Differentially Private Coded Computing

Hsuan-Po Liu*, Mahdi Soleymani†, and Hessam Mahdavifar*
*Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA
†Halıcıoğlu Data Science Institute, University of California San Diego, La Jolla, CA 92093, USA
Emails: hsuanpo@umich.edu, msoleymani@ucsd.edu, hessam@umich.edu

***Abstract*—Distributed computing has attracted significant recent attention for speeding up large-scale computations by disseminating computational jobs from a central master node across several worker nodes/servers. However, worker nodes are often untrusted and can also collude to gain unauthorized access to sensitive data. Hence, sharing sensitive data with them raises data privacy concerns. Coded computing has emerged as a promising framework for speeding up distributed computing and can be also adapted to address security and privacy concerns utilizing tools from secret sharing and multi-party computing. However, ensuring *perfect* information-theoretic privacy imposes a strict threshold on the maximum number of colluding workers the protocol can tolerate and, also, necessitates quantizing/mapping data to finite fields. Differential privacy is a widely accepted practical measure to capture the privacy leakage of the shared data. The mainstream approach is then to add perturbations to the data via randomized mechanisms. In this paper, we revisit coded computing, and especially when it is adapted to handle real-valued data, and analyze the privacy guarantees through the lens of differential privacy in terms of the (ϵ, δ)-differential privacy metric, for the first time in the literature. All the computations are done over the field of real/complex numbers and data privacy, in terms of differential privacy, is attained by adding noise terms in a certain structured way. In particular, the noise is added through the secret sharing mechanism (which can be, in principle, decoded and cancelled out at the master) as means of ensuring differential privacy. Furthermore, we propose a differentially private distributed matrix multiplication protocol for matrix multiplications that keeps the privacy of data in the worst adversarial case.**

I. Introduction

Distributed computation on large-scale data has received significant attention for modern and emerging computational paradigms [1]–[3]. In general, distributed computing is much more susceptible to adversarial attacks, compared to centralized ones, which necessitates addressing the critical issues of security and privacy in such systems [4]–[6]. Specifically, the dataset held by the central master node may contain highly sensitive information, such as financial transactions, personal medical records, biometrical data, etc. During the computation, some workers may collude with others to gain unauthorized access to the data. Hence, one of the major challenges is to utilize the computational power of the workers while preserving data privacy, in such a way that (almost) no information is revealed to the workers or certain subsets of the colluding workers, often up to a threshold in size.

Coded computing has emerged as a promising framework for speeding up distributed computing in the presence of straggler workers [1], [7]–[9] and can be also adapted to utilize a wide range of coding mechanisms [10]–[14] and to address security and privacy concerns utilizing tools from secret sharing and multi-party computing [15]–[24]. In these protocols, in order to achieve perfect privacy guarantees, the data symbols are assumed to be elements of a finite field. However, this often leads to accuracy losses due to the fixed-point representation of the data and may lead to computation overflows. In particular, this becomes a major barrier to the scalability of such protocols with respect to the dataset size. Furthermore, perfect information-theoretic privacy/security, i.e., with absolutely no leakage, cannot be guaranteed in the analog domain of real/complex numbers. Hence, adapting the coded computing protocols to the analog domain necessitates careful characterization of privacy leakage which is often lacking in the literature.

In [25]–[27], a framework is provided to construct the counterpart of Shamir's secret sharing scheme [28] in the analog domain, i.e., in the real/complex numbers, and to leverage it to deploy coded computing protocols over analog data. In this work, we analyze the privacy guarantees of such protocols through the lens of differential privacy by viewing the process of adding noise terms to the data/secret, as part of the analog Shamir secret sharing, as a randomized mechanism that can be utilized to as provide differential privacy guarantees. Note that the noise terms added during the secret sharing encoding process will be decoded and cancelled out later during the decoding process at the master node. This is done up to a certain accuracy loss in practical floating-point implementations. Roughly speaking, the magnitude of such an accuracy loss is directly related to the variance of noise terms. Therefore, a fundamental trade-off between the privacy guarantees of the protocol in terms of the (ϵ, δ)-differential privacy and the accuracy of outcome in a practical setting, assuming floating-point operations, is observed and characterized.

Differential privacy [29], [30] is a formal mathematical notion of privacy that provides a statistical characterization of protection against strong adversaries. Note that differential privacy is a *worst-case* notion of privacy, while several other notions, such as those based on mutual information or distinguishability (based on total variation distance) are *average-case* notions, see, e.g., [31]. Therefore, in a sense, differential privacy is considered to be a stronger notion compared to the *average-case* notions. The standard approach to achieving differential privacy guarantees is through perturbing the original data by introducing random noises generated according to a certain distribution. The key aspect of differential privacy is that the presence or absence of any individual symbol in the

data does not *noticeably* affect the final released statistical information. Differential privacy has been widely considered as one of the primary notions of data privacy in machine learning algorithms that enable training models over distributed data in a privacy-preserving fashion [32]–[35].

In this paper, we revisit coded computing, and especially when it is adapted to handle real-valued data [25], [26], and establish privacy guarantees in terms of (ϵ, δ)-differential privacy metric in the worst adversarial case. Furthermore, we propose a differentially private distributed computing protocol for matrix multiplication with privacy-preserving guarantees, also in the worst adversarial case. The proposed protocol allows multiplication between shares while maintaining the degree of the underlying polynomial, which relates to the tolerated threshold T on the number of colluding workers, by leveraging the so-called Beaver triples [36]. Specifically, our protocol enables performing multiplications between shares with $T = N - 1$, where N is the number of workers, which is the worst adversarial case, under the (ϵ, δ)-differential privacy guarantees.

The rest of the paper is structured as follows. In Section II, we present the protocol and analyze its accuracy. In Section III, the privacy guarantees of the protocol are characterized in terms of differential privacy measures. In Section IV, the protocol for multiplications between shares is proposed. Finally, we conclude the paper in Section V.

II. System Model and the Proposed Protocol

Consider a centralized setting with a master and N workers indexed by $[N] = \{1, 2, \ldots, N\}$. The protocol is executed in a synchronous environment with point-to-point secure communication channels between the master and workers. A data symbol s is referred to as a *secret/data*. We denote the computation on a data symbol by a P-degree polynomial as $f(s) = \sum_{i=0}^{P} f_i s^i$, which needs to be computed by the computation power of the N workers, and the data should remain *private* assuming there are T workers colluding, for some $T < N$, trying to infer the data. All workers are honest-but-curious, which means that they strictly follow the protocol, but the colluding workers would attempt to infer information about the data from the master. Note that collusion of $T + 1$ or more workers may fully reveal the data. We summarize the problem setting in Fig. 1.

The master randomly generates a polynomial $\mathcal{S}(x)$ for sharing the data $s \in \mathbb{R}$, where $\mathcal{S}(0) = s$ and $\deg(\mathcal{S}(x)) = T$, such that

$$\mathcal{S}(x) = s + \sum_{k=1}^{T} x^k n_k, \tag{1}$$

where $n_k \in \mathbb{R}$'s are noises sampled from *i.i.d.* Gaussian distribution $\mathcal{N}(0, \sigma_s^2)$. We resample $\sum_{k=1}^{T} x^k n_k$ by randomly generating n_k's, until both $\text{Real}(\sum_{k=1}^{T} x^k n_k)$, as the real part of $\sum_{k=1}^{T} x^k n_k$, and $\text{Imag}(\sum_{k=1}^{T} x^k n_k)$, as the imaginary part of $\sum_{k=1}^{T} x^k n_k$, are within the range of $[-t, t]$, for some $t \in \mathbb{R}^+$. The shares for the workers are given based on the publicly known complex-valued evaluation points $\omega_1, \ldots, \omega_N$, where

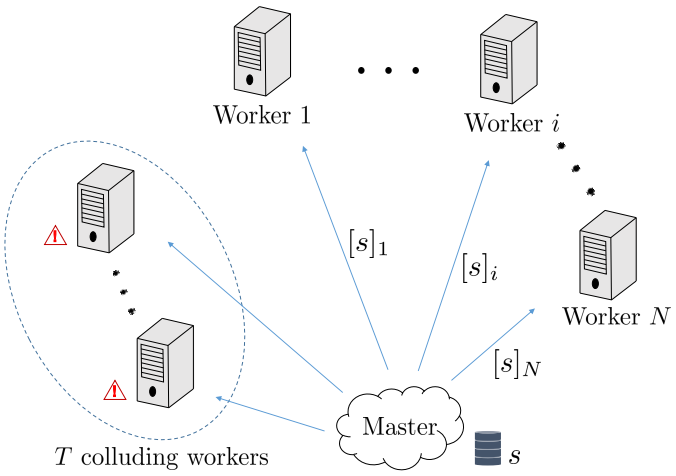

Fig. 1: Problem setting

$\omega_i = \exp(j\frac{2\pi i}{N})$ for $j^2 = -1$, such that the i-th worker receives a share as $\mathcal{S}(\omega_i)$, for all $i \in [N]$. For simplicity, we denote $\mathcal{S}(\omega_i)$ by $[s]_i \in \mathbb{C}$, and regard the secret sharing as applying a randomized mechanism $\mathcal{M}_i$ on the data s, which we referred to as the secret sharing mechanism, such that $[s]_i = \mathcal{M}_i(s)$, for $i \in [N]$. Thus, we formulate the polynomial as follows:

$$\mathcal{S}(\omega_i) = [s]_i = \mathcal{M}_i(s) = s + \sum_{k=1}^{T} \omega_i^k n_k = s + \tilde{n}_i, \tag{2}$$

where $\tilde{n}_i = \sum_{k=1}^{T} \omega_i^k n_k$, for $i \in [N]$. We resample $\tilde{n}_i$ by randomly generating n_k's, until both $\text{Real}(\tilde{n}_i)$, as the real part of $\tilde{n}_i$, and $\text{Imag}(\tilde{n}_i)$, as the imaginary part of $\tilde{n}_i$, are within the range of $[-t, t]$, for $i \in [N]$ and $t \in \mathbb{R}^+$. The truncated Gaussian distribution with zero mean, variance of σ^2, and a resampling parameter t is denoted by $\mathcal{TN}(0, \sigma^2; [-t, t])$ and its pdf is given as

$$p_{\tilde{N}}(y) = \frac{\phi(y)}{2\Phi(\frac{t}{\sigma}) - 1} \cdot \mathbb{I}_{[-t,t]}(y), \tag{3}$$

where $\phi(y)$ is the pdf for $\mathcal{N}(0, \sigma_s^2)$, $\mathbb{I}_{[-t,t]}(y)$ is an indicator function that $\mathbb{I}_{[-t,t]}(y) = 1$ for $y \in [-t, t]$, and $\mathbb{I}_{[-t,t]}(y) = 0$ otherwise. The following lemma shows the distribution of $\tilde{n}_i$, for $i \in [N]$.

Lemma 1. *The distribution of the combined noise $\tilde{n}_i$ is $\mathcal{TN}(0, \sigma^2; [-t, t])$, for $i \in [N]$, where $\sigma^2 = \sum_{k=1}^{T} |\omega_i^k|^2 \sigma_s^2$.*

The system of equations can be written as

$$[[s]_1, \ldots, [s]_N]^\top = \mathbf{G}[s, n_1, \ldots, n_T]^\top, \tag{4}$$

where $\mathbf{G}$ is the Vandermonde matrix associated with ω_i's as

$$\mathbf{G} = \begin{bmatrix} 1 & \omega_1 & \cdots & \omega_1^T \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N & \cdots & \omega_N^T \end{bmatrix}. \tag{5}$$

The i-th worker computes $f([s]_i)$ and then returns the result to the master, for $i \in [N]$. The master then recovers $f(s) = f(\mathcal{S}(0))$ based on the received results

$f([s]_1), \ldots, f([s]_N)$. We have $f([s]_i) = f(s) + a_1\omega_i + \cdots + a_{PT}\omega_i^{PT} = [1, \omega_i, \ldots, \omega_i^{PT}]\mathbf{a}$, for $i \in [N]$, where $\mathbf{a} = [f(s), a_1, \ldots, a_{PT}]^\top$ is the coefficients of $f(\mathcal{S}(x))$. Note that the degree of $f(\mathcal{S}(x))$ is PT since $\deg(f(s)) = P$ and $\deg(\mathcal{S}(x)) = T$. Thus, we can formulate a system of linear equations as

$$[f([s]_1), \ldots, f([s]_N)]^\top = \tilde{\mathbf{G}}\mathbf{a}, \tag{6}$$

where

$$\tilde{\mathbf{G}} = \begin{bmatrix} 1 & \omega_1 & \cdots & \omega_1^{PT} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N & \cdots & \omega_N^{PT} \end{bmatrix}. \tag{7}$$

To guarantee a successful interpolation of $f(\mathcal{S}(x))$ that $\deg(f(\mathcal{S}(x))) = PT$, we require $N \geq PT + 1$. For the rest of this section, we assume that $N = PT + 1$, i.e., all computation results for all workers should be returned to the master. To recover the result, the master does not need to recover the entire entries in $\mathbf{a}$ but only interests in the first entry, i.e., $f(s)$. Therefore, we need only the first row of $\tilde{\mathbf{G}}^{-1}$, which we denote it by $\tilde{\mathbf{g}}$. To recover the computation result $f(s)$, the master only needs to compute $\tilde{\mathbf{g}}[f([s]_1), \ldots, f([s]_N)]^\top$. It is worth noting that, since ω_i's are fixed for all $i \in [N]$, we only need to compute $\tilde{\mathbf{g}}$ once, then store it at the master for each recovery.

In [26], it is shown that the perturbation of the computations is bounded. Theoretically, if all computations are done under infinite precision, the outcome $f(s)$ would be computed accurately. However, data is represented by a finite number of bits in practice, either in fixed-point or floating-point. In floating-point representation, there are two parts, referred to fixed-precision part and the exponent part. It is assumed that the computations performed by the workers do not impose any errors other than precision loss due to the finite representation.

III. Analysis of Differential Privacy

In this section, we provide a preliminary of differential privacy. Then, we provide the definitions of differential privacy for the protocol and characterize the noise variance required in the protocol in order to satisfy the desired privacy level.

A. Overview of Differential Privacy

We provide the formal definition of (ϵ, δ)-differential privacy. Then, we analyze the privacy loss function.

Definition 1 ((ϵ, δ)-differential privacy)**.** *Let D and D' be two neighboring datasets, where $D, D' \in \mathcal{D}$, that only differ by a single record, i.e., $\mathrm{dist}(D, D') = 1$. Then D and D' satisfy (ϵ, δ)-differential privacy for any $\epsilon > 0$ and $\delta \in [0, 1]$ under a randomized mechanism $\mathcal{M}$ provided that under any event $\mathcal{E} \subseteq \mathrm{Range}(\mathcal{M})$, we have*

$$\mathbb{P}[\mathcal{M}(D) \in \mathcal{E}] \leq e^\epsilon \cdot \mathbb{P}[\mathcal{M}(D') \in \mathcal{E}] + \delta, \tag{8}$$

where δ is reffered to as the failure probability.

The sensitivity for a query function $f(\cdot)$ is the largest difference between the actual outcome and the perturbed output.

Definition 2 (l_2 sensitivity)**.** *For two neighboring datasets D and D' together with a query function $f : \mathcal{D} \to \mathbb{R}$, the l_2 sensitivity is defined as follows:*

$$\Delta \stackrel{\mathrm{def}}{=} \max_{\mathrm{dist}(D,D')=1} ||f(D) - f(D')||_2. \tag{9}$$

The Gaussian mechanism is defined as follows.

Definition 3 (Gaussian mechanism)**.** *Consider a query function f to be applied to a dataset D. Then the Gaussian mechanism $\mathcal{M}$ is defined as*

$$\mathcal{M}(D) \stackrel{\mathrm{def}}{=} f(D) + \mathcal{N}(0, \sigma^2),$$

which adds random noise to the query result according to a zero-mean Gaussian distribution with variance σ^2.

Analysis of Privacy Loss: Consider two neighboring datasets D and D' and a function f. For a randomized mechanism $\mathcal{M}$, the probability density function (pdf) corresponding to the datasets D and D' are denoted as $p_{\mathcal{M}(D)}(y)$ and $p_{\mathcal{M}(D')}(y)$, respectively. Let

$$l_{\mathcal{M},D,D'}(y) \stackrel{\mathrm{def}}{=} \ln[\frac{p_{\mathcal{M}(D)}(y)}{p_{\mathcal{M}(D')}(y)}], \tag{10}$$

which is referred to as the privacy loss function, and, $L_{\mathcal{M},D,D'} = l_{\mathcal{M},D,D'}(Y)$, is referred to the privacy loss random variable. Then, the (ϵ, δ)-differential privacy implies $\mathbb{P}[L_{\mathcal{M},D,D'} \leq \epsilon] \geq 1 - \delta$. Consider the case where $f(D) = 0$ and $f(D') = \Delta$, we have

$$p_{\mathcal{M}(D)}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}, \tag{11}$$

and

$$p_{\mathcal{M}(D')}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\Delta)^2}{2\sigma^2}}. \tag{12}$$

Note that although the ratio of probabilities is always positive, the result after taking the logarithm may become negative. Thus, typically, the absolute value of the privacy loss function is considered as

$$|l_{\mathcal{M},D,D'}(y)| = |ln(\frac{e^{(-1/2\sigma^2)y^2}}{e^{(-1/2\sigma^2)(y-\Delta)^2}})| = |\tfrac{1}{2\sigma^2}(2y\Delta - \Delta^2)|. \tag{13}$$

By the definition in (8), one needs

$$|\frac{1}{2\sigma^2}(2y\Delta - \Delta^2)| < \epsilon \tag{14}$$

with probability at least $1 - \delta$ to guarantee (ϵ, δ)-differential privacy.

B. Differential Privacy Analysis of the Protocol

Now we analyze the differential privacy of the protocol. Note that the analysis of differential privacy build upon the worst adversarial case of worker collusion, i.e., $T = N - 1$. Furthermore, we extend the analysis to matrices for more general consideration, i.e., the master has a dataset $\mathbf{S} \in \mathbb{R}^{m \times n}$, and its share to the i-th worker becomes $[\mathbf{S}]_i = \mathbf{S} + \sum_{k=1}^{T} \omega_i^k \mathbf{N}_k$, for $i \in [N]$, where $\mathbf{N}_k \in \mathbb{R}^{m \times n}$.

Definition 4 (Differential privacy)**.** *Given a randomized mechanism $\mathcal{M}_i : \mathbb{R}^{m\times n} \to \mathbb{C}^{m\times n}$, for $i \in [N]$. A protocol is (ϵ, δ)-differentially private, if for the neighboring datasets $\mathbf{S}, \mathbf{S}' \in \mathbb{R}^{m\times n}$, where $\text{dist}(\mathbf{S}, \mathbf{S}') = 1$, and $\mathcal{T} \subset \mathbb{C}^{m\times n}$,*

$$\mathbb{P}(\mathcal{M}_i(\mathbf{S}) \in \mathcal{T}) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{M}_i(\mathbf{S}') \in \mathcal{T}) + \delta. \tag{15}$$

Note that the randomized mechanism $\mathcal{M}_i$ is the secret sharing mechanism for the master sharing its dataset to the i-th worker as $\mathcal{M}_i(\mathbf{S}) = [\mathbf{S}]_i = \mathbf{S} + \sum_{k=1}^{T} \omega_i^k \mathbf{N}_k$, for $i \in [N]$. Then, we define the sensitivity of the protocol.

Definition 5 (Sensitivity)**.** *Let $\mathbf{S}$ and $\mathbf{S}'$ denote two neighboring datasets, where $\mathbf{S}, \mathbf{S}' \in \mathbb{R}^{m\times n}$, such that $\text{dist}(\mathbf{S}, \mathbf{S}') = 1$. We define the sensitivity of the datasets as*

$$\Delta = \max_{\text{dist}(\mathbf{S},\mathbf{S}')=1} ||\mathbf{S} - \mathbf{S}'||_F. \tag{16}$$

We formulate the absolute value of the privacy loss function for scalar-valued computations in the following lemma.

Lemma 2. *Consider a pair of neighboring datasets $D, D' \in \mathbb{R}$ in the protocol, where $D = D' - \Delta$. Then the absolute value of the privacy loss for $i \in [N]$ is*

$$|l_{\mathcal{M}_i,D,D'}(y)| = |\frac{1}{2\sigma^2}(-2y\Delta + (\Delta)^2)| \cdot \mathbb{I}_{[-t+\Delta,t]}(y).$$

In the following theorem, we show that, under a certain constraint on the neighboring datasets, the problem of designing the protocol with input datasets in the matrix form being (ϵ, δ)-differential private can be reduced to the one with scalar-valued data as the input.

Theorem 3. *For a pair of neighboring datasets $\mathbf{S}$ and $\mathbf{S}'$, where $\mathbf{S}, \mathbf{S}' \in \mathbb{R}^{m\times n}$, the high-dimensional case can be relaxed to a scalar-valued case if $\mathbf{S} = \mathbf{S}' + \mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{m\times n}$ and $||\mathbf{W}||_F \leq \Delta$.*

The result of Theorem 3 implies that we can only consider the scalar-valued computations for satisfying the (ϵ, δ)-differential privacy. The following theorem characterizes the (ϵ, δ)-differential privacy for the protocol.

Theorem 4. *The protocol is (ϵ, δ)-differentially private if and only if*

$$1 - \frac{\Phi(\frac{\sigma\epsilon}{\Delta} + \frac{\Delta}{2\sigma}) - \Phi(-\frac{\sigma\epsilon}{\Delta} + \frac{\Delta}{2\sigma})}{2\Phi(\frac{t}{\sigma}) - 1} \leq \delta, \tag{17}$$

where $\sigma \in (0, \sqrt{\frac{t\cdot\Delta}{\epsilon} - \frac{\Delta^2}{2\epsilon}})$ and $\Phi(\cdot)$ is the cumulative density function (cdf) for the standard normal distribution.

For the sake of simplifying the derivations in the rest of the paper, we introduce a variable α where $\sigma = \frac{\alpha\cdot\Delta}{\sqrt{2\epsilon}}$.

Corollary 5. *Given $\epsilon, \delta, t, \Delta$, we can simplify* (17) *by introducing a variable α which satisfies $\sigma = \frac{\alpha\cdot\Delta}{\sqrt{2\epsilon}}$ as follows:*

$$1 - \frac{\Phi(\sqrt{\frac{\epsilon}{2}}(\alpha + \frac{1}{\alpha})) - \Phi(\sqrt{\frac{\epsilon}{2}}(-\alpha + \frac{1}{\alpha}))}{2\Phi(\frac{t\sqrt{2\epsilon}}{\alpha\cdot\Delta}) - 1} \leq \delta, \tag{18}$$

where $\alpha \in (0, \sqrt{\frac{2t}{\Delta} - 1})$.

For simplicity, let the LHS of (18) be denoted by $B(\alpha)$, i.e.,

$$B(\alpha) = 1 - \frac{\Phi(\sqrt{\frac{\epsilon}{2}}(\alpha + \frac{1}{\alpha})) - \Phi(\sqrt{\frac{\epsilon}{2}}(-\alpha + \frac{1}{\alpha}))}{2\Phi(\frac{t\sqrt{2\epsilon}}{\alpha\cdot\Delta}) - 1}. \tag{19}$$

The following lemmas demonstrate that $B(\alpha)$ is a non-negative monotonically decreasing function for $\alpha \in (0, \sqrt{\frac{2t}{\Delta} - 1})$.

Lemma 6. *$B(\alpha)$, specified in* (19)*, is a non-negative function.*

Lemma 7. *$B(\alpha)$, specified in* (19)*, is monotonically decreasing for $\alpha \in (0, \sqrt{\frac{2t}{\Delta} - 1})$.*

Building upon Lemmas 6 and 7, we are now ready to present the proposed analytical truncated Gaussian mechanism of choosing the optimal noise for the protocol.

Theorem 8 (Analytical truncated Gaussian mechanism)**.** *The protocol satisfies (ϵ, δ)-differential privacy for $\sigma = \frac{\alpha^*\cdot\Delta}{\sqrt{2\epsilon}}$, where α^* is obtained by:*

$$\begin{aligned} \alpha^* = &\operatorname*{argmax}_{\alpha} B(\alpha) \\ &\text{s.t. } 0 \leq B(\alpha) \leq \delta, \\ &0 < \alpha < \sqrt{\frac{2t}{\Delta} - 1}. \end{aligned} \tag{20}$$

Theorem 8 characterizes an optimization problem for finding the optimal noise for satisfying the (ϵ, δ)-differential privacy. The following theorem demonstrates the detailed steps for obtaining the value of noise variance.

Theorem 9 (The protocol with differential privacy guarantee under the analytical truncated Gaussian mechanism)**.** *In order to guarantee (ϵ, δ)-differential privacy for the protocol, it is sufficient to set the variance of each element in the noise matrices $\mathbf{N}_1, ..., \mathbf{N}_T$ as*

$$\sigma_s^2 = \frac{(\alpha^*)^2 \cdot \Delta^2}{2\epsilon T}. \tag{21}$$

IV. Differentially Private Multiplication Triples in the Analog Domain

In this section, we propose a protocol that is capable of multiplying two shares with both of them having their datasets embedded in a polynomial at degrees of $T = N - 1$, which is the worst adversarial case, while maintaining the degree of the polynomial at $T = N - 1$, so that the same privacy analysis based on the (ϵ, δ)-differential privacy metric could be recycled. Naturally, if we directly multiply two shares with degrees at T, the degree of the polynomial increases to $2T$. To ensure a successful recovery at the master, the degree of polynomial after multiplication cannot exceeds $N - 1$.

To tackle the issue of the degree of polynomials, [36] proposed a protocol for a fully decentralized setting, where all computations are performed as elements of a finite field. The protocol in [36], however under the setting of a decentralized system, still requires a third trusted entity in an *offline phase* to generate a pair of *triple* and then shared the triple with all parties/workers. Inspired by [36], since we naturally have a

trusted entity, i.e., the master itself, we propose a differentially private protocol for matrix multiplications for computations in the infinite field as real/complex numbers, while capable of keeping the setting in the worst adversarial case as $T = N - 1$. With our proposed protocol for multiplications, we can multiply any shares even considering the worst adversarial case of an adversary as $T = N - 1$ since the protocol does not increase the degree of the polynomial in the shares. It is known to reduce communication overhead between the master and the worker nodes by dividing the process into an *offline phase* and an *online phase*.

A. Offline phase

In the offline phase, the master first generates random triple $\mathbf{A}, \mathbf{B}, \mathbf{C}$ such that $\mathbf{AB} = \mathbf{C}$ where $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$, and $\mathbf{C} \in \mathbb{R}^{m_1 \times n_2}$, for $n_1 = m_2$. Note that the entries of $\mathbf{A}$ and $\mathbf{B}$ are *i.i.d.* and chosen according to a Gaussian distribution $\mathcal{N}(0, \sigma_s^2)$. For the entries whose absolute values exceed t, we resample them until the result is within the range of $[-t, t]$. Then, the master secretly shares $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ to all workers by utilizing the protocol, in such a way that the i-th worker holds three shares of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, denoted by $[\mathbf{A}]_i$, $[\mathbf{B}]_i$, and $[\mathbf{C}]_i$, for $i \in [N]$, respectively. Note that this phase does not involve the input data to the protocol, and that is why it is referred to as the offline phase.

B. Online phase

At the beginning of this phase, the master receives two input data as matrices $\mathbf{U}$ and $\mathbf{V}$, where $\mathbf{U} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{V} \in \mathbb{R}^{m_2 \times n_2}$. Note that we must have $n_1 = m_2$ for the matrix multiplication. The matrices are then secretly shared with all workers, thus the i-th worker holds shares $[\mathbf{U}]_i$ and $[\mathbf{V}]_i$, for $i \in [N]$. Together with the shares from the offline phase, at this stage, the i-th worker holds shares $[\mathbf{A}]_i$, $[\mathbf{B}]_i$, $[\mathbf{C}]_i$, $[\mathbf{U}]_i$, and $[\mathbf{V}]_i$, for $i \in [N]$.

To proceed the algorithm, we assign two shares $[\mathbf{D}]_i$ and $[\mathbf{E}]_i$ as $[\mathbf{D}]_i = [\mathbf{U}]_i - [\mathbf{A}]_i$ and $[\mathbf{E}]_i = [\mathbf{V}]_i - [\mathbf{B}]_i$, where $\mathbf{D} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{E} \in \mathbb{R}^{m_2 \times n_2}$, respectively. Then, both the matrices $[\mathbf{D}]_i$ and $[\mathbf{E}]_i$ are returned to the master. The master then recover the result of $\mathbf{D}$ and $\mathbf{E}$ such that $\mathbf{D} = \mathbf{U} - \mathbf{A}$ and $\mathbf{E} = \mathbf{V} - \mathbf{B}$, respectively. At this stage, the i-th worker holds shares of $[\mathbf{A}]_i$, $[\mathbf{B}]_i$, $[\mathbf{C}]_i$, $[\mathbf{U}]_i$, $[\mathbf{V}]_i$, and public matrices $\mathbf{D}$ and $\mathbf{E}$, for $i \in [N]$.

Next, the i-th worker computes its share of the multiplication, for $i \in [N]$, denoted by $[\mathbf{UV}]_i$, as follows

$$[\mathbf{UV}]_i = \mathbf{D}[\mathbf{B}]_i + [\mathbf{A}]_i\mathbf{E} + \mathbf{DE} + [\mathbf{C}]_i. \tag{22}$$

In order to decode the desired computation result $\mathbf{UV}$, each worker returns shares $[\mathbf{UV}]_i$, for $i \in [N]$, to the master. By substituting $\mathbf{D} = \mathbf{U} - \mathbf{A}$ and $\mathbf{E} = \mathbf{V} - \mathbf{B}$, the master obtains

$$\begin{aligned}&\mathbf{DB} + \mathbf{AE} + \mathbf{DE} + \mathbf{C}\\ &= (\mathbf{U} - \mathbf{A})\mathbf{B} + (\mathbf{V} - \mathbf{B})\mathbf{E} + (\mathbf{U} - \mathbf{A})(\mathbf{V} - \mathbf{B}) + \mathbf{C}\\ &= \mathbf{UV}.\end{aligned}$$

At the end of this step, the master successfully recovers the multiplication result for the two matrices $\mathbf{U}$ and $\mathbf{V}$ as $\mathbf{UV}$.

Algorithm 1 Differentially private matrix multiplication

Input: Number of workers N, number of colluding workers T, public parameters ω_i's for $i \in [N]$, the inputs $\mathbf{U} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{V} \in \mathbb{R}^{m_2 \times n_2}$.

Output: $\mathbf{UV}$

Offline stage.

Randomly generate matrices $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$, where all entries in both $\mathbf{A}$ and $\mathbf{B}$ are in $\mathcal{TN}(0, \sigma^2; [-t, t])$, and $\mathbf{C} \in \mathbb{R}^{m_1 \times n_2}$ given that $\mathbf{AB} = \mathbf{C}$. Note that $n_1 = m_2$.

Secretly shares $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to all workers so that worker i holds $[\mathbf{A}]_i$,$[\mathbf{B}]_i$, and $[\mathbf{C}]_i$, for $i \in [N]$.

Online stage.

The master receives $\mathbf{U} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{V} \in \mathbb{R}^{m_2 \times n_2}$.

The master secretly shares the inputs $\mathbf{U}, \mathbf{V}$ to all workers.
// Worker i holds $[\mathbf{U}]_i, [\mathbf{V}]_i$, $[\mathbf{A}]_i, [\mathbf{B}]_i$, $[\mathbf{C}]_i$, for $i \in [N]$.

Worker i locally computes $[\mathbf{D}]_i = [\mathbf{U}]_i - [\mathbf{A}]_i$ and $[\mathbf{E}]_i = [\mathbf{V}]_i - [\mathbf{B}]_i$, for $i \in [N]$.

Worker i send the shares of $[\mathbf{D}]_i$ and $[\mathbf{E}]_i$ to the master, for $i \in [N]$.

The master recovers $\mathbf{D}$ and $\mathbf{E}$. Then the master shares $\mathbf{D}$ and $\mathbf{E}$ publicly with all workers.
// Worker i now holds $\mathbf{D}, \mathbf{E}$, $[\mathbf{A}]_i, [\mathbf{B}]_i, [\mathbf{C}]_i$, for $i \in [N]$.

Worker i computes $[\mathbf{UV}]_i = \mathbf{D}[\mathbf{B}]_i + [\mathbf{A}]_i\mathbf{E} + \mathbf{DE} + [\mathbf{C}]_i$, for $i \in [N]$.

Worker i sends the share $[\mathbf{UV}]_i$ to the master, for $i \in [N]$.

The master recovers $\mathbf{UV}$ based on $[\mathbf{UV}]_i$'s, for $i \in [N]$.

Note that for each multiplication between a pair of shares, we must use new pair of triples. We summarize the differentially private matrix multiplication in Algorithm 1.

Remark 1. *Note that in the online stage for the multiplication triple, the publicly revealed parameters* $\mathbf{D}$ *and* $\mathbf{E}$ *involve the input data matrices* $\mathbf{U}$ *and* $\mathbf{V}$*. Therefore, there is some privacy leakage as the result of this particular step that needs to be carefully characterized. Since we randomly generate the entries of* $\mathbf{A}$ *and* $\mathbf{B}$ *according to* $\mathcal{TN}(0, \sigma^2; [-t, t])$*, the entries in* $\mathbf{D}$ *and* $\mathbf{E}$ *are, in a sense from the privacy-preserving perspective, in the same form as the shares specified in* (2)*. Therefore, the same privacy analysis could be recycled for the protocol of differentially private matrix multiplication.*

V. Conclusion

In this paper, we revisit coded computing under the computation of real-valued data, and analyze the privacy guarantees in terms of the (ϵ, δ)-differential privacy metric. All the computations are done over the field of real/complex numbers. Data privacy, in terms of differential privacy, is attained by adding noise terms through the secret sharing mechanism as means of ensuring differential privacy. Furthermore, we propose a differentially private distributed matrix multiplication protocol for matrix multiplications that keeps the privacy of data in the worst adversarial case.

REFERENCES

[1] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1143–1147.

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

[3] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 909–910.

[4] 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, p. 713–718.

[5] 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.

[6] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explor. Newsl.*, vol. 4, no. 2, p. 28–34, 2002.

[7] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 4406–4416.

[8] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[9] N. Charalambides, H. Mahdavifar, and A. O. Hero, "Numerically stable binary gradient coding," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 2622–2627.

[10] A. Khalesi and P. Elia, "Multi-user linearly separable computation: A coding theoretic approach," in *2022 IEEE Information Theory Workshop (ITW)*. IEEE, 2022, pp. 428–433.

[11] N. Charalambides, M. Pilanci, and A. O. Hero, "Secure linear mds coded matrix inversion," in *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2022, pp. 1–8.

[12] N. A. Khooshemehr and M. A. Maddah-Ali, "Vers: fully distributed coded computing system with distributed encoding," *arXiv preprint arXiv:2304.05691*, 2023.

[13] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 4185–4196, 2021.

[14] D. Fathollahi and M. Mondelli, "Polar coded computing: The role of the scaling exponent," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 2154–2159.

[15] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," in *International Conference on Artificial Intelligence and Statistics*, 2018.

[16] S. R. H. Najarkolaei, M. A. Maddah-Ali, and M. R. Aref, "Coded secure multi-party computation for massive matrices with adversarial nodes," in *2020 Iran Workshop on Communication and Information Theory (IWCIT)*, 2020, pp. 1–6.

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

[18] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[19] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7420–7437, 2021.

[20] R. G. L. D'Oliveira, S. El Rouayheb, and D. Karpuk, "GASP codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4038–4050, 2020.

[21] M. Kim and J. Lee, "Private secure coded computation," 2019. [Online]. Available: https://arxiv.org/abs/1902.00167

[22] M. Soleymani, M. V. Jamali, and H. Mahdavifar, "Coded computing via binary linear codes: Designs and performance limits," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 879–892, 2021.

[23] N. Charalambides, H. Mahdavifar, M. Pilanci, and A. O. Hero, "Orthonormal sketches for secure coded regression," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 826–831.

[24] C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the singleton bound," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 275–285, 2022.

[25] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog privacy-preserving coded computing," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1865–1870.

[26] ——, "Analog secret sharing with applications to private distributed learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1893–1904, 2022.

[27] ——, "Analog lagrange coded computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

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

[29] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, p. 211–407, 2014.

[30] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*. Springer, 2006, pp. 265–284.

[31] W. Wang, L. Ying, and J. Zhang, "On the relation between identifiability, differential privacy, and mutual-information privacy," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5018–5029, 2016.

[32] H. B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz, "A general approach to adding differential privacy to iterative training procedures," 2018. [Online]. Available: https://arxiv.org/abs/1812.06210

[33] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[34] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[35] N. Wu, F. Farokhi, D. Smith, and M. A. Kaafar, "The value of collaboration in convex machine learning with differential privacy," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 304–317.

[36] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology - CRYPTO '91*. Springer, 1992, pp. 420–432.