC compared to the state-of-the-art LCC implemented using fixed-point numbers assuming both schemes use an equal number of bits to represent each data symbol.

Note that LCC simultaneously provides resiliency against stragglers, security against malicious workers or workers with erroneous returned results, and privacy of the dataset [4]. Here, we are mainly concerned with the privacy of dataset in the analog domain. Our analysis of the proposed scheme also takes into account the issue with slow/unresponsive nodes, also referred to as stragglers. However, the issue with malicious workers is left for future work.

B. Related work

Privacy-preserving distributed computing protocols have been recently studied in a wide range of scenarios to fulfill specific privacy requirements [17]–[21]. Furthermore, secure matrix-matrix multiplication, as one of the main building blocks for various machine learning algorithms, has been extensively studied in the literature [22]–[27]. Also, Lagrange coded computing [4] and its variations [28], [29] provide a framework for evaluating a given polynomial function over a dataset with perfect privacy [4]. Such protocols have been recently adopted to perform various machine learning tasks. Recently, it is shown that LCC can be employed to break the aggregation barrier in secured federated learning [21]. However, these prior works often regard data as elements of a finite filed. As a result, they suffer from scalability issues, as discussed earlier. By enabling privacy in the analog domain, ALCC provides a framework to perform several large-scale tasks, e.g., secure aggregation in federated learning [21], more efficiently in practice.

There is also another line of work on privacy-preserving machine learning that utilizes off-the-shelf multi party computation (MPC) protocols [30], [31] to train a model over distributed datasets [1], [2], [17], [32]–[34]. In these MPC-based machine learning problems, often more than one client are assumed that aim at learning model parameters collaboratively without sharing sensitive data with each other and worker nodes. On the other hand, in (A)LCC, it is assumed that all the data is present in one central node/client, called the master node, that utilizes the computational power of worker nodes for speed up while keeping the data private from the workers. In other words, the MPC-based schemes mainly concern with the privacy of sensitive datasets over which a model is trained in a fully distributed fashion, while (A)LCC-based methods provide a framework for privacy-preserving machine learning in which the dataset is offloaded to a cloud-computing environment to gain speed up [19]. However, MPC-based ML schemes are also adopted in the case with one central client in the literature [19], [33]. In this approach, only a few number of worker nodes are often considered mainly due to inefficiency of underlying MPC schemes. In a recent work, a fully distributed implementation of LCC is introduced in [35], which is then utilized to train a linear regression model over distributed datasets without considering a *central entity* and is significantly faster than MPC-based methods. In general, (A)LCC schemes reduce the amount of randomness needed in data encoding and have less storage overhead as well as computation complexity. Moreover, no communication is needed between worker nodes in (A)LCC, a factor that contributes the most to the inefficiency of MPC-based ML in practical systems. There is also a line of work concerning with floating-point implementation of MPC protocols [36]–[38] which requires significantly more rounds of communications and computations compared to the conventional MPC protocols

with fixed-point implementation. As a result, the inefficiency of such protocols poses a major difficulty in their implementation as well.

Another line of work on performing computations over real-valued data is considered in [39]–[43]. In these works, the coded distributed computing schemes are adapted to the analog domain by addressing the numerical stability issues arising in the inversion of underlying Vandermonde matrices. However, the privacy constraints are not considered in these works. In this paper, however, our main focus is on providing privacy-preserving schemes in the analog domain. Also, codes in the analog domain have been recently studied in the context of block codes [44] as well as subspace codes [45] for analog error correction. However, secret sharing and privacy-preserving computation in the analog domain are not discussed in these works.

The rest of this paper is organized as follows. In Section II, the system model is discussed and the proposed protocol is described. The accuracy of the protocol is analyzed in Section III. In Section IV the privacy level of data in ALCC is characterized in terms of two well-known notions of security. Various experimental results are provided in Section V. Finally, the paper is concluded in Section VI.

II. System Model

Consider a dataset $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ with $\boldsymbol{X}_i \in \mathbb{R}^{m\times n}$ for all $i \in [k]$, where $[k]$ denotes $\{1, 2, \ldots, k\}$. Each entry of $\boldsymbol{X}_i$'s is assumed to be an instance of a continuous random variable with the range $[-r, r]$. No further assumptions is made on the probability distribution of the entries of $\boldsymbol{X}_i$'s.

We consider the problem of evaluating a polynomial $f : \mathbb{R}^{m\times n} \to \mathbb{R}^{u\times h}$ over the dataset $\boldsymbol{X}$ in a distributed fashion while keeping the *privacy* of $\boldsymbol{X}$. More specifically, we say $f(\cdot)$ is a D-degree polynomial function if all entries of the output matrix are multivariate polynomial functions of the entries of the input with total degree at most D, i.e., $\boldsymbol{Y} = f(\boldsymbol{X})$ implies that

$$y_{ij} = f_{ij}(x_{11}, x_{12}, \cdots, x_{mn}), \tag{1}$$

where y_{ij} is the (i, j) entry of $\boldsymbol{Y}$, for $i \in [u]$ and $j \in [h]$, x_{lk} is the (l, k) entry of $\boldsymbol{X}$, for $l \in [m]$ and $k \in [n]$, and, f_{ij} is a multivariate polynomial of total degree at most D. We may write the right hand side of (1) as $f_{ij}(\boldsymbol{X})$, or simply f_{ij} when the argument is clear from the context, throughout the rest of the paper. The distributed computing setup consists of a master node and N worker nodes/parties. It is assumed that there is no communication link between the parties. More specifically, in this setup, the goal of the master node is to compute $f(\boldsymbol{X}_i)$ for all $i \in [k]$, where f is a degree-D polynomial, using the computational power of the parties. This is done in such a way that the dataset is kept *private* from the parties assuming up to a certain threshold, denoted by t, of them can collude. The notion of privacy in the analog domain will be clarified in Section IV. Note that this setup is similar to the one considered for LCC in [4] with the main difference that in [4] the dataset and all the computations are assumed to be over a finite field. More specifically, our problem setup can be regarded as an extension of the problem setup considered in [4] to the analog domain.

Next we discuss the encoding process in analog Lagrange coded computing (ALCC), i.e., how to encode the dataset $\boldsymbol{X}$ into the shares distributed to the worker nodes. Let $\boldsymbol{W}$ denote $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k, \boldsymbol{N}_1, \ldots, \boldsymbol{N}_t)$ where $\boldsymbol{N}_i$'s are $m \times n$ random matrices with i.i.d. entries drawn from a zero-mean circular symmetric complex Gaussian distribution with standard deviation $\frac{\sigma_n}{\sqrt{t}}$, denoted by $\mathcal{N}(0, \frac{\sigma_n^2}{t})$, with t being the maximum number of colluding parties. Note that a zero-mean circular symmetric complex Gaussian random variable (RV) with variance σ^2 consists of two i.i.d. zero-mean Gaussian RV's with variance $\frac{\sigma^2}{2}$ as its real and imaginary part. Let γ and ω denote the N-th and the $(k + t)$-th root of unity, respectively. In other words, $\gamma = \exp(\frac{2\pi i}{N})$ and $\omega = \exp(\frac{2\pi i}{k+t})$, where $i^2 = -1$. In ALCC, the Lagrange polynomial is constructed as

$$u(z) = \sum_{j=1}^{k} \boldsymbol{X}_j l_j(z) + \sum_{j=k+1}^{k+t} \boldsymbol{N}_{j-k} l_j(z) = \sum_{j=1}^{k+t} \boldsymbol{W}_i l_j(z), \tag{2}$$

where $l_j(\cdot)$'s are Lagrange monomials defined as

$$l_j(z) = \prod_{l\in[k+t]\setminus j} \frac{z - \beta_l}{\beta_j - \beta_l}, \tag{3}$$

for all $j \in [k + t]$. Furthermore, the parameters β_j's are picked to be equally spaced on the circle of radius β centered around 0 in the complex plane, for some $\beta \in \mathbb{R}$, i.e.,

$$\beta_j = \beta\omega^{j-1}. \tag{4}$$

The shares of encoded dataset to be distributed to the worker nodes consist of the evaluation of $u(z)$ over the N-th roots of unity in the complex plane, i.e.,

$$\boldsymbol{Y}_i = u(\alpha_i), \tag{5}$$

where

$$\alpha_i = \gamma^{i-1}, \tag{6}$$

is sent to node i, for $i \in [N]$. The choice of α_i's and β_j's are demonstrated in Figure 1 for the choice of parameters $k = 6$, $t = 2$, and $N = 16$ in the complex plane. It will be clarified in Section III-A that the specific choice of β_j's according to (4) enables characterizing a closed-form upper bound on the *absolute error* of the outcomes of ALCC. In this context, the absolute error is the magnitude of the difference between the ALCC outcome in a practical setting and the true result of the computation.

Next, we discuss the decoding step during which the master node recovers the desired outcome by collecting and processing the results returned by a sufficient number of worker nodes. The i-th node computes $f(\boldsymbol{Y}_i)$ and returns the result back to the master node. The master node then recovers $f(\boldsymbol{X}_i)$, for $i \in [k]$, in two steps. In the first step, it recovers the polynomial $f(u(z))$ by using the results returned from at least $(k + t - 1)D + 1$ worker nodes. Note that this is the minimum number of returned evaluations needed to guarantee a successful interpolation of $f(u(z))$ since $f(u(z))$ has degree $(k + t - 1)D$. For ease of notation, let

$$\tilde{D} = (k + t - 1)D. \tag{7}$$

In the second step, to recover $f(\boldsymbol{X}_i)$'s, the master node com-

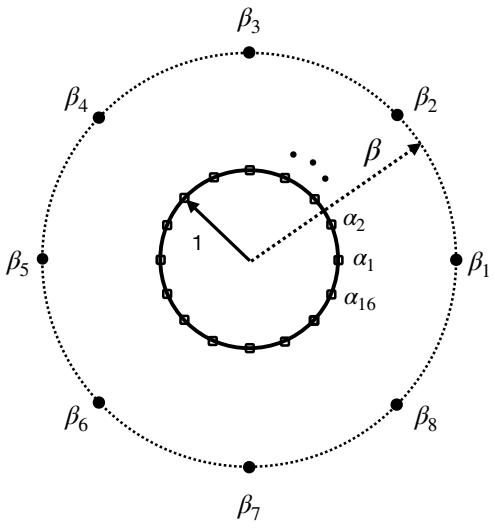

Fig. 1: Demonstration of the locations of α_i's and β_j's in the complex plane for $k = 6, t = 2, N = 16$. Both circles are centered at the origin.

putes $f(\beta_j)$ for $j \in [k]$. Note that $u(\beta_j) = \boldsymbol{W}_j$ for $j \in [k+t]$, since $l_j(\beta_i)$ is 1 for $i = j$ and is zero otherwise.

The ALCC protocol, as described above, can also take into account the issue regarding stragglers same as how it is done in LCC [4]. Let the maximum number of stragglers be denoted by s. Hence, the number of computational parties is assumed to be $N = \tilde{D} + s + 1$.

***Remark** 1:* In theory, if the computations are done over the complex numbers with infinite precision, then $f(\boldsymbol{X}_i)$'s are computed accurately. In practice, however, data is represented using a finite number of bits, either as fixed point or floating point. We assume floating-point representation for data symbols and operations involving them in our analysis. This is more suitable to mimic operations over complex (real) numbers. Let b_m denote the number of precision bits in the floating-point representation, referred to as the *mantissa*, and b_e denote the number of bits used to represent the exponent part, referred to as the *exponent*. Note that the entries of the noise matrices $\boldsymbol{N}_i$'s are bounded in practice. In other words, for practical purposes, it is assumed that the entries of the noise matrices are drawn from the Gaussian distribution that is truncated to $\left[-\theta\frac{\sigma_n}{\sqrt{t}},\ \theta\frac{\sigma_n}{\sqrt{t}}\right]$, for some $\theta \in \mathbb{R}$.

III. Accuracy Analysis

In this section, accuracy of the final outcome of ALCC, specified in Section II, is characterized in terms of various other parameters of the scheme. This is done by assuming that floating-point numbers are used to represent data symbols and to carry out operations involving them.

A. Analytical results

We start by providing an alternative characterization for the Lagrange monomials, defined in (3), given the certain values for β_j's specified in (4). This is done in the following lemma.

Lemma 1: For all $j \in [k+t]$, we have

$$l_j(z) = \frac{1}{k+t} \sum_{l=0}^{k+t-1} (\frac{z}{\beta_j})^l. \tag{8}$$

Proof: Using (3), one can write

$$l_j(z) = \prod_{l \in [k+t]\setminus j} \frac{\frac{z}{\beta_j} - \frac{\beta_l}{\beta_j}}{1 - \frac{\beta_l}{\beta_j}} = \prod_{h=1}^{k+t-1} \frac{\frac{z}{\beta_j} - \omega^h}{1 - \omega^h}. \tag{9}$$

Note that ω^h, for $h = 0, 1, \ldots, k+t-1$, is a $(k+t)$-th root of unity. Hence, we have

$$\prod_{h=1}^{k+t-1} (x - \omega^h) = \frac{x^{k+t} - 1}{x - 1} = \sum_{h=0}^{k+t-1} x^h. \tag{10}$$

Using (10) one can write

$$\prod_{h=1}^{k+t-1} (\frac{z}{\beta_j}) - \omega^h = \frac{(\frac{z}{\beta_j})^{k+t} - 1}{\frac{z}{\beta_j} - 1} = \sum_{h=0}^{k+t-1} (\frac{z}{\beta_j})^h, \tag{11}$$

and

$$\prod_{h=1}^{k+t-1} (1 - \omega^h) = \sum_{h=0}^{k+t-1} 1 = k + t. \tag{12}$$

Combining (11) and (12) completes the proof. ■

In the following lemma, we use Lemma 1 to characterize the coefficients of Lagrange polynomial in terms of $\boldsymbol{W}$ and other parameters of the scheme. The result will be used later to derive an upper bound on the absolute error of the outcome of ALCC.

Lemma 2: The Lagrange polynomial, as specified in (2), can be written as

$$u(z) = \sum_{l=0}^{k+t-1} \frac{\tilde{\boldsymbol{W}}_l}{\beta^l} z^l, \tag{13}$$

where

$$\tilde{\boldsymbol{W}}_l \stackrel{\text{def}}{=} \sum_{j=0}^{k+t-1} \boldsymbol{W}_j \omega^{-jl}. \tag{14}$$

Proof: The proof is by combining (2) and Lemma 1 as follows:

$$u(z) = \sum_{j=0}^{k+t-1} \boldsymbol{W}_j \frac{1}{k+t} \sum_{l=0}^{k+t-1} (\frac{z}{\beta_j})^l \tag{15}$$

$$= \frac{1}{k+t} \sum_{j,l=0}^{k+t-1} \boldsymbol{W}_j (\frac{z}{\beta})^l \omega^{-jl} \tag{16}$$

$$= \frac{1}{k+t} \sum_{l=0}^{k+t-1} (\frac{z}{\beta})^l (\sum_{j=0}^{k+t-1} \boldsymbol{W}_j \omega^{-jl}) \tag{17}$$

$$= \frac{1}{k+t} \sum_{l=0}^{k+t-1} \frac{\tilde{\boldsymbol{W}}_l}{\beta^l} z^l. \tag{18}$$

■

Let w_{gl}^j and $\tilde{w}_{gl}^j$ denote the (g,l) entry of $\boldsymbol{W}_j$ and $\tilde{\boldsymbol{W}}_j$, for $(g,l) \in [m] \times [n]$, respectively. Then (14) implies that $(\tilde{w}_{gl}^0, \ldots, \tilde{w}_{gl}^{k+t-1})$ is the discrete Fourier transform (DFT) of $(w_{gl}^0, \ldots, w_{gl}^{k+t-1})$. Hence, the encoder can utilize the fast algorithms developed for the DFT implementation to compute $\tilde{\boldsymbol{W}}_j$'s and then computes the shares sent to the nodes according to (13), see, e.g., [46].

The decoder's task is to interpolate the polynomial $f(u(z))$ followed by evaluating it over α_i's, for $i \in [N]$. Let the polynomial $f(u(z))$ be expressed as

$$f(u(z)) = \sum_{i=0}^{\tilde{D}} \boldsymbol{V}_i z^i, \tag{19}$$

with $\boldsymbol{V}_i \in \mathbb{R}^{u \times h}$. Let $A = \{i_1, \ldots, i_{\tilde{D}+1}\}$ denote the indices of *non-straggler* users, i.e., users that have returned their computation results to the master node. The interpolation step at the decoder is equivalent to inverting the following matrix:

$$\mathbf{B}_{(\tilde{D}+1)\times(\tilde{D}+1)} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & \gamma^{i_1} & \gamma^{2i_1} & \ldots & \gamma^{\tilde{D}i_1} \\ 1 & \gamma^{i_2} & \gamma^{2i_2} & \ldots & \gamma^{\tilde{D}i_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \gamma^{i_{\tilde{D}+1}} & \gamma^{2i_{\tilde{D}+1}} & \ldots & \gamma^{\tilde{D}i_{\tilde{D}+1}} \end{bmatrix}. \tag{20}$$

***Remark** 2:* In general, in a system of linear equations $\boldsymbol{Ax} = \boldsymbol{y}$, where $\boldsymbol{x}$ is a vector of unknown variables and $\boldsymbol{A}$ is referred to as *coefficient matrix*, the perturbation in the solution caused by the perturbation in $\boldsymbol{y}$ is characterized as follows. Let $\hat{\boldsymbol{y}}$ denote a noisy version of $\boldsymbol{y}$, where the noise can be caused by round-off errors, truncation, etc. Let also $\hat{\boldsymbol{x}}$ denote the solution to the considered linear system when $\boldsymbol{y}$ is replaced by $\hat{\boldsymbol{y}}$. Let $\Delta\boldsymbol{x} \stackrel{\text{def}}{=} \hat{\boldsymbol{x}} - \boldsymbol{x}$ and $\Delta\boldsymbol{y} \stackrel{\text{def}}{=} \hat{\boldsymbol{y}} - \boldsymbol{y}$ denote the perturbation in $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Then the relative perturbations of $\boldsymbol{x}$ is bounded in terms of that of $\boldsymbol{y}$ as follows [47]:

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \kappa_{\boldsymbol{A}} \frac{\|\Delta\boldsymbol{y}\|}{\|\boldsymbol{y}\|}, \tag{21}$$

where $\kappa_{\boldsymbol{A}}$ is the condition number of $\boldsymbol{A}$ and $\|.\|$ denotes the l^2-norm.

For $(g,l) \in [u] \times [h]$, let v_{gl}^i denote the (g,l) element of $\boldsymbol{V}_i$ for $i = 0, 1, \ldots, \tilde{D}$, where $\boldsymbol{V}_i$ is specified in (19), and f_{gl}^j denote the (g,l) element of $f(\boldsymbol{X}_j)$ for $j \in [k]$. Let also

$$\boldsymbol{v}_{gl} \stackrel{\text{def}}{=} (v_{gl}^0, \ldots, v_{gl}^{\tilde{D}}),$$

for $g \in [u]$ and $l \in [h]$. For ease of notation, let

$$\overline{\beta} = \frac{\beta^{\tilde{D}+2} - 1}{\beta^2 - 1}. \tag{22}$$

The following lemma establishes a relation between the error in the entries of the outcome of the scheme, i.e., f_{gl}^j, and the entries of $\boldsymbol{V}_i$, i.e., v_{gl}^i. Note that in this analysis the error due to representing α_j's and β_j's using floating-point numbers is discarded as it is dominated in practice by the error imposed by the precision loss in the elements of $\boldsymbol{V}_i$'s, specified in (19).

Lemma 3: For all $g \in [u]$, $l \in [h]$, and $j \in [k]$ we have

$$\Delta f_{gl}^j \leqslant \overline{\beta}\, \|\boldsymbol{v}_{gl}\|\, \kappa_{\boldsymbol{B}} 2^{-b_m}, \tag{23}$$

where $\overline{\beta}$ is defined in (22), $\boldsymbol{B}$ is defined in (20), and b_m is the number of precision bits in the floating-point representation, specified in Remark 1.

Proof: Let $\tilde{f}_{gl}^i \stackrel{\text{def}}{=} f_{gl}(u(\alpha_i))$ for all $i \in [N]$, $g \in [u]$ and $l \in [h]$, and $\tilde{\boldsymbol{f}}_{gl} \stackrel{\text{def}}{=} (\tilde{f}_{gl}^{i_1}, \ldots, \tilde{f}_{gl}^{i_{\tilde{D}+1}})$, where $i_1, \ldots, i_{\tilde{D}+1}$ represent the indices of worker nodes that returned the computation results. Note that the evaluations of (19) over α_i's, for $i \in A$, can be regarded as $u \times h$ systems of linear equations all with $\boldsymbol{B}$ as the underlying coefficient matrix, i.e.,

$$\tilde{\boldsymbol{f}}_{gl} = \boldsymbol{B}\boldsymbol{v}_{gl}, \tag{24}$$

for all $g \in [u]$ and $l \in [h]$. By utilizing the statement in Remark 2 one can write

$$\frac{\|\Delta\boldsymbol{v}_{gl}\|}{\|\boldsymbol{v}_{gl}\|} \leqslant \kappa_{\boldsymbol{B}} \frac{\left\|\Delta\tilde{\boldsymbol{f}_{gl}}\right\|}{\left\|\tilde{\boldsymbol{f}_{gl}}\right\|}. \tag{25}$$

Note that the precision error in the considered floating-point numbers is bounded by 2^{-b_m} since it is assumed that no other error is imposed on the computation results in the worker nodes. Hence, one can write

$$\frac{\left\|\Delta\tilde{\boldsymbol{f}_{gl}}\right\|}{\left\|\tilde{\boldsymbol{f}_{gl}}\right\|} \leqslant 2^{-b_m}. \tag{26}$$

Combining (25) with (26) results in

$$\frac{\|\Delta\boldsymbol{v}_{gl}\|}{\|\boldsymbol{v}_{gl}\|} \leqslant \kappa_{\boldsymbol{B}} 2^{-b_m}, \tag{27}$$

for all $g \in [u]$ and $l \in [h]$. Moreover, note that

$$f(\boldsymbol{X}_j) = f(u(\beta_j)) = \sum_{i=0}^{\tilde{D}} \boldsymbol{V}_i \beta_j^i.$$

Let $\boldsymbol{\beta}_j$ denote $(1, \beta_j, \beta_j^2, \ldots, \beta_j^{\tilde{D}})$, for $j \in [k]$. Then one can write

$$f_{gl}^j = \boldsymbol{\beta_j} \cdot \boldsymbol{v}_{gl}, \tag{28}$$

which implies that

$$\Delta f_{gl}^j \leqslant \left\|\boldsymbol{\beta}_j\right\| \left\|\Delta\boldsymbol{v}_{gl}\right\|, \tag{29}$$

where $\cdot$ denotes the inner product operation. Note that for all $j \in [k]$,

$$\left\|\boldsymbol{\beta}_j\right\|^2 \leqslant \sum_{i=0}^{\tilde{D}} \beta^{2i} = \frac{\beta^{\tilde{D}+2} - 1}{\beta^2 - 1} = \overline{\beta}. \tag{30}$$

Combining (27), (29) and (30) yields

$$\Delta f_{gl}^j \leqslant \overline{\beta}\, \|\boldsymbol{v}_{gl}\|\, \kappa_{\boldsymbol{B}} 2^{-b_m}, \tag{31}$$

which completes the proof. ■

Let c_{ij} denote the maximum absolute value of the coefficients of $f_{ij}(\cdot)$ for all $i \in [u]$ and $j \in [h]$, $c \stackrel{\text{def}}{=} \max_{i,j} c_{ij}$, and $\lambda_{\min}$ denote the minimum singular value of $\boldsymbol{B}$, defined in (20). In the next theorem, an upper bound on the absolute error in the outcome of the protocol is provided for the general class of polynomials over matrices, defined in (1).

Theorem 4: The absolute error on the entries of $f(\boldsymbol{X}_j)$, for $j \in [k]$, in the outcome of ALCC is bounded as follows:

$$\Delta f_{gl}^j \leqslant \overline{\beta} \frac{c(mne)^D}{\lambda_{\min}} \sqrt{\tilde{D}+1} (kr + t\theta\sigma_n)^D \kappa_{\boldsymbol{B}} 2^{-b_m} (1 + O(\frac{1}{\sigma_n})), \tag{32}$$

where $\overline{\beta}$ is defined in (22), $\boldsymbol{B}$ is defined in (20), and b_m is the number of precision bits in the floating-point representation,

specified in Remark 1.

Proof: Note that (24) implies

$$\|\boldsymbol{v}_{gl}\| \leqslant \frac{\left\|\tilde{\boldsymbol{f}}_{gl}\right\|}{\lambda_{\min}}, \tag{33}$$

for any arbitrary set of non-straggler indices A. Also, Lemma 2 implies that the j-th entry of the DFT of $(w_{gl}^0, \ldots, w_{gl}^{k+t-1})$ is equal to $\tilde{w}_{gl}^j$. Hence, we have

$$\tilde{w}_{gl}^j \leqslant kr + t\theta\sigma_n, \tag{34}$$

where we used the fact that the absolute value of entries of $\boldsymbol{X}_j$'s and $\boldsymbol{N}_j$'s are less than r and $\theta\frac{\sigma_n}{\sqrt{t}}$, respectively, as discussed in Section II. Moreover, for $i \in [N]$ one can write

$$\tilde{f}_{gl}^i \leqslant c(mne)^D(kr + t\theta\sigma_n)^D(1 + O(\frac{1}{\sigma_n})), \tag{35}$$

which holds by noting that the number of total monomials of degree D in mn variables is equal to $\binom{mn+D-1}{D-1}$ and we have

$$\binom{mn + D - 1}{D - 1} \leqslant (\frac{e(mn + D - 1)}{D})^D \leqslant (emn)^D,$$

where e is the natural number. Then (35) implies that

$$\left\|\tilde{\boldsymbol{f}}_{gl}\right\| \leqslant c(mne)^D\sqrt{\tilde{D} + 1}(kr + t\theta\sigma_n)^D(1 + O(\frac{1}{\sigma_n})), \tag{36}$$

since $\tilde{\boldsymbol{f}}_{gl}$ has $\tilde{D} + 1$ components. Substituting (36) into (33) together with the result of Lemma 3 complete the proof. ∎

Theorem 4 provides an upper bound on the accuracy of the outcome of ALCC with floating-point implementation for a general polynomial function $f(\cdot)$. However, the polynomial $f(\cdot)$ often has a certain structure in practice that can be leveraged to strengthen the result of Theorem 4. More specifically, we say that $f(\cdot)$ is a *matrix polynomial function*, or simply a *matrix polynomial*, if it can be expressed by matrix addition, multiplication, and transposition as well as addition and multiplication by a constant matrix/vector/scalar. For instance, $f(\boldsymbol{X}) = \boldsymbol{a}\boldsymbol{X}\boldsymbol{X}^t$, for some vector $\boldsymbol{a}$, is such a matrix polynomial function. The difference between a general polynomial, defined in (1), and a matrix polynomial is illustrated in the following example.

Example 3.1: Let $\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$. Then the function $g_1(\boldsymbol{X}) \stackrel{\text{def}}{=} \begin{bmatrix} x_{11}^2 + x_{11}x_{12} + x_3^2 & x_1 \\ x_{21}^2 + x_{11}x_{22} & x_{22}^2 \end{bmatrix}$ is a polynomial function of degree 2, as defined in (1), but is not a matrix polynomial function. The function $g_2(\boldsymbol{X}) \stackrel{\text{def}}{=} \begin{bmatrix} x_{11}^2 + x_{12}^2 & x_{11}x_{21} + x_{12}x_{22} \\ x_{21}x_{11} + x_{22}x_{12} & x_{21}^2 + x_{22}^2 \end{bmatrix} = \boldsymbol{X}\boldsymbol{X}^t$ is a matrix polynomial function of degree 2. Note also that the determinant of a matrix, i.e., $g_3(\boldsymbol{X}) \stackrel{\text{def}}{=} det(\boldsymbol{X}) = x_{11}x_{22} - x_{12}x_{21}$ is a polynomial but not a matrix polynomial. The matrix inversion function, i.e., $g_4(\boldsymbol{X}) \stackrel{\text{def}}{=} \boldsymbol{X}^{-1}$ is not even a polynomial function.

The following corollary provides a stronger accuracy bound on the outcome of ALCC with matrix polynomial as its underlying function.

Corollary 5: Let $f(\cdot)$ be a matrix polynomial function. Then, the absolute error on the entries of $f(\boldsymbol{X}_j)$, for $j \in [k]$, in the outcome of ALCC is bounded as follows:

$$\Delta f_{gl}^j \leqslant C(kr + t\theta\sigma_n)^D\kappa_{\boldsymbol{B}}2^{-b_m}(1 + O(\frac{1}{\sigma_n})), \tag{37}$$

where $C \stackrel{\text{def}}{=} \overline{\beta}\frac{c\max(m,n)^D}{\lambda_{\min}}\sqrt{\tilde{D} + 1}$.

Proof: Note that the number of D-degree monomials in a matrix polynomial is at most $\max(m, n)^D$. The remaining of the proof is similar to that of Theorem 4. ∎

Remark 3: Note that when no stragglers are assumed, i.e., $s = 0$, picking α_i's in the proposed ALCC protocol according to (6) implies that the matrix $\boldsymbol{B}$, defined in (20), is a unitary matrix. Hence, we have $\kappa_{\boldsymbol{B}} = 1$ which is the minimum possible for the condition number $\kappa_{\boldsymbol{B}}$. For the case of ALCC with stragglers, i.e., $s > 0$, one can utilize the following upper bond on $\kappa_{\boldsymbol{B}}$ [40, Theorem 1]:

$$\kappa_{\boldsymbol{B}} \leqslant O(\tilde{N}^{s+6}), \tag{38}$$

where $\tilde{N}$ is the smallest odd number larger than N. Combining (38) with (32) leads to an upper bound on the accuracy of ALCC scheme with s stragglers.

B. Comparisons with LCC and numerical results

In this section we compare the accuracy of ALCC with that of LCC that employs finite field operations. In LCC the computations are preformed over a finite field of a prime size p, denoted by $\mathbb{F}_p$, and are implemented using fixed-point numbers [4]. In order to have a fair comparison, we assume that the number of bits that are used to represent data symbols in ALCC, that uses floating point, and LCC, that uses fixed point, are equal. Let that number be denoted by b. It is also assumed that b is fixed throughout the implementation of the scheme. It is shown in Section III-A how the accuracy of ALCC depends on b. Same as in ALCC, the accuracy of LCC also depends on b as discussed next.

In LCC the real-valued data are assumed to be first quantized and then mapped to elements of $\mathbb{F}_p$. Also, note that if a symbol computed during the process by one of the workers in LCC becomes larger than p, an incident referred to as an overflow error, then a successful recovery of the outcome of the computation can not be guaranteed. Let Δ denote the corresponding quantization step. Let also s_{ij} denote the sum of the absolute values of the coefficients of the polynomial f_{ij}, for $i \in [u]$ and $j \in [h]$, and let $s_a \stackrel{\text{def}}{=} \max_{i,j} s_{ij}$. Then, in order to avoid overflow errors, it is required that

$$\frac{s_a}{\Delta}(\frac{r}{\Delta})^D \leqslant \frac{p}{2}, \tag{39}$$

since the left hand-side of (39) corresponds to the maximum value of the polynomial $f(\cdot)$ when evaluated over the quantized data. The latter is by noting that $f_{ij}(x) \leqslant s_a x^D \leqslant s_a r^D$ for all i, j, the quantization step is Δ, and the magnitude of the entries of $\boldsymbol{X}$ is upper bounded by r.

The next step is to characterize how large p can be given the fixed number of representation bits b. To this end, two different scenarios can be considered regarding how the intermediate multiplications, at the worker nodes, are carried out. More specifically, the intermediate multiplications may or may not

be done modulo p. We consider the two cases separately and provide bounds on the accuracy in both cases. Performing intermediate multiplications modulo p leads, in general, to a better accuracy, as will be also shown in the remaining of this section. However, this improvement comes at the cost of increased *latency* of the fixed-point implementation. This is because, in practice, performing multiplications over large finite fields require further processing, compared to the regular multiplication, and are slower than the regular multiplications.

In the first case, it is assume that the intermediate multiplications are done modulo p. Then in order to avoid overflow errors in multiplications while employing fixed-point implementation, it is necessary to have

$$p^2 \leqslant 2^b. \tag{40}$$

In the second case, it is assumed that the underlying multiplications are done over $\mathbb{Z}$ and the worker nodes need to compute the result modulo p only once after the polynomial evaluations are completed. In this case, the condition in (40) is modified as follows:

$$\frac{s_a}{\Delta} p^D \leqslant 2^b. \tag{41}$$

Combining (39) with (40), for the first case, and with (41), for the second case, provides lower bounds on the absolute error of the outcomes of LCC. In particular, one must have

$$\left(\frac{s_a r^D}{2^{(\frac{b}{2}-1)}}\right)^{\frac{1}{D+1}} \leqslant \Delta, \tag{42}$$

for the first case, and

$$\left(\frac{s_a^{(1+\frac{1}{D})} r^D}{2^{(\frac{b}{D}-1)}}\right)^{(\frac{D}{D^2+D+1})} \leqslant \Delta, \tag{43}$$

for the second case.

Now, consider ALCC with floating-point implementation where each symbol is represented by b bits. In current standard systems, 8 bits are allocated to represent the exponent. Also, one bit is reserved for indication of zero and one bit is reserved for the sign flag. Hence, the total number of precision bits b_m is equal to $b - 10$. We use these parameters to plot the bounds on the accuracy of ALCC versus that of LCC. In Figure 2, the upper bound on the absolute error in ALCC with floating-point implementation, provided in Corollary 5, is plotted and is compared with the lower bounds on the absolute error in LCC with fixed-point implementation, provided in (42) and (43), for the two aforementioned cases. Note that for the experiments with the results shown in Figure 2 the terms $O(\frac{1}{\sigma_n})$, that are used in the bounds provided in Theorem 4 and Corollary 5, are equal to zero due to the certain matrix polynomial function considered. In general, such terms can be often discarded in ALCC with general underlying polynomial functions as σ_n considered in practice is relatively large due to privacy concerns, e.g., $\sigma_n = 10^{12}$ in the considered experiments. Note that these bounds are plotted as a function of b, i.e., the total number of bits reserved to represent a data symbol in both the fixed-point and the floating-point implementation while the other parameters of the system are fixed. It can be observed that for b larger than a certain threshold, the upper bound derived on the error in ALCC is smaller than both of the lower bounds

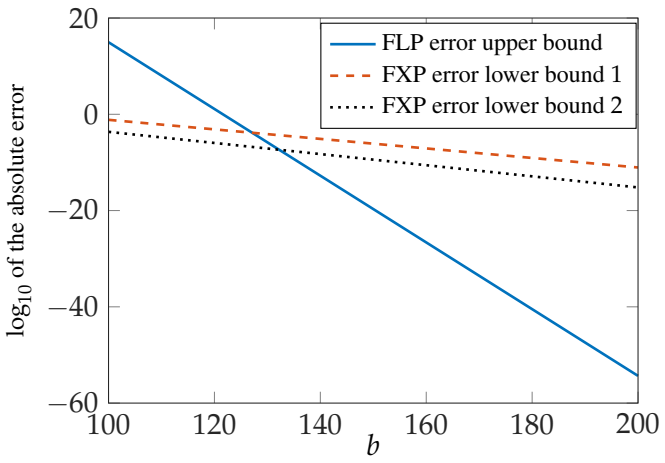

Fig. 2: Comparison of upper bound on the absolute error in floating-point (FLP) implementation with lower bounds on the error in fixed-point (FXP) implementation employing conventional (FXP 1) and (FXP 2). The underlying function considered is $f(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{X}^t$. The parameters are as follows: $k = 5, t = 3, s = 0, m = n = 1000, r = 100, \theta = 3, \sigma_n = 10^{12}$.

on the error in LCC. Since these bounds may not be tight, the actual threshold would perhaps be lower than what is shown in Figure 2.

One can also analyze the aforementioned bounds in terms of the decay rate of the absolute error as b increases. More specifically, the lower bounds on the absolute error in LCC, derived in (41) and (42), decay exponentially in b with an exponent between $\frac{1}{2(D+1)}$ and $\frac{1}{D^2+D+1}$. However, the upper bound on the absolute error in ALCC, derived in (32), decays exponentially in b with exponent 1. This significant improvement in accuracy comes at the expense of deviating from the perfect privacy, in an information-theoretic sense, in ALCC compared to LCC. This will be discussed in details in the next section. In particular, it will be shown that this deviation is negligible for practical purposes.

IV. Privacy Analysis

In this section, we analyze the privacy level of data in ALCC by considering two notions of security, namely, the mutual information security (MIS) and the distinguishing security (DS) over the continuous probability space.

A. Privacy analysis with Gaussian noise

We first characterize the privacy of ALCC in terms of the MIS metric by utilizing existing results on the capacity of multiple-input-multiple-output (MIMO) channel. Furthermore, by using the relation between the MIS and the DS security metrics in the analog domain the privacy of ALCC is also characterized in terms of the DS metric. Such a relation is observed in the context of wiretap channels in [48] and has been also utilized in [16].

Consider the ALCC protocol described in Section II. For $j \in [k]$ and $i \in [t]$, let X_j and N_i denote the (g, l) element of the matrices $\boldsymbol{X}_j$ and $\boldsymbol{N}_i$, respectively, for some fixed $g \in [m]$ and $l \in [n]$. For the sake of clarity, g and l are fixed throughout this

section. However, the analysis does not depend on the specific choice of g and l.

The Lagrange polynomial introduced in (2) is written for the fixed considered indices as follows:

$$U(z) = \sum_{j=1}^{k} X_j l_j(z) + \sum_{j=k+1}^{k+t} N_{j-k} l_j(z), \tag{44}$$

where data symbols X_j's are random variables with arbitrary distribution and with the range $\mathbb{D}_x \stackrel{\text{def}}{=} [-r, r]$, for $j \in [k]$, and $N_i \sim \mathcal{N}(0, \frac{\sigma_n^2}{t})$, for $i \in [t]$. Let Y_i denote the corresponding entry of $\boldsymbol{Y}_i$, for $i \in [N]$. In other words, $Y_i = U(\alpha_i)$. Let $T = \{i_1, \cdots, i_t\}$ denote the set of indices for the colluding parties. Let also X, N, and Y_T denote $(X_1, \cdots, X_k)^{\mathrm{T}}$, $(N_1, \cdots, N_t)^{\mathrm{T}}$, and $(Y_{i_1}, \cdots, Y_{i_t})^{\mathrm{T}}$, respectively, where $(\cdot)^{\mathrm{T}}$ is the transpose operation. By convention, a random variable/vector is denoted by a capital letter and its instance is denoted by the corresponding lower case letter.

The following equation relates the encoded symbols received by the colluding set of parties T to the dataset symbols and the added noise symbols:

$$Y_T = \boldsymbol{L}_T X + \tilde{\boldsymbol{L}}_T N, \tag{45}$$

where

$$\boldsymbol{L}_T \stackrel{\text{def}}{=} \begin{bmatrix} l_1(\alpha_{i_1}) & \dots & l_k(\alpha_{i_1}) \\ l_1(\alpha_{i_2}) & \dots & l_k(\alpha_{i_2}) \\ \vdots & \vdots & \vdots \\ l_1(\alpha_{i_t}) & \dots & l_k(\alpha_{i_t}) \end{bmatrix}_{t \times k}, \tag{46}$$

and

$$\tilde{\boldsymbol{L}}_T \stackrel{\text{def}}{=} \begin{bmatrix} l_{k+1}(\alpha_{i_1}) & \dots & l_{k+1}(\alpha_{i_1}) \\ l_{k+2}(\alpha_{i_2}) & \dots & l_{k+2}(\alpha_{i_2}) \\ \vdots & \vdots & \vdots \\ l_{k+t}(\alpha_{i_t}) & \dots & l_{k+t}(\alpha_{i_t}) \end{bmatrix}_{t \times t}. \tag{47}$$

The amount of information revealed to the set of colluding parties can be measured in terms of the MIS metric, denoted by η_c, defined as follows:

$$\eta_c \stackrel{\text{def}}{=} \max_{T} \max_{P_X : |X_j| < r, \forall j \in [k]} I(Y_T; X), \tag{48}$$

where P_X is the probability density function (PDF) of X and the maximization is taken over all $T \subset [N]$ with $|T| = t$. Since $|X_j| \leqslant r$, we have $E[X_j]^2 \leqslant r^2$. Then, one can write

$$\eta_c \leqslant \max_{T} \max_{P_X : E[X_j^2] \leqslant r^2} I(X; Y_T). \tag{49}$$

Next, we characterize the right hand side of (49) in terms of other parameters of the system. To this end, the capacity results of MIMO channels are utilized as discussed next. Consider a MIMO channel with k transmit and t receive antennas and the input-output relation

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{50}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the $k \times 1$ transmitted signal and the $t \times 1$ received signal vectors, respectively, $\boldsymbol{H}_{t \times k}$ represent the channel gain matrix known to both the transmitter and the receiver, and $\boldsymbol{n}_{t \times 1}$ is an additive zero-mean Gaussian noise vector. Let $\boldsymbol{N}_c$ denote the noise correlation matrix, i.e., the covariance matrix of the vector $\boldsymbol{n}$. By using the results on the capacity of MIMO channel with equal-power allocation constraint and correlated noise, one can get an upper bound on the right-hand side of (49). The capacity of this MIMO channel, under equal-power allocation constraint, is well-known and is expressed as follows [49, IV-A]:

$$C = \log_2 |\boldsymbol{I}_t + P \boldsymbol{N}_c^{-1} \boldsymbol{H}\boldsymbol{H}^H|, \tag{51}$$

where P is the maximum transmission power of each antenna at the transmitter side, $\boldsymbol{I}_t$ is the $t \times t$ identity matrix and $|\cdot|$ denotes matrix determinant.

Theorem 6: In the proposed ALCC, the MIS metric η_c, defined in (48), is upper bounded as follows:

$$\eta_c \leqslant \max_{T} \log_2 |\boldsymbol{I}_t + \frac{r^2 t}{\sigma_n^2} \tilde{\boldsymbol{\Sigma}}_T^{-1} \boldsymbol{\Sigma}_T|, \tag{52}$$

where $\tilde{\boldsymbol{\Sigma}}_T \stackrel{\text{def}}{=} \tilde{\boldsymbol{L}}_T \tilde{\boldsymbol{L}}_T^H$ and $\Sigma_T \stackrel{\text{def}}{=} \boldsymbol{L}_T \boldsymbol{L}_T^H$. Also, $\tilde{\boldsymbol{\Sigma}}_T$ and $\tilde{\boldsymbol{L}}_T$ are specified in (46) and (47), respectively.

Proof: Note that in (45) the term $\tilde{\boldsymbol{L}}_T N$ can be regarded as the noise vector and, consequently, (45) can be turned into an equation similar to (50) describing the MIMO channel model. Hence, by using this observation together with (51) and the definition of capacity, (49) leads to (52). ■

Corollary 7: For $r = o(\sigma_n)$, we have

$$\eta_c \leqslant \frac{1}{\ln(2)} \max_{T} \mathrm{tr}(\tilde{\boldsymbol{\Sigma}}_T^{-1} \boldsymbol{\Sigma}_T) \frac{r^2 t}{\sigma_n^2} + o(\frac{r^2}{\sigma_n^2}). \tag{53}$$

Proof: The proof is by utilizing $|\boldsymbol{I}_t + \epsilon \boldsymbol{A}| = 1 + \epsilon \mathrm{tr}(\boldsymbol{A}) + o(\epsilon)$ together with $\log_2(1+\epsilon) = \frac{\epsilon}{\ln(2)} + o(\epsilon)$ in the upper bound presented in Theorem 6. ■

Next, we characterize the privacy of ALCC in terms of the DS metric. The DS metric is defined using the notion of the *total variation* (TV) distance $D_{TV}(.,.)$. In general, for any two probability measures P_1 and P_2 on a σ-algebra $\mathcal{F}$, the TV distance is defined as $D_{TV}(P_1, P_2) \stackrel{\text{def}}{=} \sup_{B \in \mathcal{F}} |P_1(B) - P_2(B)|$. While DS metric is often defined for discrete random variables in the cryptography literature, it can be also extended to the case of continuous random variables [16]. In particular, in the proposed ALCC protocol η_s is defined as as follows:

$$\eta_s \stackrel{\text{def}}{=} \max_{T} \max_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{D}_X} D_{\mathrm{TV}}(P_{Y_T | X = \boldsymbol{x}_1}, P_{Y_T | X = \boldsymbol{x}_2}), \tag{54}$$

where $\mathbb{D}_X = [-r, r]^k$ is the support of X. Note that, roughly speaking, a smaller value for η_s implies data is kept more private against any set of t colluding parties.

Next, we discuss the privacy guarantee for ALCC in terms of the DS metric η_s. This is done by utilizing relations between η_c and η_s and the upper bound on η_c derived in Theorem 6.

Relations between MIS and DS metrics was first established in [50] though for discrete random variables. In particular, it is shown in [50] that:

$$\eta_s \leqslant \sqrt{2\eta_c}, \tag{55}$$

assuming all underlying random variables are discrete. This result is also extended to the analog domain in [48]. In other words, it is shown that (55) also holds when the underlying random variables are continuous. Then, combining (52) with

(55) yields the following upper bound on the DS metric η_s:

$$\eta_s \leqslant \sqrt{2 \max_T \log_2 |\boldsymbol{I}_t + \frac{r^2 t}{\sigma_n^2} \tilde{\boldsymbol{\Sigma}}_T^{-1} \boldsymbol{\Sigma}_T|}. \tag{56}$$

In particular, for $r = o(\sigma_n)$, we have

$$\eta_s \leqslant \sqrt{\frac{2t}{\ln(2)} \max_T \mathrm{tr}(\tilde{\boldsymbol{\Sigma}}_T^{-1} \boldsymbol{\Sigma}_T)} \frac{r}{\sigma_n} + o(\frac{r}{\sigma_n}). \tag{57}$$

Remark 4: Note that both (52) and (56) imply that increasing the standard deviation of the added noise, i.e., σ_n, while other parameters of ALCC are fixed, improves bounds on the privacy level of ALCC. However, this improvement comes at the expense of degrading the accuracy of the outcome of ALCC, according to Theorem 4. This exhibits a fundamental trade-off between the accuracy and privacy of ALCC. Such a trade-off between accuracy and privacy in the analog domain has been observed for the first time in [16] for a privacy-preserving distributed computing setup.

Next, the provided upper bounds on the maximum amount of information revealed about the dataset to a subset T of colluding parties with size t are numerically evaluated. This is done for both the MIS security metric, bounded in (52), as well as the DS metric, bound in (56), for a certain set of parameters and the results are shown in Figure 3. Both η_s and η_c are plotted versus β. It can be observed that both the bounds are decreased by increasing β. In other words, this indicates that increasing β, in general, leads to enhancement in the privacy of the ALCC protocol. However, the provided upper bound on the accuracy of the outcome of ALCC, provided in (32) and (37), implies that the precision loss would also grow by increasing β. The upper bound on the absolute error in ALCC with general underlying matrix polynomial function is plotted versus β in Figure 4 for a certain set of parameters. Note that, same as in the experiments with results demonstrated in Figure 2, the terms $o(\frac{1}{\sigma_n})$ are discarded in the plot in Figure 4 as well since $\frac{1}{\sigma_n} = 10^{-23}$ is negligible. This together with the plot in Figure 3 demonstrates a new fundamental trade-off between the accuracy and the privacy of the ALCC protocol which is specific to ALCC and is controlled by the choice of β. It can be also observed from Figure 3 and Figure 4 that a reasonable value for β, e.g., $\beta = 1.5$, can be picked for which the upper bounds on η_s and η_c are reasonably low, e.g., $\sim 10^{-10}$ and $\sim 10^{-20}$, respectively, while the upper bound on the error in the outcome is reasonable for practical purposes, e.g., $\sim 10^{-3}$.

B. Privacy analysis with truncated noise

The results presented in Section IV-A are derived assuming that the entries of the noise matrices $\boldsymbol{N}_i$'s in (2) are drawn from a complex circular-symmetric Gaussian distribution. However, as discussed in Section III, these noise terms should be bounded for practical implementations. In other words, the actual PDF of the noise terms is a properly scaled version of the Gaussian PDF truncated between $-\theta \frac{\sigma_n}{\sqrt{t}}$ and $\theta \frac{\sigma_n}{\sqrt{t}}$, for some $\theta \in \mathbb{R}$. Roughly speaking, we say that the noise terms are truncated. In this section, we extend the results on bounding η_s to the case with the noise terms being truncated. We use the upper bound on the DS metric in a similar setup with truncated complex Gaussian noise [16] that involves the following quantity:

$$d_{\text{mean}} \stackrel{\text{def}}{=} \max_T \max_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{D}_X} |\boldsymbol{L}_T(\boldsymbol{x}_1 - \boldsymbol{x}_2)|. \tag{58}$$

Note that conditioned on $X = \boldsymbol{x}$ in (45), Y_T is a complex Gaussian vector with mean $\boldsymbol{L}_T \boldsymbol{x}$. Then the parameter d_{mean}, defined in (58), is the maximum Euclidean distance between the means of any two such conditional random vectors in the t-dimensional complex vector space, where the maximum is taken over the set of all colluding sets T with size t and all $\boldsymbol{x}_1, \boldsymbol{x}_2$ in the range of the random vector X. In the following lemma an upper bound on d_{mean} is obtained by using the alternative characterization of Lagrange monomials derived in Lemma 1.

Lemma 8: The parameter d_{mean}, defined in (58), is upper bounded as follows:

$$d_{\text{mean}} \leqslant \frac{kr}{k+t} \frac{(\frac{1}{\beta})^{k+t} - 1}{(\frac{1}{\beta}) - 1}. \tag{59}$$

Proof: For all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{D}_X$ and $T \subset [N]$ with $|T| = t$, we have

$$\begin{aligned}
|\boldsymbol{L}_T(\boldsymbol{x}_1 - \boldsymbol{x}_2)| &\leqslant |\boldsymbol{L}_T \boldsymbol{x}_1| + |\boldsymbol{L}_T \boldsymbol{x}_1| && (60)\\
&\leqslant 2 \max_{\boldsymbol{x} \in \mathbb{D}_X} |\boldsymbol{L}_T \boldsymbol{x}| && (61)\\
&\leqslant 2\sqrt{t} \max_{\substack{\alpha_i: \\ i \in [N]}} \max_{\substack{x_1, \cdots, x_k: \\ |x_i| \leqslant r}} \sum_{j=1}^{k} x_j l_j(\alpha_i) && (62)\\
&\leqslant kr \max_{\substack{\alpha_i: \\ i \in [N]}} |l_j(\alpha_i)| && (63)\\
&\leqslant \frac{kr}{k+t} \frac{(\frac{1}{\beta})^{k+t} - 1}{(\frac{1}{\beta}) - 1}, && (64)
\end{aligned}$$

where (62) is by the definition of $\boldsymbol{L}_T$ in (46) and noting that $\boldsymbol{L}_T \boldsymbol{x}$ is a t-dimensional vector, (63) holds by noting that $|x_i| < r$ and the summation has k terms, and (64) is by $|x_i| < r$, upper bounding $|l_j(\alpha_i)|$ by (8) and noting that $|\alpha_i| = 1$ for all i. ∎

Let η_s' denote the DS metric for the case where the noise terms in (2) are truncated. The following theorem provides an upper bound on η_s' in terms of η_s, the upper bound on d_{mean}, and other parameters of the ALCC protocol.

Theorem 9: The DS metric, defined in (54), for the case where the entries of $\boldsymbol{N}_i$'s in (2) are drawn from a truncated complex Gaussian distribution with truncation level $\theta \frac{\sigma_n}{\sqrt{t}}$ satisfies the following inequality:

$$\eta_s' \leqslant \frac{1}{w} \eta_s + \frac{1}{w} (2 \exp(-\frac{1}{2}(\theta - \frac{\overline{d}_{\text{mean}} \sqrt{t}}{\sigma_n})^2))^t,$$

where $w = (1 - 2\exp(-\frac{\theta^2}{2}))^t$ and

$$\overline{d}_{\text{mean}} \stackrel{\text{def}}{=} \frac{kr}{k+t} \frac{(\frac{1}{\beta})^{k+t} - 1}{(\frac{1}{\beta}) - 1}.$$

Proof: The proof follows by combining [16, Theorem 5] and Lemma 8. ∎

A numerical evaluation of the bound provided in Theorem 9 implies that, for instance, having $\theta = 10$ with $t = 10$, together

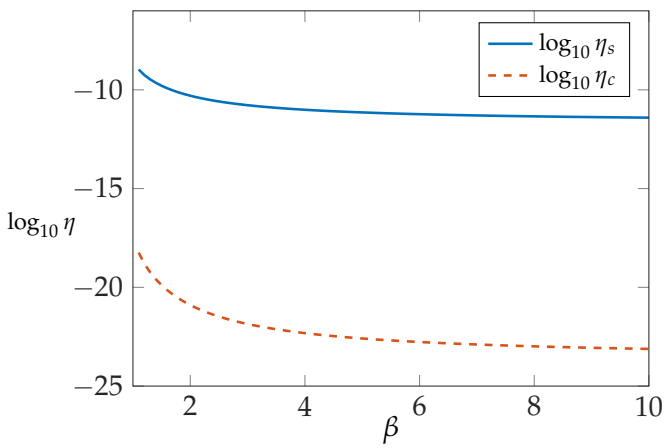

Fig. 3: Upper bounds on η_s and η_c for $N = 15, k = 4, t = 4, \sigma_n = 10^{23}, r = 10^{10}$.

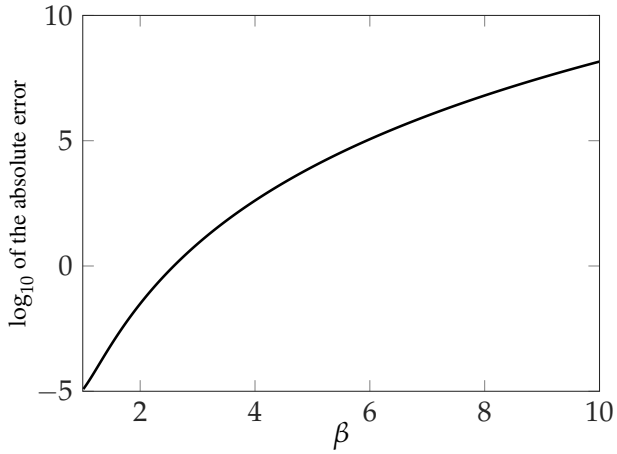

Fig. 4: Upper bound on the accuracy of ALCC versus β for $D = 2, k = 4, t = 4, s = 0, c = 1, m = n = 1000, r = 10^{10}, \theta = 3, \sigma_n = 10^{23}$ and $v = 200$.

with a very small $\frac{r}{\sigma_n}$, which is often the case in practice, we get $\eta_s' \approx \eta_s$. In other words, the privacy of dataset is not compromised by truncating the noise terms as long as θ is large enough, e.g., $\theta = 10$.

V. Experiments

In this section, we demonstrate the performance of ALCC when applied to a certain computational task through experiments. In the first part of this section, it is shown that the precision of ALCC outcome closely follows that of a *centralized* computation, that is when the computations are done directly at a central node without any encoding and decoding. In particular, it is shown that the accuracy of ALCC is *scalable* with dataset size, i.e., the precision of the results remains *almost* the same for a wide range of sizes of the dataset. In the second part, the performance of LCC [4] employing fixed-point representation applied to the same computational task is demonstrated. It is shown that the error in the outcome of LCC experiences a sharp increase due to overflow errors as the dataset size passes a certain threshold.

We consider the task of performing a certain matrix-matrix multiplication. For the sake of clarity, we consider computing $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ where $\boldsymbol{X} \in \mathbb{R}^{m' \times n}$ is a *tall* real-valued matrix, i.e., $m' \gg n$. Such computation is one of the main building blocks in various learning algorithms including training a linear regression model [4], or a logistic regression model [16], [19], etc., where $\boldsymbol{X}$ represents a dataset consisting of m' samples in an n-dimensional feature space. The matrix $\boldsymbol{X}$ can be represented as a batch of matrices $\boldsymbol{X} = (\boldsymbol{X}_1^{\mathrm{T}}, \cdots, \boldsymbol{X}_k^{\mathrm{T}})^{\mathrm{T}}$, where $\boldsymbol{X}_i \in \mathbb{R}^{m \times n}$ with $m' = k \times m$. Then we have

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \sum_{i=1}^{k} \boldsymbol{X}_i^{\mathrm{T}}\boldsymbol{X}_i.$$

Hence, the task of computing $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is reduced to evaluating a degree-2 polynomial over a batch of matrices, consisting of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$, for which ALCC can be utilized to provide speed up by leveraging the computational power of distributed nodes in parallel.

Let $\boldsymbol{Y}$ denote the result of computing $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ in a centralized fashion employing floating-point operations. Let also $\boldsymbol{Y}'$ denote the result of a distributed computing protocol, e.g., ALCC. In order to measure the accuracy loss of the outcome in the distributed protocol compared to the centralized one, we consider the following notion of *relative error*:

$$e_{\text{rel}} \stackrel{\text{def}}{=} \frac{\|\boldsymbol{Y}' - \boldsymbol{Y}\|}{\|\boldsymbol{Y}\|}. \tag{65}$$

In a sense, e_{rel} measures how much the outcome precision is proportionally compromised by utilizing a distributed protocol while providing privacy/speed up. The entries of the dataset $\boldsymbol{X}$ in our experiments are drawn independently from a zero-mean Gaussian distribution with variance 1. We use 64 bits for both the fixed-point and the floating-point numbers to implement both the LCC and the ALCC protocols in our experiments, respectively.

The relative error e_{rel}, defined in (65), is computed for the outcome of ALCC in our experiment and is shown in Table I for a range of values for the dataset size, that is represented by m', and the Lagrange monomials parameter β. As discussed in Section IV-A, it is expected that increasing β results in lower precision outcomes which is also shown in our experiment. But note that it also leads to better privacy as shown in Figure 3. Also, no notable dependence between the relative error in the outcome of ALCC and the size of dataset m' is observed in Table I. This implies that ALCC is scalable with the dataset size as far as the relative error is concerned.

Next, the performance of LCC [4] employing fixed-point numbers is compared to that of ALCC from the relative error perspective. In Figure 5, the relative error is plotted for both LCC and ALCC versus the parameter m', that is proportional to the size of the dataset. For LCC, this is plotted for a few different choices for the size of the underlying finite field p, according to the discussion in Section III-B and keeping in mind that the total number of available bits for representation is 64. In particular, the first case discussed in Section III-B is assumed where the worker nodes compute the results module

m' \ β	1.1	1.5	1.8	2
10^4	4.466	3.304	2.316	1.699
2×10^4	4.532	3.307	2.320	1.713
4×10^4	4.584	3.306	2.331	1.723
6×10^4	4.602	3.316	2.326	1.727
8×10^4	4.612	3.313	2.332	1.731
10^5	4.614	3.320	2.334	1.728

TABLE I: Demonstration of $-\log_{10}(e_{\text{rel}})$ in ALCC for multiple dataset sizes and $\beta = 1.1, 1.5, 1.8, 2$. Other parameters are $k = 5, t = 3, s = 0, N = 15, \sigma_n = 10^6, n = 100$ for all schemes.

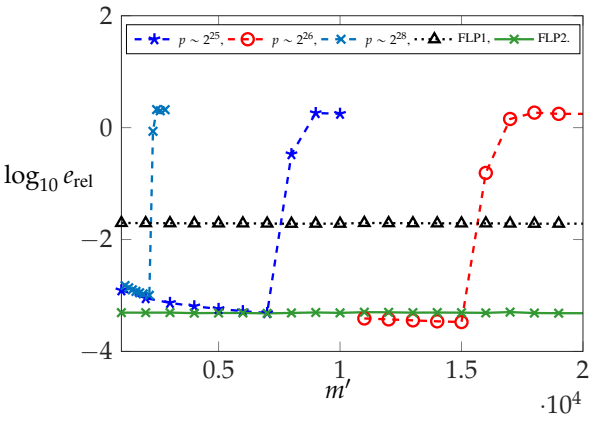

Fig. 5: Comparison of the relative error in the outcome between ALCC and LCC. For LCC, different values of p are considered ($p \sim 2^{25}$, $p \sim 2^{26}$, $p \sim 2^{28}$). For ALCC, $\beta = 2$ and $\beta = 1.5$ are considered for FLP1 and FLP2, respectively. Also, in both protocols we have $k = 5, t = 3, s = 0, N = 15, \sigma_n = 10^6$, and $n = 100$.

p only once after the polynomial evaluations are completed. Also, for ALCC, e_{rel} is plotted for two values of β. It can be observed in Figure 5 that for all the scenarios considered for LCC, there exists a certain threshold for m' after which the computation results become very unreliable due to a very high e_{rel}. As discussed earlier, this significant precision loss is mostly due to overflow errors that are inherent to the fixed-point implementation employed by LCC. As expected, the sharp increase in e_{rel} occurs at a larger value for m' when a larger p is picked. However, the choice of p is limited by the number of bits available for representing fixed-point numbers. Furthermore, the advantage of ALCC compared to LCC is evident in Figure 5 by observing that the relative error in the outcome of ALCC with floating-point implementation is *almost* constant for the considered range of sizes of the dataset. This motivates employing ALCC in certain problems involving very large datasets.

VI. Conclusion

In this paper, the Lagrange coded computing framework is extended to the analog domain in order to efficiently evaluate polynomials over real-valued datasets in a distributed fashion. To this end, the analog Lagrange coded computing (ALCC) protocol is proposed that leverages Lagrange polynomials with a certain set of parameters carefully chosen in the complex plane. The privacy of ALCC is measured in terms of the DS and the MIS security metrics in the analog domain. By utilizing the relations between the DS and the MIS security measures and the existing results on the capacity of MIMO channel with correlated noise, bounds on the privacy level of data in ALCC, amidst possible collusion of workers, is characterized in terms of the aforementioned measures. Moreover, the accuracy of the outcome of ALCC is characterized assuming that the floating-point numbers are employed in the implementation of the protocol. Furthermore, a new trade-off between the accuracy of the outcome and the privacy level of the protocol is characterized that is controlled by the choice of Lagrange polynomial parameters. In our experiments, the ALCC is adopted to perform matrix-matrix multiplication and the outcome is compared to the computation result in a centralized fashion. Finally, the scalability of ALCC and LCC with respect to the dataset size are compared together. It is shown that the accuracy of LCC significantly diminishes after the dataset size passes a certain threshold while the accuracy of ALCC remains almost constant for a wide range of dataset sizes.

There are several directions for future work. Characterizing the accuracy and the privacy level of the ALCC protocol for a general choice of Lagrange monomial parameters β_j's in the complex plane is an interesting direction. In particular, it is not known what choice of β_j's provides the best possible accuracy-privacy trade-off in ALCC and how tight the bounds provided in this paper are with respect to such an optimal scenario. Another direction is to extend ALCC in order to take into account the presence of Byzantine workers, i.e., the worker nodes that deliberately send erroneous computation results [51]–[54]. Providing an efficient and numerically accurate counterpart of Reed-Solomon decoding algorithm in the analog domain would be the main challenge in this direction. Adopting ALLC to provide speed up in performing computational tasks involved in a wide range of applications such as decentralized control, distributed optimization, data mining, etc. [55]–[59] is another future direction. Generalizing ALCC in order to evaluate multiple polynomials in one round by applying techniques utilized in multi-user secret sharing [60] is another approach to be considered for future work.

References

[1] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 439–450.

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

[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[4] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 1215–1225.

[5] S. Li, S. Avestimehr *et al.*, "Coded computing," *Foundations and Trends® in Communications and Information Theory*, vol. 17, no. 1, pp. 1–148, 2020.

[6] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[7] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 20–27.

[8] R. Wright and Z. Yang, "Privacy-preserving bayesian network structure computation on distributed heterogeneous data," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 713–718.

[9] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE transactions on knowledge and data engineering*, vol. 16, no. 9, pp. 1026–1037, 2004.

[10] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM Sigkdd Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[11] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[12] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[13] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2003, pp. 202–210.

[14] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM, 1988, pp. 1–10.

[16] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," *arXiv preprint arXiv:2007.08803*, 2020.

[17] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, "Private machine learning in tensorflow using secure computation," *arXiv preprint arXiv:1810.08130*, 2018.

[18] A. Barak, D. Escudero, A. P. Dalskov, and M. Keller, "Secure evaluation of quantized neural networks." *IACR Cryptology ePrint Archive*, vol. 2019, p. 131, 2019.

[19] J. So, B. Guler, A. S. Avestimehr, and P. Mohassel, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," *arXiv preprint arXiv:1902.00641*, 2019.

[20] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," *arXiv preprint arXiv:1909.07814*, 2019.

[21] J. So, B. Guler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *arXiv preprint arXiv:2002.04156*, 2020.

[22] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the "cubic" barrier," *arXiv preprint arXiv:2001.05101*, 2020.

[23] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.

[24] R. G. D'Oliveira, S. El Rouayheb, and D. Karpuk, "GASP codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, pp. 4038–4050, 2020.

[25] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. E. Rouayheb, and H. Seferoglu, "Private and rateless adaptive coded matrix-vector multiplication," *arXiv preprint arXiv:1909.12611*, 2019.

[26] H. A. Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *arXiv preprint arXiv:1908.04255*, 2019.

[27] Q. Yu and A. S. Avestimehr, "Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 59–72, 2021.

[28] N. Raviv and D. A. Karpuk, "Private polynomial computation from lagrange encoding," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 553–563, 2019.

[29] M. Fahim and V. R. Cadambe, "Lagrange coded computing with sparsity constraints," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 284–289.

[30] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[31] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 351–371.

[32] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 345–364, 2017.

[33] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[34] V. Chen, V. Pastro, and M. Raykova, "Secure computation for machine learning with spdz," *arXiv preprint arXiv:1901.00329*, 2019.

[35] J. So, B. Guler, and S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[36] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish, "Taking proof-based verified computation a few steps closer to practicality," in *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, 2012, pp. 253–268.

[37] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele, "Secure computation on floating point numbers." in *NDSS*, 2013.

[38] O. Catrina, "Towards practical secure computation with floating-point numbers," in *3rd Annual International Conference on Cryptography and Information Security*, 2018.

[39] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 3017–3021.

[40] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *arXiv preprint arXiv:1910.06515*, 2019.

[41] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 3022–3026.

[42] M. V. Jamali, M. Soleymani, and H. Mahdavifar, "Coded distributed computing: Performance limits and code designs," in *2019 IEEE Information Theory Workshop (ITW)*. IEEE, 2019, pp. 1–5.

[43] N. Charalambides, H. Mahdavifar, and A. O. Hero III, "Numerically stable binary gradient coding," *arXiv preprint arXiv:2001.11449*, 2020.

[44] R. M. Roth, "Analog error-correcting codes," *IEEE Transactions on Information Theory*, 2020.

[45] M. Soleymani and H. Mahdavifar, "Analog subspace coding: A new approach to coding for non-coherent wireless networks," *arXiv preprint arXiv:1909.07533*, 2019.

[46] H. J. Nussbaumer, "The fast Fourier transform," in *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, pp. 80–111.

[47] J. W. Demmel, *Applied numerical linear algebra*. Siam, 1997, vol. 56.

[48] C. Ling, L. Luzzi, J.-C. Belfiore, and D. Stehlé, "Semantically secure lattice codes for the gaussian wiretap channel," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 6399–6416, 2014.

[49] L. Schumacher, K. I. Pedersen, and P. E. Mogensen, "From antenna spacings to theoretical capacities-guidelines for simulating MIMO systems," in *The 13th IEEE international symposium on personal, indoor and mobile radio communications*, vol. 2. IEEE, 2002, pp. 587–592.

[50] M. Bellare, S. Tessaro, and A. Vardy, "A cryptographic treatment of the wiretap channel," *Advances in Cryptology – CRYPTO*, 2012.

[51] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[52] R. Guerraoui, F. Huc, and A.-M. Kermarrec, "Highly dynamic distributed computing with Byzantine failures," in *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, 2013, pp. 176–183.

[53] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure multiparty computation*. Cambridge University Press, 2015.

[54] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 192–206.

[55] N. Heydaribeni and A. Anastasopoulos, "Distributed mechanism design for unicast transmission," in *2018 Information Theory and Applications Workshop (ITA)*. IEEE, 2018, pp. 1–6.

[56] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Transactions on knowledge and Data Engineering*, vol. 18, no. 1, pp. 92–106, 2005.

[57] N. Heydaribeni and A. Anastasopoulos, "Distributed mechanism design for network resource allocation problems," *IEEE Transactions on Network Science and Engineering*, 2019.

[58] X. Zhang, M. M. Khalili, and M. Liu, "Improving the privacy and accuracy of ADMM-based distributed algorithms," *arXiv preprint arXiv:1806.02246*, 2018.

[59] N. Heydaribeni and A. Anastasopoulos, "Distributed mechanism design for multicast transmission," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4200–4205.
[60] M. Soleymani and H. Mahdavifar, "Distributed multi-user secret sharing," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1141–1145.

Mahdi Soleymani (Student Member, IEEE) received his B.S. and M.S. degrees in Electrical Engineering at Sharif University of Technology, Tehran, Iran, in 2014 and 2016, respectively. He is currently pursuing his Ph.D. degree in Electrical Engineering and Computer Science at University of Michigan, Ann Arbor. His research interests lie in the area of algebraic coding theory with applications to distributed storage systems, wireless networks and distributed computing.

Hessam Mahdavifar (Member, IEEE) is an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan Ann Arbor. He received the B.Sc. degree from the Sharif University of Technology, Tehran, Iran, in 2007, and the M.Sc. and the Ph.D. degrees from the University of California San Diego (UCSD), La Jolla, in 2009, and 2012, respectively, all in electrical engineering. He was with the Samsung US R&D between 2012 and 2016, in San Diego, US, as a staff research engineer.

He received the NSF career award in 2020. He also received Best Paper Award in 2015 IEEE International Conference on RFID, and the 2013 Samsung Best Paper Award. He also received two Silver Medals at International Mathematical Olympiad in 2002 and 2003, and two Gold Medals at Iran National Mathematical Olympiad in 2001 and 2002. His main area of research is coding and information theory with applications to wireless communications, storage systems, security, and privacy.

A. Salman Avestimehr (Fellow, IEEE) is a Professor and director of the Information Theory and Machine Learning (vITAL) research lab at the Electrical and Computer Engineering Department of University of Southern California. He received his Ph.D. in 2008 and M.S. degree in 2005 in Electrical Engineering and Computer Science, both from the University of California, Berkeley. Prior to that, he obtained his B.S. in Electrical Engineering from Sharif University of Technology in 2003. His research interests include information theory and coding theory, and large-scale distributed computing and machine learning, secure and private computing, and blockchain systems.

Dr. Avestimehr 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 Communication Society Joint Paper Award, a Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House (President Obama), a Young Investigator Program (YIP) award from the U. S. Air Force Office of Scientific Research, a 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 a general Co-Chair of the 2020 International Symposium on Information Theory (ISIT).