

Secure Coded Multi-Party Computation for Massive Matrix Operations

Hanzaleh Akbari Nodehi*, Mohammad Ali Maddah-Ali†

*Department of Electrical Engineering, Sharif University of Technology

†Nokia Bell Labs

Abstract

In this paper, we consider a secure multi-party computation problem (MPC), where the goal is to offload the computation of an arbitrary polynomial function of some massive private matrices (inputs) to a cluster of workers. The workers are not reliable. Some of them may collude to gain information about the input data (semi-honest workers). The system is initialized by sharing a (randomized) function of each input matrix to each server. Since the input matrices are massive, each share's size is assumed to be at most $1/k$ fraction of the input matrix, for some $k \in \mathbb{N}$. The objective is to minimize the number of workers needed to perform the computation task correctly, such that even if an arbitrary subset of $t-1$ workers, for some $t \in \mathbb{N}$, collude, they cannot gain any information about the input matrices. We propose a sharing scheme, called *polynomial sharing*, and show that it admits basic operations such as adding and multiplication of matrices and transposing a matrix. By concatenating the procedures for basic operations, we show that any polynomial function of the input matrices can be calculated, subject to the problem constraints. We show that the proposed scheme can offer order-wise gain in terms of the number of workers needed, compared to the approaches formed by the concatenation of job splitting and conventional MPC approaches.

Index Terms

multi-party computation, polynomial sharing, secure computation, massive matrix computation

I. INTRODUCTION

With the growing size of datasets in use cases such as machine learning and data science, it is inevitable to distribute the computation tasks to some external entities, which are not necessarily trusted. In this set-up, some of the major challenges are protecting the privacy of the data, guaranteeing the correctness of the result, and ensuring the efficiency of the computation.

The problem of processing private information on some external parties has been studied in the context of *secure multi-party computation (MPC)*. Informally, in an MPC problem, some private data inputs are available in some source nodes, and the goal is to offload the computation of a specific function of those inputs to some parties (workers). These parties are not reliable. Some of them may collude to gain information about the private inputs (semi-honest workers) or even behave adversarially to make the result incorrect. Thus, the objective is to design a scheme, probably based on coding and randomization techniques, to ensure data privacy and correctness of the result. To ensure privacy, some MPC solutions, such as [2], [3], rely on cryptographic hardness assumptions, while others, such as [4]–[6], are protected based on information-theoretic measures (See [7] for a survey on different approaches of MPC). In particular, the BGW scheme, named after its inventors, Ben-Or, Goldwasser, and Wigderson in [4], relies on Shamir secret sharing [8] to develop an information-theoretically private MPC scheme to calculate any polynomial of private inputs. Shamir secret sharing is an approach that allows us to share a secret, i.e., private input, among some parties, such that if the number of colluding nodes is less than a threshold, they cannot gain any information about the data. The BGW scheme exploits the fact that Shamir secret sharing admits basic operations such as addition and multiplication (at the cost of some communication among nodes).

This is an extended version of the paper, partially presented in IEEE Communication Theory Workshop (CTW), May 2018, and IEEE International Symposium on Information Theory (ISIT), June 2018 [1].

There have been some efforts to improve MPC algorithms' efficiency, but mainly focusing on the communication loads (see [9]). However, less effort has been dedicated to the cases where the input data is massive. One approach would be to split the computation to some smaller subtasks and dedicate a group of workers to execute each subtask, using conventional MPC approaches. In this paper, we argue that the idea of the concatenation of job splitting and multi-party computation could be significantly sub-optimal in terms of the number of workers needed.

In a seemingly irrelevant area, extensive efforts have been dedicated to using coding theory to improve the efficiency of distributed computing, mainly to cope with the stragglers [10]–[20]. The core idea is based on partitioning each input data into some smaller inputs and then encoding the smaller inputs. The workers then process those encoded inputs. The ultimate goal is to design the code such that the computation per worker node is limited to what it can handle, and also the final result can be derived from the outcomes of a subset of worker nodes. This is done for matrix to vector multiplication in [11], and matrix to matrix multiplication in [12]–[16]. In particular, in [13], the code is designed such that the results of different workers form a maximum distance separable (MDS), meaning that the final result can be recovered from any subset of servers with the minimum size. That approach has been extended to general matrix partitioning in [14], [15], and also for the cases where only an approximate result of the matrix multiplication is needed in [17], [20]. This paper aims to design efficient MPC schemes for massive inputs, exploiting the ideas developed in the context of coding for computing. Let us first review a motivating example that justifies our objective.

Example 1 (Maximum likelihood regression on private data:). Assume that a research center wants to run a machine learning algorithm, say maximum likelihood regression, on a collection of private data sets, where each data set belongs to an organization. For example, each data set can be the salary information of a company (see [21]) or the medical records of a hospital. Those organizations want to help the research center, but do not want to reveal any information about their data sets beyond the final result. In particular, assume that there are $\Gamma \in \mathbb{N}$ organizations, where organization $\gamma \in \{1, 2, \dots, \Gamma\}$, has a matrix $\mathbf{X}^{[\gamma]}$ and a corresponding target vector $\mathbf{b}^{[\gamma]}$. The target value of each row of $\mathbf{X}^{[\gamma]}$ is the corresponding entry of vector $\mathbf{b}^{[\gamma]}$. The research center aims to find a regression model $b = \mathbf{w}^T \mathbf{x}$, representing the relationship between each row of \mathbf{X} and the corresponding entry of \mathbf{b} , where

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^{[1]} \\ \vdots \\ \mathbf{X}^{[\Gamma]} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}^{[1]} \\ \vdots \\ \mathbf{b}^{[\Gamma]} \end{bmatrix}. \quad (1)$$

Following the maximum likelihood solution for regression (see [22][Chapter 3.1.1], the center needs to calculate

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \quad (2)$$

where $(\mathbf{X}^T \mathbf{X})^{-1} \approx \mathbf{I} + \mathbf{A} + \dots + \mathbf{A}^k$, for $\mathbf{A} = \mathbf{I} - \mathbf{X}^T \mathbf{X}$ and some $k \in \mathbb{N}$ (assuming that the datasets are normalized such that the largest eigenvalue of \mathbf{A} is less than one). For processing, assume that there are some servers available, where none of them has enough computing and storage resources to execute this computation alone. On top of that, at most $t - 1$, for some integer t , of them may collude to gain information about the private data. Conventional MPC schemes, such as the BGW scheme [4], can handle computing polynomial functions of some private inputs (See [7]), but are not designed to handle scenarios where the inputs are massive matrices. In this paper, we aim to address such scenarios. \diamond

In this paper we consider a system, including Γ sources, N workers, and one master. There is a link between each source and each worker. All of the workers are connected to each other and also are connected to the master. Each source sends a function of its data (so-called a share) to each worker. We assume that the workers have limited computation resources. As a proxy to that limit, we assume that each share's size can be up to a certain fraction of the corresponding input size. The workers process

their inputs, and in between, they may communicate with each other. After that, each worker sends a message to the master, such that the master can recover the required function of the inputs. The sharing and the computation procedures must be designed such that if any subset of $t - 1$ workers collude, for some $t \in \mathbb{N}$, they can not gain any information about the inputs. Also, the master must not gain any additional information, beyond the result, about the inputs. Motivated by recent results in coding for matrix multiplication, as an extension to Shamir secret sharing, in this paper, we propose a new sharing approach called *polynomial sharing*. We show that the proposed sharing approach admits basic operations such as addition, multiplication, and transposing, by developing a procedure for each of them. Finally, we prove that we can compute any polynomial function using these procedures while preserving the privacy subject to the storage limit of each worker node. We show that the number of servers needed to compute a function using this approach is order-wise less than what we need in approaches based on job splitting and the conventional BGW scheme.

The rest of the paper is organized as follows. In Section II, we formally state the problem setting. In Section III, we review some preliminaries and conventional approaches for MPC. In section IV, we state the main result. In Section V, we review some motivating examples. In Section VI, we present the *polynomial sharing* scheme. In Section VII, we show several procedures to perform basic operations, such as addition, multiplication, and transposing, using the proposed sharing scheme. In Section VIII, we present the algorithm to calculate general polynomials. Finally in Section IX, we present some extensions.

Notation: In this paper matrices and vectors are denoted by upper boldface letters and lower boldface letters respectively. For $n_1, n_2 \in \mathbb{Z}$ the notation $[n_1, n_2]$ represents the set $\{n_1, n_1 + 1, \dots, n_2\}$. Also, $[n]$ denotes the set $\{1, \dots, n\}$ for $n \in \mathbb{N}$. Furthermore, the cardinality of a set \mathcal{S} is denoted by $|\mathcal{S}|$. For the arbitrary field \mathbb{F} , the notation \mathbb{F}^* means any matrix with any possible size, with entries from \mathbb{F} . For a matrix \mathbf{A} , $\mathbf{A}(a : b, :)$ denotes a matrix including rows a to b of matrix \mathbf{A} .

A. Concurrent and Follow-up Results

The early version of this paper was submitted to IEEE ISIT 2018 in Jan. 2018, which has been appeared in June 2018 [1]. In addition, it was presented in CTW 2018 in May 2018. We have also presented a generalized version of [1] in [23].

In parallel in [19] and [24], the authors introduce Lagrange coded computing, targeting the case where the computation of interest can be decomposed into computing one polynomial function for several (private) inputs. In other words, the computing task can be stated as computing $\tilde{\mathbf{G}}(\mathbf{X}^{[1]}), \tilde{\mathbf{G}}(\mathbf{X}^{[2]}), \dots, \tilde{\mathbf{G}}(\mathbf{X}^{[\Gamma]})$ for an arbitrary polynomial function $\tilde{\mathbf{G}}(\cdot)$, and inputs $\mathbf{X}^{[1]}, \mathbf{X}^{[2]} \dots, \mathbf{X}^{[\Gamma]}$. The decomposition must be such that one worker can handle one computing of $\tilde{\mathbf{G}}(\cdot)$ for an input. The idea is then to use coding over those computations to form coded redundancy to deal with stragglers and/or guarantee privacy.

The major difference between [19], [24], and what we do in this paper is that in [19] and [24], one server *can* handle computing $\tilde{\mathbf{G}}(\mathbf{X}^{[\gamma]})$, for the input $\mathbf{X}^{[\gamma]}$, $\gamma \in [\Gamma]$. However, in this work, the objective is to compute $\mathbf{G}(\mathbf{X}^{[1]}, \mathbf{X}^{[2]} \dots, \mathbf{X}^{[\Gamma]})$, for an arbitrary polynomial \mathbf{G} , where one server cannot handle computing $\mathbf{G}(\mathbf{X}^{[1]}, \mathbf{X}^{[2]} \dots, \mathbf{X}^{[\Gamma]})$ alone. It is clear that for an arbitrary polynomial function $\mathbf{G}(\mathbf{X}^{[1]}, \mathbf{X}^{[2]} \dots, \mathbf{X}^{[\Gamma]})$, we cannot always transform it into computing some independent calculations $\tilde{\mathbf{G}}(\mathbf{X}^{[1]}), \tilde{\mathbf{G}}(\mathbf{X}^{[2]}) \dots, \tilde{\mathbf{G}}(\mathbf{X}^{[\Gamma]})$ (see Example 1).

Another difference is that in [19] and [24], there is no communication among the workers and the number of workers needed for coding to be effective, is at least $(\Gamma - 1) \deg(\tilde{\mathbf{G}}) + 1$. However, in the scheme that we propose in this paper, there is communication among the workers and the number of servers needed does not grow with Γ or the degree of \mathbf{G} .

The idea of [19] and [24] has been used to train some machine learning algorithms [25], in which the computation can be decomposed into calculating one specific function for several inputs. To avoid requiring too many workers, in [25], it is suggested to approximate the specific function to a low degree polynomial function.

Another related line of research, started by [26], published in June 2018 on arXiv, is known as secure matrix multiplication. In that set-up, the objective is to calculate the multiplication of two matrices without

leaking information to the workers. The scheme of [26] was later improved by the follow-up results [27], [28]. The difference between what we do in this paper and secure matrix multiplication [26]–[28] is that we are interested in the result of a general polynomial function of multiple private inputs, while in secure matrix multiplication, the master is interested in the result of multiplication of two matrices. Also, there is no privacy constraint for the master at the secure matrix multiplication, and there is no communication between the workers. Still, we can consider the problem of secure matrix multiplication as a special case of the proposed scenario in this paper, ignoring the extra privacy constraint that we have at the master. Indeed, the scheme that we had already proposed in [1] outperforms the scheme of the concurrent work [26] and the follow-up work [27]. The other follow-up paper [28] reports two schemes, named as GAPS-Big and GAPS-Small. Indeed, GAPS-Big is the same as what we had already reported in [1]. The scheme GAPS-Small performs better than what we report in this paper for the case of $3 \leq t < k$.

Some recent result [29] on secure matrix multiplications focuses on calculating pairwise multiplications of *several pairs* of matrices, rather than just one pair. This will reduce the cost of randomization per pair of multiplication (using ideas such as cross-subspace alignment) and improve the efficiency [29]. [30] considers private matrix multiplications where workers do not collude, but still the master wants to keep the input matrices private from each worker. [30] also considers the case where one of the matrices is selected from a finite and known set, and the objective is to keep the index of that matrix private. In [31], the authors propose a code for private matrix multiplication that is flexible to achieve a trade-off between the number of workers needed and the communication load. They also consider the case where one of the matrices is selected from a finite and publicly known set of matrices. A different approach to privacy in computing is introduced in [32], [33], where the function of interest is formed by a specific concatenation and combination of several known linear functions, represented by matrix operations, and the objective is to keep the order of concatenation/combination private.

II. PROBLEM SETTING

Consider an MPC system including Γ source nodes, N worker nodes, and one master node, for some $\Gamma, N \in \mathbb{N}$ (see Fig. 1).

Each source is connected to every single worker. In addition, every pair of workers are connected to each other. However, there is no communication link between the sources. In addition, all of the workers are connected to the master. All of these links are orthogonal¹, secure, and error free.

Each source $\gamma \in [\Gamma]$ has access to a matrix $\mathbf{X}^{[\gamma]}$, chosen from an arbitrary distribution over $\mathbb{F}^{m \times m}$ for some finite field \mathbb{F} and $m \in \mathbb{N}$. The master aims to know the result of a function $\mathbf{Y} = \mathbf{G}(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]})$, where $\mathbf{G}: (\mathbb{F}^{m \times m})^\Gamma \rightarrow \mathbb{F}^{m \times m}$ is an arbitrary polynomial function. The system operates in three phases: (i) sharing, (ii) computation and communication, and (iii) reconstruction. The detailed description of these phases is as follows.

- 1) *Sharing*: In this phase, the source γ sends $\tilde{\mathbf{X}}_{\gamma n} = \mathbf{F}_{\gamma \rightarrow n}(\mathbf{X}_\gamma)$ to worker n where $\mathbf{F}_{\gamma \rightarrow n}: \mathbb{F}^{m \times m} \rightarrow \mathbb{F}^{m \times \frac{m}{k}}$, for some $k \in \mathbb{N}$, $k|m$, denotes the sharing function, used at source $\gamma \in [\Gamma]$, to share data with worker $n \in [N]$. The number k represents the limit on the storage size at each worker.
- 2) *Computation and Communication*: In this phase, the workers process what they received, and in between, they may send some messages to other workers and continue processing. We define the set $\mathcal{M}_{n \rightarrow n'} \in \mathbb{F}^*$ as the set of all messages that worker n sends to the worker n' in this phase, for $n, n' \in [N]$.
- 3) *Reconstruction*: In this phase, every worker sends a message to the master. More precisely, worker n sends the message $\mathbf{O}_n \in \mathbb{F}^*$ to the master.

This scheme must satisfy three constraints.

- 1) *Correctness*: The master must be able to recover \mathbf{Y} from $\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N$. More precisely

$$H(\mathbf{Y} | \mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N) = 0, \quad (3)$$

¹By orthogonal link, we mean that they do not interfere with each other. For example, they can be wired links, or if they are wireless, they communicate at different time/frequency slots.

where H denotes the Shannon entropy.

- 2) *Privacy at the workers*: Let $t \in [N]$. Any arbitrary subset of workers, including $t - 1$ workers, must not gain any information about the inputs. In particular, for any $\mathcal{S} \subset [N]$, $|\mathcal{S}| \leq t - 1$

$$H(\mathbf{X}^{[j]}, j \in [\Gamma] | \bigcup_{n \in \mathcal{S}} \{\mathcal{M}_{n' \rightarrow n}, n' \in [N]\}, \tilde{\mathbf{X}}_{\gamma n}, \gamma \in [\Gamma], n \in \mathcal{S}) = H(\mathbf{X}^{[j]}, j \in [\Gamma]). \quad (4)$$

t is called *the security threshold* of the system. In other words, we assume that there are $t - 1$ semi-honest worker nodes among the workers. It means that even-though they follow the protocol and report any calculations correctly, they are curious about the input data and may collude to gain information about it. Condition (4) guarantees that those colluding servers gain no information about the private inputs.

- 3) *Privacy at the master*: The master must not gain any additional information about the inputs, beyond the result of the function. In other words, \mathbf{Y} is the only new information that is revealed to the master. More precisely,

$$H(\mathbf{X}^{[1]}, \mathbf{X}^{[2]} \dots, \mathbf{X}^{[\Gamma]} | \mathbf{Y}, \mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N) = H(\mathbf{X}^{[1]}, \mathbf{X}^{[2]} \dots, \mathbf{X}^{[\Gamma]} | \mathbf{Y}). \quad (5)$$

Definition 1. For some $k, t \in \mathbb{N}$ and polynomial function \mathbf{G} , we define $N_{\mathbf{G}}^*(t, k)$ as the minimum number of workers needed to calculate \mathbf{G} , while the correctness, privacy at the workers, and privacy at the master are satisfied.

The objective of this paper is to find an upper bound on $N_{\mathbf{G}}^*(t, k)$.

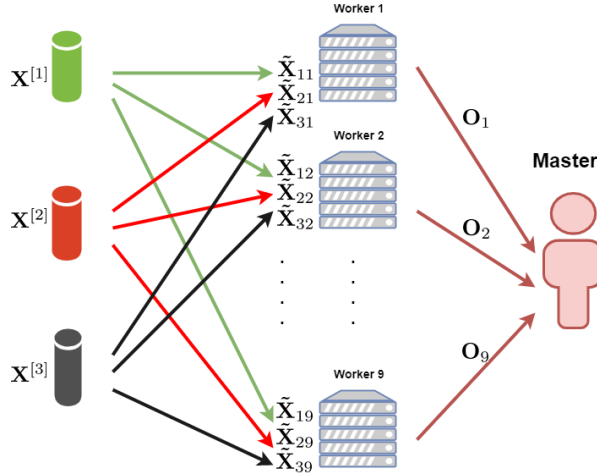


Fig. 1. An MPC system including $\Gamma = 3$ private inputs $\mathbf{X}^{[1]}, \mathbf{X}^{[2]}$, and $\mathbf{X}^{[3]} \in \mathbb{F}^{m \times m}$, for $m \in \mathbb{N}$, $N = 9$ workers, and a master. All communication links are secure and error free. The size of the shares given to each worker is a fraction of the size of the inputs. Input node γ sends $\tilde{\mathbf{X}}_{\gamma n} \in \mathbb{F}^{m \times \frac{m}{k}}$ to worker n , for some $k \in \mathbb{N}, k|m$. Workers process their inputs while interacting with each other. Finally worker n sends \mathbf{O}_n to the master. The master aims to know the result of a function e.g., $\mathbf{G}(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \mathbf{X}^{[3]}) = (\mathbf{X}^{[1]})^T \mathbf{X}^{[2]} + \mathbf{X}^{[3]}$, subject to the correctness condition (3) and privacy conditions (4), (5).

III. PRELIMINARIES: $k = 1$

As described in Section II, we have four constraints: limited storage size at each worker represented by k , the correctness of the result, and the privacy at the workers and master. If there is no storage limit at the workers, i.e., $k = 1$, the problem is reduced to a version of secure multi-party computation, which has been extensively studied in the literature. In Particular, in [4], Ben-Or, Goldwasser, and Wigderson propose a scheme referred as the BGW scheme. To be self contained, here we briefly describe the BGW scheme in several examples. As we will see having the limit of $k > 1$ will drastically change the problem.

Example 2 (The BGW scheme for addition). Assume that we have two sources 1 and 2 with private inputs $\mathbf{X}^{[1]} = \mathbf{A} \in \mathbb{F}^{m \times m}$ and $\mathbf{X}^{[2]} = \mathbf{B} \in \mathbb{F}^{m \times m}$, respectively. These sources share their inputs with the workers. There are at most $t - 1$ semi-honest adversaries among the workers, where $t \in [N]$, meaning that they follow the protocol but may collude to gain information about the input data.

The BGW protocol for this problem is as follows:

1) *Phase 1 - Sharing:*

First phase of the BGW scheme is based on Shamir secret sharing [8]. Source 1 forms the polynomial

$$\mathbf{F}_\mathbf{A}(x) = \mathbf{A} + \bar{\mathbf{A}}_1 x + \bar{\mathbf{A}}_2 x^2 + \cdots + \bar{\mathbf{A}}_{t-1} x^{t-1}, \quad (6)$$

where $\mathbf{F}_\mathbf{A}(0) = \mathbf{A}$ is the private input at source 1 and the other coefficients, $\bar{\mathbf{A}}_i, i \in \{1, 2, \dots, t-1\}$, are chosen from $\mathbb{F}^{m \times m}$ independently and uniformly at random. Source 1 uses $\mathbf{F}_\mathbf{A}(x)$ to share \mathbf{A} with the workers. This means that it sends $\mathbf{F}_\mathbf{A}(\alpha_n)$ to worker n , for some distinct $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$. We note that from Lagrange interpolation rule [34], if we have t points from $\mathbf{F}_\mathbf{A}(x)$ we can uniquely determine this polynomial [34]. Therefore, any subset of including t workers collaboratively can reconstruct $\mathbf{F}_\mathbf{A}(0) = \mathbf{A}$. Intuitively, the reason is that there are t unknown coefficients in $\mathbf{F}_\mathbf{A}(x)$ and thus, we need at least t equations to solve for those coefficients. Also, since coefficients, i.e., $\bar{\mathbf{A}}_i, i \in \{1, 2, \dots, t-1\}$, are chosen in $\mathbb{F}^{m \times m}$ independently and uniformly at random, then any subset of including less than t workers cannot reconstruct \mathbf{A} and indeed gain no information about it. For formal proof, see [8].

This approach is called (N, t) sharing of \mathbf{A} (or interchangeably (N, t) Shamir secret sharing of \mathbf{A}) and $\mathbf{F}_\mathbf{A}(\alpha_n)$ is called the share of \mathbf{A} at worker n .

Similarly, source 2 forms $\mathbf{F}_\mathbf{B}(x)$ according to the following equation and shares its secrets among the workers,

$$\mathbf{F}_\mathbf{B}(x) = \mathbf{B} + \bar{\mathbf{B}}_1 x + \bar{\mathbf{B}}_2 x^2 + \cdots + \bar{\mathbf{B}}_{t-1} x^{t-1}, \quad (7)$$

where $\mathbf{F}_\mathbf{B}(0) = \mathbf{B}$ is the private input at source 2 and $\bar{\mathbf{B}}_i, i \in \{1, 2, \dots, t-1\}$, are chosen in $\mathbb{F}^{m \times m}$ independently and uniformly at random. Source 2 sends $\mathbf{F}_\mathbf{B}(\alpha_n)$ to worker n .

2) *Phase 2 - Computation and Communication:*

Shamir secret sharing scheme has the linearity property, which is very important. It means that if we have the shared secrets \mathbf{A} and \mathbf{B} using $\mathbf{F}_\mathbf{A}(x)$ and $\mathbf{F}_\mathbf{B}(x)$, respectively among N workers, in order to share the secret $p\mathbf{A} + q\mathbf{B}$ among the workers, for some constants $p, q \in \mathbb{F}$, we just need that worker n locally calculates $p\mathbf{F}_\mathbf{A}(\alpha_n) + q\mathbf{F}_\mathbf{B}(\alpha_n)$. This implies that the share of $p\mathbf{A} + q\mathbf{B}$ are available at the workers using sharing polynomial $p\mathbf{F}_\mathbf{A}(x) + q\mathbf{F}_\mathbf{B}(x)$. For addition, it is sufficient to choose $p = q = 1$. Note that in this particular example, no communication among the nodes is needed.

3) *Phase 3 - Reconstruction:*

In this phase worker n sends $\mathbf{O}_n = \mathbf{F}_\mathbf{A}(\alpha_n) + \mathbf{F}_\mathbf{B}(\alpha_n)$, calculated in phase two, to the master. If the master has $\mathbf{F}_\mathbf{A}(\alpha_n) + \mathbf{F}_\mathbf{B}(\alpha_n)$ for t or more distinct α_n 's, it can recover all the coefficients of degree $t - 1$ polynomial $\mathbf{F}_\mathbf{A}(x) + \mathbf{F}_\mathbf{B}(x)$. In particular, it recovers $\mathbf{Y} = \mathbf{F}_\mathbf{A}(0) + \mathbf{F}_\mathbf{B}(0) = \mathbf{A} + \mathbf{B}$. One can verify that both privacy constraints (4) and (5) are satisfied.

◇

Example 3 (The BGW scheme for multiplication). Assume that we have two sources 1 and 2 with private inputs $\mathbf{X}^{[1]} = \mathbf{A} \in \mathbb{F}^{m \times m}$ and $\mathbf{X}^{[2]} = \mathbf{B} \in \mathbb{F}^{m \times m}$, respectively. In addition, the master aims to know $\mathbf{Y} = \mathbf{A}^T \mathbf{B}$. There are $t - 1$ semi-honest adversaries among the N workers.

The BGW protocol for this problem operates as follows:

1) *Phase 1 - Sharing:*

This phase is exactly the same as the first phase of Example 2. Therefore, at the end of this phase worker n has $\mathbf{F}_\mathbf{A}(\alpha_n)$ and $\mathbf{F}_\mathbf{B}(\alpha_n)$ for some distinct $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, where $\mathbf{F}_\mathbf{A}(x)$ and $\mathbf{F}_\mathbf{B}(x)$ are defined in (6) and (7), respectively.

2) Phase 2 - Computation and Communication:

In this phase, worker n calculates $\mathbf{F}_A^T(\alpha_n)\mathbf{F}_B(\alpha_n)$, simply by multiplying its shares of \mathbf{A} and \mathbf{B} . Let us define the polynomial $\mathbf{H}(x) \triangleq \mathbf{F}_A^T(x)\mathbf{F}_B(x)$. We note that $\mathbf{H}(0) = \mathbf{A}^T\mathbf{B}$ and $\deg(\mathbf{H}(x)) = 2t - 2$. Therefore, by having at least $2t - 1$ samples of this polynomial, $\mathbf{A}^T\mathbf{B}$ can be solved for. Therefore, if $N \geq 2t - 1$, and these workers send their result to the master, it can calculate $\mathbf{A}^T\mathbf{B}$. However, we do not do that. For some important reasons that we explain later, in Remark (1), we prefer to first have the Shamir sharing of $\mathbf{A}^T\mathbf{B}$ at each node. As explained before, different workers have samples of $\mathbf{F}_A^T(x)\mathbf{F}_B(x)$, and $\mathbf{A}^T\mathbf{B} = \mathbf{F}_A^T(0)\mathbf{F}_B(0)$. However, $\deg(\mathbf{F}_A^T(x)\mathbf{F}_B(x)) = 2t - 2 \neq t - 1$. In addition, the coefficients of $\mathbf{F}_A^T(x)\mathbf{F}_B(x)$ do not have the distribution that we wish in Shamir secret sharing. In the BGW scheme, to have Shamir shares of $\mathbf{A}^T\mathbf{B}$ at each node, we use the following approach. We note that from Lagrange interpolation rule [34], if $N \geq 2t - 1$, there exists a vector $\mathbf{r} = (r_1, r_2, \dots, r_N) \in \mathbb{F}^N$ such that

$$\mathbf{H}(0) = \mathbf{A}^T\mathbf{B} = \sum_{n=1}^N r_n \mathbf{H}(\alpha_n). \quad (8)$$

In this approach, worker n shares $\mathbf{H}(\alpha_n) = \mathbf{F}_A^T(\alpha_n)\mathbf{F}_B(\alpha_n)$ with other workers using Shamir secret sharing. In other words, worker n forms a polynomial of degree $t - 1$,

$$\mathbf{F}_n(x) = \mathbf{H}(\alpha_n) + \bar{\mathbf{H}}_1^{(n)}x + \bar{\mathbf{H}}_2^{(n)}x^2 + \dots + \bar{\mathbf{H}}_{t-1}^{(n)}x^{t-1},$$

where $\bar{\mathbf{H}}_i^{(n)}$, $i \in [t - 1]$, $n \in [N]$, are chosen independently and uniformly at random in $\mathbb{F}^{m \times m}$. Worker n sends the value of $\mathbf{F}_n(\alpha_{n'})$ to worker n' , for all $n, n' \in [N]$. Then each worker n calculates $\sum_{n'=1}^N r_{n'} \mathbf{F}_{n'}(\alpha_n)$. We claim that the result is indeed Shamir sharing of $\mathbf{A}^T\mathbf{B}$. To verify that, let us define $\mathbf{F}(x)$ as

$$\begin{aligned} \mathbf{F}(x) &\triangleq \sum_{n'=1}^N r_{n'} \mathbf{F}_{n'}(x) \\ &= \sum_{n'=1}^N r_{n'} \mathbf{H}(\alpha_{n'}) + x \sum_{n'=1}^N r_{n'} \bar{\mathbf{H}}_1^{(n')} + x^2 \sum_{n'=1}^N r_{n'} \bar{\mathbf{H}}_2^{(n')} + \dots + x^{t-1} \sum_{n'=1}^N r_{n'} \bar{\mathbf{H}}_{t-1}^{(n')}. \end{aligned} \quad (9)$$

Then, we have the following observations:

- (i) Due to (8), $\mathbf{F}(0) = \sum_{n'=1}^N r_{n'} \mathbf{H}(\alpha_{n'}) = \mathbf{A}^T\mathbf{B}$.
- (ii) Worker n has access to $\mathbf{F}(\alpha_n) = \sum_{n'=1}^N r_{n'} \mathbf{F}_{n'}(\alpha_n)$.
- (iii) $\sum_{n'=1}^N r_{n'} \bar{\mathbf{H}}_i^{(n')}$, $i \in [t - 1]$, are independent with uniform distribution in $\mathbb{F}^{m \times m}$.

Thus, $\mathbf{F}(\alpha_n)$ is indeed a Shamir share of $\mathbf{A}^T\mathbf{B}$. In addition, if t of the workers send their samples of $\mathbf{F}(x)$ to the master, it can recover $\mathbf{F}(0) = \mathbf{A}^T\mathbf{B}$.

3) Phase 3 - Reconstruction:

Each worker n sends $\mathbf{F}(\alpha_n)$ to the master. It can recover $\mathbf{F}(0) = \mathbf{A}^T\mathbf{B}$, if it has $\mathbf{F}(\alpha_n)$ from t workers.

We note that the master can also recover $\sum_{n'=1}^N r_{n'} \bar{\mathbf{H}}_i^{(n')}$, for $i \in [t - 1]$, which reveal no information about \mathbf{A} and \mathbf{B} . Therefore, the privacy at the master is guaranteed. In addition, whatever is shared with a worker is based on Shamir secret sharing with new random coefficients. This can be used to prove that the privacy at the workers is guaranteed.

This example shows that if the number of workers is $N \geq 2t - 1$, the system can calculate multiplication. \diamond

Remark 1: In this remark, we explain why we prefer to have Shamir shares of $\mathbf{A}^T\mathbf{B}$ at the workers. The main reason is that this approach gives us great flexibility to use it iteratively and calculate any polynomial function of the inputs. Let us assume that there are three inputs \mathbf{A} , \mathbf{B} , and \mathbf{C} , and the goal is

to calculate $\mathbf{C}^T \mathbf{A}^T \mathbf{B}$. First, we use the above approach to have the Shamir shares of $\mathbf{D} = \mathbf{A}^T \mathbf{B}$ at each worker and use it again to calculate $\mathbf{C}^T \mathbf{D}$. Similarly, if the goal is to calculate $\mathbf{A}^T \mathbf{B} + \mathbf{C}$, we use the above scheme to have the Shamir shares of $\mathbf{D} = \mathbf{A}^T \mathbf{B}$ at each node, and use the scheme of Example 2 to calculate $\mathbf{D} + \mathbf{C}$.

Another important reason is that to calculate $\mathbf{C}^T \mathbf{A}^T \mathbf{B}$, still we need $N = 2t - 1$ workers. Otherwise, $\mathbf{F}_\mathbf{C}^T(x) \mathbf{F}_\mathbf{A}^T(x) \mathbf{F}_\mathbf{B}(x)$ will have degree $3(t - 1)$, and thus we will need $N = 3t - 2$ workers. In other words, using this approach, the number of servers needed does not grow with the number of matrix multiplication.

This approach also makes the proof of privacy simpler.

Now let us consider the scenario where there is a storage limit for each worker, i.e., $k > 1$. One approach to deal with this case is to split the job into smaller jobs and use the BGW scheme for each sub-job, as we see in the following example.

Example 4 (*Concatenation of Job Splitting and the BGW, $k = 2$*). Here, we revisit Example 3, but here, we assume that $k = 2$.

We partition each matrix into two sub-matrices as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix}, \quad (10)$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 \end{bmatrix}, \quad (11)$$

where $\mathbf{A}_i, \mathbf{B}_i \in \mathbb{F}^{m \times \frac{m}{2}}$, for $i \in \{1, 2\}$. We note that

$$\mathbf{A}^T \mathbf{B} = \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_1 & \mathbf{A}_1^T \mathbf{B}_2 \\ \mathbf{A}_2^T \mathbf{B}_1 & \mathbf{A}_2^T \mathbf{B}_2 \end{bmatrix}.$$

Therefore, to calculate $\mathbf{A}^T \mathbf{B}$, we can use the BGW scheme with four groups of workers to calculate $\mathbf{A}_i^T \mathbf{B}_j, i, j \in \{1, 2\}$, each of them with at least $2t - 1$ workers, following Example 3. Therefore, using this scheme, we need $N = 4(2t - 1)$ workers.

◇

One can see that with the concatenation of job-splitting and the BGW scheme described in Example 4, the minimum number of workers needed to calculate the addition and multiplication of two matrices is $N = kt$ and $N = k^2(2t - 1)$, respectively. In this paper, we propose an algorithm which reduces the required number of workers significantly.

IV. MAIN RESULTS

The main result of this paper is as follows:

Theorem 1. *For any $k, t \in \mathbb{N}$ and any polynomial function \mathbf{G} ,*

$$N_{\mathbf{G}}^*(t, k) \leq \min\{2k^2 + 2t - 3, k^2 + kt + t - 2\},$$

where

$$\min\{2k^2 + 2t - 3, k^2 + kt + t - 2\} = \begin{cases} 2k^2 + 2t - 3 & \text{if } k < t \\ k^2 + kt + t - 2 & \text{if } k \geq t \end{cases}.$$

Remark 2: To prove Theorem 1, we propose a scheme which is based on a novel approach for sharing called *polynomial sharing* and some procedures to calculate basic functions such as addition and multiplication. Using these procedures iteratively, we can calculate any polynomial function.

Remark 3: Recall that concatenation of job splitting and MPC for multiplication needs $k^2(2t - 1)$ workers. However in the proposed scheme we can do multiplication with at most $\min\{2k^2 + 2t - 3, k^2 + kt + t - 2\}$ workers, which is an orderwise improvement. For example for $t = 200$ and $k = 16$, the proposed scheme needs $N = 909$ workers, while the job splitting approach needs $N = 102144$ workers.

Remark 4: Theorem 1 is about the cases where there is at least one matrix multiplication in the calculation of the function \mathbf{G} . If \mathbf{G} is a linear function of the inputs, then the proposed scheme needs $k + t - 1$ workers.

Remark 5: As mentioned in the introduction, secure matrix multiplication, which has been introduced in [26], concurrently by the conference paper of this manuscript [1], focuses on calculating the multiplication of two matrices. Ignoring the privacy constraint at the master, secure matrix multiplication can be considered as a special case of our proposed problem formulation. This line of work has been followed by [27], [28] to improve its efficiency. The number of workers needed by the scheme of [26] is equal to $(k + t - 1)^2$, which is outperformed by our proposed scheme (1). The number of workers needed by the follow-up paper [27] is equal to $k^2 + tk + t - 2$, which is again outperformed by the proposed scheme (1). Reference [28] reports two schemes, named as GAPS-Big and GAPS-Small. Indeed, GAPS-Big is the same as what we had already reported in [1]. The number of servers needed by GAPS-Small is smaller than what we report in this paper, for the case where $3 \leq t < k$. The results are summarized in Table I.

TABLE I
THE NUMBER OF WORKERS NEEDED FOR MATRIX MULTIPLICATION (SMM)

| Work | Date | Number of Workers Needed |
|---------------------------------------|-----------|---|
| Our Scheme (Nodehi-MaddahAli) [1] | June 2018 | $N = \begin{cases} 2k^2 + 2t - 3 & \text{if } k < t \\ k^2 + kt + t - 2 & \text{if } k \geq t \end{cases}$ |
| SMM (Chang-Tandon [26]) | June 2018 | $N = (k + t - 1)^2$ |
| SMM (Kakar et al. [27]) | Oct. 2018 | $N = k^2 + tk + t - 2$ |
| GASP-Big (D' Oliveira et. al. [28]) | Dec. 2018 | $N = \begin{cases} 2k^2 + 2t - 3 & \text{if } k < t \\ k^2 + kt + t - 2 & \text{if } k \geq t \end{cases}$ |
| GASP-Small (D' Oliveira et. al. [28]) | Dec. 2018 | $N = \begin{cases} k^2 + 2k & \text{if } 2 = t \leq k \\ k^2 + 2k + (t - 1)^2 + t - 4 & \text{if } 3 \leq t \leq k \\ k^2 + kt + 2t - 5 - \lfloor \frac{t-3}{k} \rfloor & \text{if } k < t \leq k(k - 1) + 2 \\ 2k^2 + kt + t - 2k - 1 & \text{if } k(k - 1) + 2 < t \end{cases}$ |

V. MOTIVATING EXAMPLE

Here, we revisit Example 4 with $k = 2$ and $t = 4$ and propose a solution that needs 13 workers, as compared to $4 \times 7 = 28$ workers in the solution of Example 4.

Example 5 (*The proposed Approach for $k = 2$ and $t = 4$*). In this example we explain the proposed scheme to securely calculate $\mathbf{L} = \mathbf{A}^T \mathbf{B}$.

1) Phase 1 - Sharing:

Consider the following *polynomial functions*

$$\begin{aligned} \mathbf{F}_A(x) &= \mathbf{A}_1 + \mathbf{A}_2 x + \bar{\mathbf{A}}_3 x^4 + \bar{\mathbf{A}}_4 x^5 + \bar{\mathbf{A}}_5 x^6, \\ \mathbf{F}_B(x) &= \mathbf{B}_1 + \mathbf{B}_2 x^2 + \bar{\mathbf{B}}_3 x^4 + \bar{\mathbf{B}}_4 x^5 + \bar{\mathbf{B}}_5 x^6, \end{aligned}$$

where $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1$, and \mathbf{B}_2 are defined in (10) and (11) in Example 4. In addition, $\bar{\mathbf{A}}_i, \bar{\mathbf{B}}_i$, for $i \in [3, 5]$, are matrices chosen independently and uniformly at random in $\mathbb{F}^{m \times \frac{m}{2}}$. These polynomials follow a certain pattern. More precisely, in these polynomials the coefficients of some powers of x are zero. In $\mathbf{F}_A(x)$ the coefficients of x^2 and x^3 are zero and in $\mathbf{F}_B(x)$ the coefficients of x and x^3 are zero. We choose $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, independently and uniformly at random. Source nodes 1 and 2 share $\mathbf{F}_A(\alpha_n)$ and $\mathbf{F}_B(\alpha_n)$ with worker n , respectively. We call this form of sharing as *polynomial sharing*.

2) Phase 2 - Computation and Communication:

Worker n calculates $\mathbf{F}_A^T(\alpha_n)\mathbf{F}_B(\alpha_n)$. Consider the polynomial function $\mathbf{H}(x)$ of degree 12, defined as

$$\mathbf{H}(x) = \sum_{n=0}^{12} \mathbf{H}_n x^n \triangleq \mathbf{F}_A^T(x)\mathbf{F}_B(x). \quad (12)$$

We note that

$$\begin{aligned} \mathbf{H}_0 &= \mathbf{A}_1^T \mathbf{B}_1, \\ \mathbf{H}_1 &= \mathbf{A}_2^T \mathbf{B}_1, \\ \mathbf{H}_2 &= \mathbf{A}_1^T \mathbf{B}_2, \\ \mathbf{H}_3 &= \mathbf{A}_2^T \mathbf{B}_2. \end{aligned} \quad (13)$$

If the master has $\mathbf{H}(\alpha_n)$ for $N \geq 13$ distinct α_i 's, then it can calculate all the coefficients of $\mathbf{H}(x)$, including $\mathbf{H}_0 = \mathbf{A}_1^T \mathbf{B}_1$, $\mathbf{H}_1 = \mathbf{A}_2^T \mathbf{B}_1$, $\mathbf{H}_2 = \mathbf{A}_1^T \mathbf{B}_2$, and $\mathbf{H}_3 = \mathbf{A}_2^T \mathbf{B}_2$, with probability approaching to one, as $|\mathbb{F}| \rightarrow \infty$. More precisely, according to Lagrange interpolation rule [34], there are some $r_n^{(i,j)}$, $i, j \in \{1, 2\}$ and $n \in [N]$, such that

$$\mathbf{A}_i^T \mathbf{B}_j = \sum_{n=1}^N r_n^{(i,j)} \mathbf{H}(\alpha_n). \quad (14)$$

Remark 6: Note that $r_n^{(i,j)}$, $i, j \in \{1, 2\}$ and $n \in [N]$ are only functions of α_n , $n \in [N]$, which are known by all workers.

Similar to Example 3 and following Remark 1, in order to prepare the scene for the next stage and cast the result of the local computation in the form of the *polynomial sharing*, we use the following steps. Worker n forms $\mathbf{Q}^{(n)}(x)$, defined as

$$\begin{aligned} \mathbf{Q}^{(n)}(x) &\triangleq \begin{bmatrix} r_n^{(1,1)} \mathbf{H}(\alpha_n) \\ r_n^{(2,1)} \mathbf{H}(\alpha_n) \end{bmatrix} + \begin{bmatrix} r_n^{(1,2)} \mathbf{H}(\alpha_n) \\ r_n^{(2,2)} \mathbf{H}(\alpha_n) \end{bmatrix} x^2 \\ &\quad + \mathbf{R}_0^{(n)} x^4 + \mathbf{R}_1^{(n)} x^5 + \mathbf{R}_2^{(n)} x^6, \end{aligned} \quad (15)$$

where $\mathbf{R}_i^{(n)}$, $i \in \{0, 1, 2\}$, are chosen independently and uniformly at random in $\mathbb{F}^{m \times \frac{m}{2}}$. Recall that $\mathbf{H}(\alpha_n)$, $r_n^{(1,1)}$, $r_n^{(1,2)}$, $r_n^{(2,1)}$ and $r_n^{(2,2)}$ are available at worker n . Thus, worker n has all the information to form $\mathbf{Q}^{(n)}(x)$. Then worker n sends $\mathbf{Q}^{(n)}(\alpha_{n'})$ to worker n' , for all $n' \in [N]$.

Thus, worker n' will have access to the matrices $\{\mathbf{Q}^{(1)}(\alpha_{n'}), \mathbf{Q}^{(2)}(\alpha_{n'}), \dots, \mathbf{Q}^{(N)}(\alpha_{n'})\}$. Then worker n' calculates $\sum_{n=1}^N \mathbf{Q}^{(n)}(\alpha_{n'})$. We claim that this summation is indeed the polynomial share of the matrix $\mathbf{A}^T \mathbf{B}$. To verify that, consider the polynomial function

$$\mathbf{Q}(x) \triangleq \sum_{n=1}^N \mathbf{Q}^{(n)}(x). \quad (16)$$

We note that

$$\begin{aligned} \mathbf{Q}(x) &\triangleq \sum_{n=1}^N \mathbf{Q}^{(n)}(x) \\ &= \sum_{n=1}^N \begin{bmatrix} r_n^{(1,1)} \mathbf{H}(\alpha_n) \\ r_n^{(2,1)} \mathbf{H}(\alpha_n) \end{bmatrix} + x^2 \sum_{n=1}^N \begin{bmatrix} r_n^{(1,2)} \mathbf{H}(\alpha_n) \\ r_n^{(2,2)} \mathbf{H}(\alpha_n) \end{bmatrix} + x^4 \sum_{n=1}^N \mathbf{R}_0^{(n)} + x^5 \sum_{n=1}^N \mathbf{R}_1^{(n)} + x^6 \sum_{n=1}^N \mathbf{R}_2^{(n)} \\ &\stackrel{(a)}{=} \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_1 \\ \mathbf{A}_2^T \mathbf{B}_1 \end{bmatrix} + x^2 \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_2 \\ \mathbf{A}_2^T \mathbf{B}_2 \end{bmatrix} + x^4 \sum_{n=1}^N \mathbf{R}_0^{(n)} + x^5 \sum_{n=1}^N \mathbf{R}_1^{(n)} + x^6 \sum_{n=1}^N \mathbf{R}_2^{(n)} \\ &= \mathbf{L}_1 + \mathbf{L}_2 x^2 + \bar{\mathbf{L}}_3 x^4 + \bar{\mathbf{L}}_4 x^5 + \bar{\mathbf{L}}_5 x^6, \end{aligned} \quad (17)$$

where (a) follows from (14) and $\mathbf{L}_1 \triangleq \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_1 \\ \mathbf{A}_2^T \mathbf{B}_1 \end{bmatrix}$, $\mathbf{L}_2 \triangleq \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_2 \\ \mathbf{A}_2^T \mathbf{B}_2 \end{bmatrix}$, $\bar{\mathbf{L}}_3 \triangleq \sum_{n=1}^N \mathbf{R}_0^{(n)}$, $\bar{\mathbf{L}}_4 \triangleq \sum_{n=1}^N \mathbf{R}_1^{(n)}$, and $\bar{\mathbf{L}}_5 \triangleq \sum_{n=1}^N \mathbf{R}_2^{(n)}$. Note that, $\bar{\mathbf{L}}_i, i \in \{3, 4, 5\}$, have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{2}}$. Because of the randomness of $\mathbf{R}_i^{(n)}(x)$, for $i \in \{0, 1, 2\}$, $n \in [N]$, and degree of $\mathbf{Q}(x)$, one can easily verify that $\mathbf{Q}(x)$ is in the form of polynomial sharing and worker n has access to $\mathbf{Q}(\alpha_n)$.

3) *Phase3 - Reconstruction:*

Worker n sends $\mathbf{Q}(\alpha_n)$ to the master, $n \in [N]$. The master then calculates $\mathbf{A}^T \mathbf{B}$ from what it receives by polynomial interpolation.

◇

This algorithm guarantees conditions (4), the privacy at the workers. An intuitive explanation is that in each interaction among the workers, the algorithm uses some independent random matrices, as the coefficients of the sharing polynomials, such that any subset, including 3 workers, cannot gain any information about the input data. This will be proven formally later in Appendix B for the general cases. Also there is no information leakage at the master, as required according to the privacy constraints (5), which will be formally proven in Appendix B.

Example 6. In this example, we aim to show how to use a new approach to securely calculate $\mathbf{M} = \mathbf{A}^T \mathbf{B} + \mathbf{C}$, for $k = 2$ and $t = 4$.

1) *Phase 1 - Sharing:*

Consider the following *polynomial functions*

$$\begin{aligned} \mathbf{F}_\mathbf{A}(x) &= \mathbf{A}_1 + \mathbf{A}_2 x + \bar{\mathbf{A}}_3 x^4 + \bar{\mathbf{A}}_4 x^5 + \bar{\mathbf{A}}_5 x^6, \\ \mathbf{F}_\mathbf{B}(x) &= \mathbf{B}_1 + \mathbf{B}_2 x^2 + \bar{\mathbf{B}}_3 x^4 + \mathbf{B}_4 x^5 + \bar{\mathbf{B}}_5 x^6, \\ \mathbf{F}_\mathbf{C}(x) &= \mathbf{C}_1 + \mathbf{C}_2 x^2 + \bar{\mathbf{C}}_3 x^4 + \bar{\mathbf{C}}_4 x^5 + \bar{\mathbf{C}}_5 x^6, \end{aligned}$$

where in the above equations, $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2, \mathbf{C}_1$, and \mathbf{C}_2 are defined similar to (10). In addition, $\bar{\mathbf{A}}_i, \bar{\mathbf{B}}_i, \bar{\mathbf{C}}_i$, for $i \in [3, 5]$, are matrices chosen independently and uniformly at random in $\mathbb{F}^{m \times \frac{m}{2}}$. Similar to Example 5, source nodes 1, 2, and 3 share $\mathbf{F}_\mathbf{A}(\alpha_n)$, $\mathbf{F}_\mathbf{B}(\alpha_n)$, and $\mathbf{F}_\mathbf{C}(\alpha_n)$ with worker n .

2) *Phase 2 - Computation and Communication:* In this phase, first, all of the workers follow phase 2 of Example 5, in order to have access to $\mathbf{Q}(\alpha_n)$ defined in (16), as the polynomial shares of $\mathbf{A}^T \mathbf{B}$. Then each worker calculates $\mathbf{O}_n = \mathbf{Q}(\alpha_n) + \mathbf{F}_\mathbf{C}(\alpha_n)$.

3) *Phase3 - Reconstruction:*

Worker n sends \mathbf{O}_n to the master, $n \in [N]$. The master then calculates $\mathbf{A}^T \mathbf{B} + \mathbf{C}$ from what it receives by polynomial interpolation.

◇

Remark 7: We would like to emphasize that recasting the result of each round of computing in the form of a polynomial-sharing is a very crucial step for us and has several important advantages:

- 1) This approach gives us great flexibility to use the procedures iteratively and calculate any polynomial function of the inputs. Let us assume that there are three inputs \mathbf{A}, \mathbf{B} , and \mathbf{C} , and the goal is to calculate $\mathbf{C}^T \mathbf{A}^T \mathbf{B}$. First, we use the multiplication procedure to have the shares of $\mathbf{D} = \mathbf{A}^T \mathbf{B}$. At the end of the multiplication procedure, the workers have the polynomial shares of \mathbf{D} (instead of the multiplications of the shares of \mathbf{A} and \mathbf{B}). The workers are also given the polynomial shares of \mathbf{C} . Thus we can use the multiplication procedure again to calculate $\mathbf{C}^T \mathbf{D}$. Similarly, if the goal is to calculate $\mathbf{A}^T \mathbf{B} + \mathbf{C}$ (see Example 6), we use the multiplication procedure to have the polynomial shares of $\mathbf{D} = \mathbf{A}^T \mathbf{B}$ at each node, and use the addition procedure to calculate $\mathbf{D} + \mathbf{C}$.
- 2) It allows us to develop a scheme, where the number of workers needed does NOT grow with the degree of the polynomial function $\mathbf{G}(\cdot)$. Again assume that the goal is to calculate $\mathbf{C}^T \mathbf{A}^T \mathbf{B}$. Reviewing the argument above, one can confirm that the number of workers needed to calculate $\mathbf{C}^T \mathbf{A}^T \mathbf{B}$ is the same as the number of workers needed to calculate $\mathbf{A}^T \mathbf{B}$.

- 3) It simplifies the proof of privacy, considering the fact that we may have several rounds of computation. Reviewing the proof of privacy in Appendix B of the paper, we observe that recasting the result of each round in the form of polynomial sharing helps us develop an iterative argument and establish that the information leakage is zero. Without that, it would be very hard to prove the privacy, considering all the interaction among the workers to keep the number of workers constant.

VI. POLYNOMIAL SHARING

In this section, we formally define a sharing scheme, called *polynomial sharing*, and in the next section, we show that it admits basic operations such as addition, multiplication of matrices, and transposing a matrix. By concatenating the procedures for basic operations, we show that any polynomial function of the input data can be calculated, subject to the problem constraints. This scheme is motivated by [13], which is a coding technique for matrix multiplication in the distributed system with stragglers.

Definition 2. Let $\mathbf{A} \in \mathbb{F}^{m \times m}$, partitioned as

$$\mathbf{A} \triangleq [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k], \quad (18)$$

where $\mathbf{A}_j \in \mathbb{F}^{m \times \frac{m}{k}}$, for some $k \in \mathbb{N}$, $k|m$, and $j \in [k]$. The polynomial function $\mathbf{F}_{\mathbf{A},b,t,k}(x)$, for some $b \in [k]$, is defined as

$$\mathbf{F}_{\mathbf{A},b,t,k}(x) \triangleq \sum_{j=1}^k \mathbf{A}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1}, \quad (19)$$

where $\mathbf{A}_j, j = 1, 2, \dots, k$, are defined in (18) and $\mathbf{R}_j, j = 1, 2, \dots, t-1$, are chosen independently and uniformly at random from $\mathbb{F}^{m \times \frac{m}{k}}$. We say that matrix \mathbf{A} is (b, t, k) polynomial-shared with workers in $[N]$, if $\mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n)$ is sent to worker n , where $\alpha_n \in \mathbb{F}$ are distinct constants assigned to worker n , $n \in [N]$, where $k + t - 1 < N$.

Remark 8: Note that for $b = 1$ and $k = 1$, the polynomial sharing is reduced to Shamir secret sharing.

The following theorem about polynomial sharing will be essential in the following sections.

Theorem 2. Let $t, k, m \in \mathbb{N}$, $N \geq \min\{2k^2 + 2t - 3, k^2 + tk + t - 2\}$, $k|m$, and $\mathbf{A}, \mathbf{B} \in \mathbb{F}^{m \times m}$. Define

$$\mathbf{H}(x) \triangleq \mathbf{F}_{\mathbf{A},1,t,k}^T(x) \mathbf{F}_{\mathbf{B},k,t,k}(x).$$

Then for some large enough $|\mathbb{F}|$, there exist distinct $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, such that for any distinct $i_1, i_2, \dots, i_{k+t-1} \in [N]$, we have

$$H(\mathbf{A} | \mathbf{F}_{\mathbf{A},1,t,k}(\alpha_{i_1}), \mathbf{F}_{\mathbf{A},1,t,k}(\alpha_{i_2}), \dots, \mathbf{F}_{\mathbf{A},1,t,k}(\alpha_{i_{k+t-1}})) = 0, \quad (20)$$

$$H(\mathbf{B} | \mathbf{F}_{\mathbf{B},k,t,k}(\alpha_{i_1}), \mathbf{F}_{\mathbf{B},k,t,k}(\alpha_{i_2}), \dots, \mathbf{F}_{\mathbf{B},k,t,k}(\alpha_{i_{k+t-1}})) = 0, \quad (21)$$

and

$$H(\mathbf{A}^T \mathbf{B} | \mathbf{H}(\alpha_1), \mathbf{H}(\alpha_2), \dots, \mathbf{H}(\alpha_N)) = 0. \quad (22)$$

In addition, if we choose $\alpha_1, \alpha_2, \dots, \alpha_N$, independently and uniformly at random in \mathbb{F} , the probability that (20), (21), and (22) hold, approaches to one, as $|\mathbb{F}| \rightarrow \infty$.

Proof. See Appendix A. □

VII. PROCEDURES

In this section, we explain several procedures to do basic operations such as addition, multiplication, and transposing, using polynomial sharing without leaking any information. By concatenating these procedures, we can calculate any polynomial function of the private input data, subject to the constraints (3), (4), and (5).

A. Addition

Let $\mathbf{A}, \mathbf{B} \in \mathbb{F}^{m \times m}$, and

$$\begin{aligned}\mathbf{A} &= [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k], \\ \mathbf{B} &= [\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_k],\end{aligned}$$

where $\mathbf{A}_i, \mathbf{B}_i \in \mathbb{F}^{m \times \frac{m}{k}}$, for $i \in [k]$ and $k|m$. In addition, assume \mathbf{A} and \mathbf{B} are (b, t, k) polynomial-shared with workers in $[N]$. The objective is to (b, t, k) polynomial-share $\mathbf{L} \triangleq \mathbf{A} + \mathbf{B}$ with workers in $[N]$. To do that we follow a one-step procedure, as follows:

1) Step 1- Computation:

In this step, worker n calculates $\mathbf{F}_{\mathbf{A}, b, t, k}(\alpha_n) + \mathbf{F}_{\mathbf{B}, b, t, k}(\alpha_n)$. We claim that this summation is indeed the polynomial share of the matrix $\mathbf{A} + \mathbf{B}$. To verify that, consider the polynomial function

$$\mathbf{Q}(x) \triangleq \mathbf{F}_{\mathbf{A}, b, t, k}(x) + \mathbf{F}_{\mathbf{B}, b, t, k}(x). \quad (23)$$

We note that

$$\begin{aligned}\mathbf{Q}(x) &= \mathbf{F}_{\mathbf{A}, b, t, k}(x) + \mathbf{F}_{\mathbf{B}, b, t, k}(x) \\ &= \sum_{j=1}^k \mathbf{A}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j x^{k^2+j-1} + \sum_{j=1}^k \mathbf{B}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \hat{\mathbf{R}}_j x^{k^2+j-1} \\ &= \sum_{j=1}^k (\mathbf{A}_j + \mathbf{B}_j) x^{b(j-1)} + \sum_{j=1}^{t-1} (\bar{\mathbf{R}}_j + \hat{\mathbf{R}}_j) x^{k^2+j-1} \\ &\stackrel{(a)}{=} \sum_{j=1}^k \mathbf{L}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1},\end{aligned} \quad (24)$$

where (a) follows from the definitions $\mathbf{L}_j \triangleq \mathbf{A}_j + \mathbf{B}_j$ for $j \in [k]$, and $\mathbf{R}_j \triangleq \bar{\mathbf{R}}_j + \hat{\mathbf{R}}_j$, for $j \in [t-1]$. We note that $\mathbf{L} = [\mathbf{L}_1 \ \mathbf{L}_2 \ \dots \ \mathbf{L}_k]$. In addition, $\mathbf{R}_j, j \in [t-1]$, have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$. Thus, $\mathbf{Q}(x)$ is in the form of (b, t, k) polynomial sharing of $\mathbf{A} + \mathbf{B}$ with workers in $[N]$, and worker n has access to $\mathbf{Q}(\alpha_n)$. This step is detailed in Algorithm 1.

Algorithm 1 Addition

Inputs: Matrices $\mathbf{A}, \mathbf{B} \in \mathbb{F}^{m \times m}$ which are (b, t, k) polynomial-shared with workers in $[N]$.

- 1: Worker n adds its shares of \mathbf{A} and \mathbf{B} .
 - 2: End.
-

B. Multiplication by Constant

Let $\mathbf{A} \in \mathbb{F}^{m \times m}$, and

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k],$$

for $\mathbf{A}_i \in \mathbb{F}^{m \times \frac{m}{k}}$, $i \in [k]$, and $k|m$. In addition, assume \mathbf{A} is (b, t, k) polynomial-shared with workers in $[N]$. The objective is to (b, t, k) polynomial-share $\mathbf{L} \triangleq q\mathbf{A}$ with workers in $[N]$, where q is a constant in \mathbb{F} . To do that, we follow a one-step procedure, as follows:

1) Step 1- Computation:

In this step, worker n calculates $q\mathbf{F}_{\mathbf{A}, b, t, k}(\alpha_n)$. We claim that this multiplication is indeed the polynomial share of the matrix $q\mathbf{A}$. To verify that, consider the polynomial function

$$\mathbf{Q}(x) \triangleq q\mathbf{F}_{\mathbf{A}, b, t, k}(x). \quad (25)$$

We note that

$$\begin{aligned}
\mathbf{Q}(x) &= q\mathbf{F}_{\mathbf{A},b,t,k}(x) \\
&= q \sum_{j=1}^k \mathbf{A}_j x^{b(j-1)} + q \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j x^{k^2+j-1} \\
&= \sum_{j=1}^k q\mathbf{A}_j x^{b(j-1)} + \sum_{j=1}^{t-1} q\bar{\mathbf{R}}_j x^{k^2+j-1} \\
&\stackrel{(a)}{=} \sum_{j=1}^k \mathbf{L}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1},
\end{aligned} \tag{26}$$

where (a) follows from the definitions $\mathbf{L}_j \triangleq q\mathbf{A}_j$, for $j \in [k]$, and $\mathbf{R}_j \triangleq q\bar{\mathbf{R}}_j$, for $j \in [t-1]$. We note that $\mathbf{L} = [\mathbf{L}_1 \ \mathbf{L}_2 \ \dots \ \mathbf{L}_k]$. In addition, $\mathbf{R}_j, j \in [t-1]$, have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$. Thus, $\mathbf{Q}(x)$ is in the form of (b, t, k) polynomial sharing of $q\mathbf{A}$ with workers in $[N]$, and worker n has access to $\mathbf{Q}(\alpha_n)$. This step is detailed in Algorithm 2.

Algorithm 2 Multiplication by Constant

Inputs: Matrix $\mathbf{A} \in \mathbb{F}^{m \times m}$ which is (b, t, k) polynomial-shared with workers in $[N]$, and $q \in \mathbb{F}$.

- 1: Worker n multiplies its share of \mathbf{A} by q .
 - 2: End.
-

Remark 9: Polynomial sharing scheme has the linearity property. It means that if matrices \mathbf{A} and \mathbf{B} are (b, t, k) polynomial-shared with workers in $[N]$, in order to (b, t, k) polynomial-share $\mathbf{L} = q\mathbf{A} + p\mathbf{B}$ with workers in $[N]$, it is enough that each worker just locally calculates the same computation on its shares.

C. Multiplication of Two Matrices

Similar to the Shamir secret sharing scheme, calculating the shares of the multiplication of two matrices is not as simple as calculating the addition. Workers need to do some communication. In what follows, we explain the procedure.

Let us assume that $\mathbf{A}, \mathbf{B} \in \mathbb{F}^{m \times m}$, and

$$\begin{aligned}
\mathbf{A} &= [\mathbf{A}_1, \ \mathbf{A}_2, \ \dots, \ \mathbf{A}_k], \\
\mathbf{B} &= [\mathbf{B}_1, \ \mathbf{B}_2, \ \dots, \ \mathbf{B}_k],
\end{aligned}$$

where $\mathbf{A}_i, \mathbf{B}_i \in \mathbb{F}^{m \times \frac{m}{k}}$, for $i \in [k]$ and $k|m$. In addition, assume \mathbf{A} and \mathbf{B} are $(1, t, k)$ and (k, t, k) polynomial-shared with workers in $[N]$, respectively. The goal is to (b, t, k) polynomial-share $\mathbf{L} \triangleq \mathbf{A}^T \mathbf{B}$ with workers in $[N]$. We follow three steps, including computation, communication, and aggregation.

1) Step 1- Computation:

Worker n calculates $\mathbf{F}_{\mathbf{A},1,t,k}^T(\alpha_n) \mathbf{F}_{\mathbf{B},k,t,k}(\alpha_n)$.

Consider the polynomial function $\mathbf{H}(x)$ of degree $2(k^2 + t - 2)$, defined as,

$$\mathbf{H}(x) = \sum_{n=0}^{2(k^2+t-2)} \mathbf{H}_n x^n \triangleq \mathbf{F}_{\mathbf{A},1,t,k}^T(x) \mathbf{F}_{\mathbf{B},k,t,k}(x). \tag{27}$$

It is important to note that

$$\mathbf{H}_{i-1+k(j-1)} = \mathbf{A}_i^T \mathbf{B}_j, \tag{28}$$

for $i, j \in [k]$.

According to (22), if $N \geq \min\{2k^2 + 2t - 3, k^2 + tk + t - 2\}$, then with probability approaching to one, as $|\mathbb{F}| \rightarrow \infty$, we can calculate all the coefficients of $\mathbf{H}(x)$, including $\mathbf{H}_{i-1+k(j-1)} = \mathbf{A}_i^T \mathbf{B}_j$, for $i, j \in [k]$, from $\mathbf{H}(\alpha_n)$, $n \in [N]$. In particular, there are some $r_n^{(i,j)}$, $i, j \in [k]$ and $n \in [N]$, such that

$$\mathbf{A}_i^T \mathbf{B}_j = \sum_{n=1}^N r_n^{(i,j)} \mathbf{H}(\alpha_n). \quad (29)$$

Note that $r_n^{(i,j)}$, $i, j \in [k]$ and $n \in [N]$ are only functions of α_n , $n \in [N]$, which are known by all workers. Up to now worker n has access to $\mathbf{H}(\alpha_n)$. The challenge is to find a way to change the local knowledge of the $\mathbf{H}(\alpha_n)$ to (b, t, k) polynomial-share of \mathbf{L} , for each worker n .

2) *Step 2- Communication:*

Worker n forms the matrix $\mathbf{H}^{(n)}$, defined as

$$\mathbf{H}^{(n)} \triangleq \begin{bmatrix} \mathbf{H}(\alpha_n) r_n^{(1,1)} & \mathbf{H}(\alpha_n) r_n^{(1,2)} & \dots & \mathbf{H}(\alpha_n) r_n^{(1,k)} \\ \mathbf{H}(\alpha_n) r_n^{(2,1)} & \mathbf{H}(\alpha_n) r_n^{(2,2)} & \dots & \mathbf{H}(\alpha_n) r_n^{(2,k)} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{H}(\alpha_n) r_n^{(k,1)} & \mathbf{H}(\alpha_n) r_n^{(k,2)} & \dots & \mathbf{H}(\alpha_n) r_n^{(k,k)} \end{bmatrix}. \quad (30)$$

Then worker n , (b, t, k) polynomial-shares $\mathbf{H}^{(n)}$ with workers in $[N]$. More precisely, according to (19), worker n forms the following polynomial,

$$\mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(x) = \sum_{j=1}^k \mathbf{H}_j^{(n)} x^{b(j-1)} + \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1}, \quad (31)$$

where $\mathbf{H}_j^{(n)} \triangleq \begin{bmatrix} \mathbf{H}(\alpha_n) r_n^{(1,j)} \\ \mathbf{H}(\alpha_n) r_n^{(2,j)} \\ \vdots \\ \mathbf{H}(\alpha_n) r_n^{(k,j)} \end{bmatrix}$, for $j \in [k]$ and $\bar{\mathbf{R}}_j^{(n)}$, $j = 1, 2, \dots, t-1$, are chosen independently and

uniformly at random in $\mathbb{F}^{m \times \frac{m}{k}}$. Worker n sends $\mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(\alpha_{n'})$ to the worker n' , for all $n' \in [N]$. All of the workers follow the same method. Thus, at the end of this step, each worker n' has access to the matrices $\{\mathbf{F}_{\mathbf{H}^{(1)}, b, t, k}(\alpha_{n'}), \mathbf{F}_{\mathbf{H}^{(2)}, b, t, k}(\alpha_{n'}), \dots, \mathbf{F}_{\mathbf{H}^{(N)}, b, t, k}(\alpha_{n'})\}$.

3) *Step 3- Aggregation:*

Now worker n' calculates $\sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(\alpha_{n'})$. We claim that this summation is indeed the polynomial share of the matrix $\mathbf{A}^T \mathbf{B}$. To verify that, consider the polynomial function

$$\mathbf{Q}(x) \triangleq \sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(x). \quad (32)$$

We note that

$$\begin{aligned} \mathbf{Q}(x) &= \sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(x) \\ &= \sum_{n=1}^N \sum_{j=1}^k \mathbf{H}_j^{(n)} x^{b(j-1)} + \sum_{n=1}^N \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &= \sum_{n=1}^N \sum_{j=1}^k \begin{bmatrix} \mathbf{H}(\alpha_n) r_n^{(1,j)} \\ \mathbf{H}(\alpha_n) r_n^{(2,j)} \\ \vdots \\ \mathbf{H}(\alpha_n) r_n^{(k,j)} \end{bmatrix} x^{b(j-1)} + \sum_{n=1}^N \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^k \sum_{n=1}^N \begin{bmatrix} \mathbf{H}(\alpha_n) r_n^{(1,j)} \\ \mathbf{H}(\alpha_n) r_n^{(2,j)} \\ \vdots \\ \mathbf{H}(\alpha_n) r_n^{(k,j)} \end{bmatrix} x^{b(j-1)} + \sum_{j=1}^{t-1} \sum_{n=1}^N \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\
&\stackrel{(a)}{=} \sum_{j=1}^k \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_j \\ \mathbf{A}_2^T \mathbf{B}_j \\ \vdots \\ \mathbf{A}_k^T \mathbf{B}_j \end{bmatrix} x^{b(j-1)} + \sum_{j=1}^{t-1} \left(\sum_{n=1}^N \bar{\mathbf{R}}_j^{(n)} \right) x^{k^2+j-1} = \\
&\stackrel{(b)}{=} \sum_{j=1}^k \mathbf{L}_j x^{b(j-1)} + \mathbf{R}_j x^{k^2+j-1}, \tag{33}
\end{aligned}$$

where (a) follows from (29), and (b) follows from the definitions $\mathbf{L}_j \triangleq \begin{bmatrix} \mathbf{A}_1^T \mathbf{B}_j \\ \mathbf{A}_2^T \mathbf{B}_j \\ \vdots \\ \mathbf{A}_k^T \mathbf{B}_j \end{bmatrix}$, for $j \in [k]$ and

$\mathbf{R}_j \triangleq \sum_{n=1}^N \bar{\mathbf{R}}_j^{(n)}$, for $j \in [t-1]$. We note that $\mathbf{L} = [\mathbf{L}_1 \ \mathbf{L}_2 \ \dots \ \mathbf{L}_k]$. In addition, $\mathbf{R}_j, j \in [t-1]$, have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$. Thus, $\mathbf{Q}(x)$ is in the form of (b, t, k) polynomial sharing of $\mathbf{A}^T \mathbf{B}$ with workers in $[N]$, and worker n has access to $\mathbf{Q}(\alpha_n)$. These steps are detailed in Algorithm 3.

Algorithm 3 Multiplication of Two Matrices

Inputs: Number N of the workers. Matrices \mathbf{A}, \mathbf{B} which are $(1, t, k)$ and (k, t, k) polynomial-shared with workers in $[N]$, respectively. Also $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, $r_n^{(i,j)}$, for $i, j \in [k]$, and $n \in [N]$ are known by all workers.

Step 1- Computation:

- 1: Worker n calculates $\mathbf{H}(\alpha_n) \triangleq \mathbf{F}_{\mathbf{A},1,t,k}^T(\alpha_n) \mathbf{F}_{\mathbf{B},k,t,k}(\alpha_n)$.

Step 2- Communication:

- 2: Worker n , (b, t, k) polynomial-shares the matrix

$$\mathbf{H}^{(n)} \triangleq \begin{bmatrix} \mathbf{H}(\alpha_n) r_n^{(1,1)} & \mathbf{H}(\alpha_n) r_n^{(1,2)} & \dots & \mathbf{H}(\alpha_n) r_n^{(1,k)} \\ \mathbf{H}(\alpha_n) r_n^{(2,1)} & \mathbf{H}(\alpha_n) r_n^{(2,2)} & \dots & \mathbf{H}(\alpha_n) r_n^{(2,k)} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{H}(\alpha_n) r_n^{(k,1)} & \mathbf{H}(\alpha_n) r_n^{(k,2)} & \dots & \mathbf{H}(\alpha_n) r_n^{(k,k)} \end{bmatrix},$$

with workers in $[N]$.

Step 3- Aggregation:

- 3: Worker n calculates the sum of the messages received in the last step.
 - 4: End.
-

D. Transposing

Let $\mathbf{A} \in \mathbb{F}^{m \times m}$, and

$$\mathbf{A} = [\mathbf{A}_1, \ \mathbf{A}_2, \ \dots, \ \mathbf{A}_k],$$

where $\mathbf{A}_i \in \mathbb{F}^{m \times \frac{m}{k}}$, for $i \in [k]$ and $k|m$. In addition, assume \mathbf{A} is (b, t, k) polynomial-shared with workers in $[N]$. The goal is to (b, t, k) polynomial-share $\mathbf{L} \triangleq \mathbf{A}^T$ with workers in $[N]$. We follow three steps, including splitting, communication, and aggregation.

1) *Step 1- Splitting:*

Let us define

$$\mathbf{F}_i(x) \triangleq \mathbf{F}_{\mathbf{A},b,t,k}(x) \left(\frac{m}{k}(i-1) : \frac{m}{k}i, : \right). \quad (34)$$

In other words, $\mathbf{F}_i(x)$, $i \in [k]$, is a sub-matrix of $\mathbf{F}_{\mathbf{A},b,t,k}(x)$, including rows from $\frac{m}{k}(i-1)$ to $\frac{m}{k}i$. One can see that we have

$$\mathbf{F}_i(x) = \sum_{j=1}^k \mathbf{A}_{ij} x^{b(j-1)} + \sum_{j=1}^{t-1} \bar{\mathbf{R}}_{ij} x^{k^2+j-1}, \quad (35)$$

where

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \dots & \mathbf{A}_{1k} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \dots & \mathbf{A}_{2k} \\ \vdots & \vdots & \dots & \vdots \\ \mathbf{A}_{k1} & \mathbf{A}_{k2} & \dots & \mathbf{A}_{kk} \end{bmatrix},$$

and $\mathbf{A}_{ij} \in \mathbb{F}^{\frac{m}{k} \times \frac{m}{k}}$, for $i, j \in [k]$ and according to (19), $\bar{\mathbf{R}}_{ij}$, for $i, j \in [k]$, are independently and uniformly distributed in $\mathbb{F}^{\frac{m}{k} \times \frac{m}{k}}$. Since each worker n has access to $\mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n)$, it has access to $\mathbf{F}_i(\alpha_n)$, too. Similar to (20) and (21), it can be proven that if $N \geq (k+t-1)$, then by having $\mathbf{F}_i(\alpha_n)$, for $n = 1, 2, \dots, N$, we can calculate all the coefficients of $\mathbf{F}_i(x)$, including \mathbf{A}_{ij} , for $j \in [k]$, with probability approaching to one, as $|\mathbb{F}| \rightarrow \infty$. More precisely, similar to (20) and (21), it can be proven that for any distinct $i_1, i_2, \dots, i_{k+t-1} \in [N]$, independently and uniformly at random chosen parameters $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, and $i, j \in [k]$, with probability approaching to one, as $|\mathbb{F}| \rightarrow \infty$ we have

$$H(\mathbf{A}_{ij} | \mathbf{F}_i(\alpha_{i_1}), \mathbf{F}_i(\alpha_{i_2}), \dots, \mathbf{F}_i(\alpha_{i_{k+t-1}})) = 0.$$

In particular, there are some $r_n^{(j)}$, such that for $i, j \in [k]$, and $n \in [N]$,

$$\mathbf{A}_{ij} = \sum_{n=1}^N r_n^{(j)} \mathbf{F}_i(\alpha_n). \quad (36)$$

Note that $r_n^{(j)}$, $j \in [k]$ and $n \in [N]$, are only functions of α_n , $n \in [N]$, which are known by all workers.

2) *Step 2- Communication:*

Worker n forms the matrix $\mathbf{H}^{(n)}$ defined as

$$\mathbf{H}^{(n)} \triangleq \begin{bmatrix} \mathbf{F}_1(\alpha_n) r_n^{(1)} & \mathbf{F}_2(\alpha_n) r_n^{(1)} & \dots & \mathbf{F}_k(\alpha_n) r_n^{(1)} \\ \mathbf{F}_1(\alpha_n) r_n^{(2)} & \mathbf{F}_2(\alpha_n) r_n^{(2)} & \dots & \mathbf{F}_k(\alpha_n) r_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{F}_1(\alpha_n) r_n^{(k)} & \mathbf{F}_2(\alpha_n) r_n^{(k)} & \dots & \mathbf{F}_k(\alpha_n) r_n^{(k)} \end{bmatrix}. \quad (37)$$

Then worker n , (b, t, k) polynomial-shares $\mathbf{H}^{(n)}$ with workers in $[N]$. More precisely, according to (19), worker n forms the following polynomial

$$\mathbf{F}_{\mathbf{H}^{(n)},b,t,k}(x) = \sum_{j=1}^k \mathbf{H}_j^{(n)} x^{b(j-1)} + \sum_{j=1}^{t-1} \hat{\mathbf{R}}_j^{(n)} x^{k^2+j-1}, \quad (38)$$

where $\mathbf{H}_j^{(n)} \triangleq \begin{bmatrix} \mathbf{F}_j(\alpha_n)r_n^{(1)} \\ \mathbf{F}_j(\alpha_n)r_n^{(2)} \\ \vdots \\ \mathbf{F}_j(\alpha_n)r_n^{(k)} \end{bmatrix}$, for $j \in [k]$ and $\hat{\mathbf{R}}_j^{(n)}, j = 1, 2, \dots, t-1$, are chosen independently and

uniformly at random in $\mathbb{F}^{m \times \frac{m}{k}}$. Worker n sends $\mathbf{F}_{\mathbf{H}^{(n)},b,t,k}(\alpha_{n'})$ to the worker n' , for all $n' \in [N]$. All of the workers follow the same method. Thus, at the end of this step, each worker n' has access to the matrices $\{\mathbf{F}_{\mathbf{H}^{(1)},b,t,k}(\alpha_{n'}), \mathbf{F}_{\mathbf{H}^{(2)},b,t,k}(\alpha_{n'}), \dots, \mathbf{F}_{\mathbf{H}^{(N)},b,t,k}(\alpha_{n'})\}$.

3) *Step 3- Aggregation:*

Now worker n' calculates $\sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)},b,t,k}(\alpha_{n'})$. We claim that this summation is indeed the polynomial share of the matrix \mathbf{A}^T . To verify that, consider the polynomial function

$$\mathbf{Q}(x) \triangleq \sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)},b,t,k}(x). \quad (39)$$

We note that

$$\begin{aligned} \mathbf{Q}(x) &= \sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)},b,t,k}(x) \\ &= \sum_{n=1}^N \sum_{j=1}^k \mathbf{H}_j^{(n)} x^{b(j-1)} + \sum_{n=1}^N \sum_{j=1}^{t-1} \hat{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &= \sum_{n=1}^N \sum_{i=1}^k \begin{bmatrix} \mathbf{F}_i(\alpha_n)r_n^{(1)} \\ \mathbf{F}_i(\alpha_n)r_n^{(2)} \\ \vdots \\ \mathbf{F}_i(\alpha_n)r_n^{(k)} \end{bmatrix} x^{b(i-1)} + \sum_{n=1}^N \sum_{j=1}^{t-1} \hat{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &= \sum_{i=1}^k \sum_{n=1}^N \begin{bmatrix} \mathbf{F}_i(\alpha_n)r_n^{(1)} \\ \mathbf{F}_i(\alpha_n)r_n^{(2)} \\ \vdots \\ \mathbf{F}_i(\alpha_n)r_n^{(k)} \end{bmatrix} x^{b(i-1)} + \sum_{j=1}^{t-1} \sum_{n=1}^N \hat{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &\stackrel{(a)}{=} \sum_{i=1}^k \begin{bmatrix} \mathbf{A}_{i1} \\ \mathbf{A}_{i2} \\ \vdots \\ \mathbf{A}_{ik} \end{bmatrix} x^{b(i-1)} + \sum_{j=1}^{t-1} \left(\sum_{n=1}^N \hat{\mathbf{R}}_j^{(n)} \right) x^{k^2+j-1} \\ &\stackrel{(b)}{=} \sum_{i=1}^k \mathbf{L}_i x^{b(i-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1}, \end{aligned} \quad (40)$$

where (a) follows from (36), and (b) follows from the definitions $\mathbf{L}_i \triangleq \begin{bmatrix} \mathbf{A}_{i1} \\ \mathbf{A}_{i2} \\ \vdots \\ \mathbf{A}_{ik} \end{bmatrix}$, for $i \in [k]$, and

$\mathbf{R}_j \triangleq \sum_{n=1}^N \hat{\mathbf{R}}_j^{(n)}$, for $j \in [t-1]$. We note that $\mathbf{L} = [\mathbf{L}_1 \ \mathbf{L}_2 \ \dots \ \mathbf{L}_k]$. In addition, $\mathbf{R}_j, j \in [t-1]$, have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$. Thus, $\mathbf{Q}(x)$ is in the form of (b, t, k) polynomial sharing of \mathbf{A}^T with workers in $[N]$, and worker n has access to $\mathbf{Q}(\alpha_n)$. These steps are detailed in Algorithm 4.

Algorithm 4 Transposing

Inputs: Number N of the workers. Matrix \mathbf{A} which is (b, t, k) polynomial-shared with workers in $[N]$. Also $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, $r_n^{(j)}$ for $j \in [k]$, and $n \in [N]$ are known by all workers.

Step 1- Splitting:

- 1: Worker n calculates $\mathbf{F}_i(\alpha_n) \triangleq \mathbf{F}_{A,b,t,k}(\alpha_n)(\frac{m}{k}(i-1) : \frac{m}{k}i, :)$, for $i \in [k]$.

Step 2- Communication:

- 2: Worker n (b, t, k) polynomial-shares the matrix

$$\mathbf{H}^{(n)} \triangleq \begin{bmatrix} \mathbf{F}_1(\alpha_n)r_n^{(1)} & \mathbf{F}_2(\alpha_n)r_n^{(1)} & \dots & \mathbf{F}_k(\alpha_n)r_n^{(1)} \\ \mathbf{F}_1(\alpha_n)r_n^{(2)} & \mathbf{F}_2(\alpha_n)r_n^{(2)} & \dots & \mathbf{F}_k(\alpha_n)r_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{F}_1(\alpha_n)r_n^{(k)} & \mathbf{F}_2(\alpha_n)r_n^{(k)} & \dots & \mathbf{F}_k(\alpha_n)r_n^{(k)} \end{bmatrix}.$$

with workers in $[N]$.

Step 3- Aggregation:

- 3: Worker n calculates the sum of the messages received in the last step.
4: End.
-

E. Changing the parameter of sharing

Let $\mathbf{A} \in \mathbb{F}^{m \times m}$, and

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k],$$

where $\mathbf{A}_i \in \mathbb{F}^{m \times \frac{m}{k}}$, for $i \in [k]$ and $k|m$. In addition, assume \mathbf{A} is (b, t, k) polynomial-shared with workers in $[N]$. The goal is to (b', t, k) polynomial-share \mathbf{A} with workers in $[N]$, where $b' \neq b$.

Similar to (20) and (21), it can be proven that if $N \geq (k + t - 1)$, then by having $\mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n)$, for $n = 1, 2, \dots, N$, we can calculate all the coefficients of $\mathbf{F}_{\mathbf{A},b',t,k}(x)$, including \mathbf{A}_j , for $j \in [k]$, with probability approaching to one, as $|\mathbb{F}| \rightarrow \infty$. More precisely, similar to (20) and (21), it can be proven that for any distinct $i_1, i_2, \dots, i_{k+t-1} \in [N]$, independently and uniformly at random chosen parameters $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, and $i, j \in [k]$, with probability approaching to one, as $|\mathbb{F}| \rightarrow \infty$ we have

$$H(\mathbf{A}_j | \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_{i_1}), \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_{i_2}), \dots, \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_{i_{k+t-1}})) = 0.$$

In particular, there are some $r_n^{(j)}$, such that for $j \in [k]$, and $n \in [N]$,

$$\mathbf{A}_j = \sum_{n=1}^N r_n^{(j)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n). \quad (41)$$

Note that $r_n^{(j)}$, $j \in [k]$ and $n \in [N]$, are only functions of α_n , $n \in [N]$, which are known by all workers.

1) *Step 1- Communication:*

Worker n forms $\mathbf{H}^{(n)}$ defined as

$$\mathbf{H}^{(n)} \triangleq \begin{bmatrix} r_n^{(1)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n), & r_n^{(2)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n), & \dots, & r_n^{(k)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n) \end{bmatrix}. \quad (42)$$

Then worker n , (b', t, k) polynomial-shares $\mathbf{H}^{(n)}$ with workers in $[N]$. More precisely, according to (19), worker n forms the following polynomial

$$\mathbf{F}_{\mathbf{H}^{(n)},b',t,k}(x) = \sum_{j=1}^k \mathbf{H}_j^{(n)} x^{b'(j-1)} + \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1}, \quad (43)$$

where $\mathbf{H}_j^{(n)} \triangleq r_n^{(j)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n)$ for $j \in [k]$, and $\bar{\mathbf{R}}_j^{(n)}$, $j = 1, 2, \dots, t-1$, are chosen independently and uniformly at random in $\mathbb{F}^{m \times \frac{m}{k}}$. Worker n sends $\mathbf{F}_{\mathbf{H}^{(n)},b',t,k}(\alpha_{n'})$ to the worker n' , for all $n' \in [N]$. All of the workers follow the same method. Thus, at the end of this step, each worker n' has access to the matrices $\{\mathbf{F}_{\mathbf{H}^{(1)},b',t,k}(\alpha_{n'}), \mathbf{F}_{\mathbf{H}^{(2)},b',t,k}(\alpha_{n'}), \dots, \mathbf{F}_{\mathbf{H}^{(N)},b',t,k}(\alpha_{n'})\}$.

2) *Step 2- Aggregation:*

Now worker n' calculates $\sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)},b',t,k}(\alpha_{n'})$. We claim that this summation is indeed the polynomial share of the matrix \mathbf{A} . To verify that, consider the polynomial function

$$\mathbf{Q}(x) \triangleq \sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)},b',t,k}(x) \quad (44)$$

We note that

$$\begin{aligned} \mathbf{Q}(x) &= \sum_{n=1}^N \mathbf{F}_{\mathbf{H}^{(n)},b',t,k}(x) \\ &= \sum_{n=1}^N \sum_{j=1}^k \mathbf{H}_j^{(n)} x^{b'(j-1)} + \sum_{n=1}^N \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &= \sum_{n=1}^N \sum_{j=1}^k r_n^{(j)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n) x^{b'(j-1)} + \sum_{n=1}^N \sum_{j=1}^{t-1} \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &= \sum_{j=1}^k \sum_{n=1}^N r_n^{(j)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n) x^{b'(j-1)} + \sum_{j=1}^{t-1} \sum_{n=1}^N \bar{\mathbf{R}}_j^{(n)} x^{k^2+j-1} \\ &\stackrel{(a)}{=} \sum_{j=1}^k \mathbf{A}_j x^{b'(j-1)} + \sum_{j=1}^{t-1} \left(\sum_{n=1}^N \bar{\mathbf{R}}_j^{(n)} \right) x^{k^2+j-1} \\ &\stackrel{(b)}{=} \sum_{j=1}^k \mathbf{A}_j x^{b'(j-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1}, \end{aligned} \quad (45)$$

where (a) follows from (41), and (b) follows from the definition $\mathbf{R}_j \triangleq \sum_{n=1}^N \bar{\mathbf{R}}_j^{(n)}$, for $j \in [t-1]$. Note that, \mathbf{R}_j , $j \in [t-1]$ have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$. Thus, $\mathbf{Q}(x)$ is in the form of (b', t, k) polynomial sharing of \mathbf{A} with workers in $[N]$, and worker n has access to $\mathbf{Q}(\alpha_n)$. These steps are explained in Algorithm 5.

Algorithm 5 Changing the parameter of sharing

Inputs: Number N of the workers. Matrix \mathbf{A} which is (b, t, k) polynomial-shared with workers in $[N]$. Also $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, $r_n^{(j)}$ for $j \in [k]$, and $n \in [N]$ are known by all workers.

Step 1- Communication:

- 1: Worker n , (b', t, k) polynomial-shares the matrix

$$\mathbf{H}^{(n)}(x) \triangleq \begin{bmatrix} r_n^{(1)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n) & r_n^{(2)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n) & \dots & r_n^{(k)} \mathbf{F}_{\mathbf{A},b,t,k}(\alpha_n) \end{bmatrix}, \quad (46)$$

with workers in $[N]$.

Step 2- Aggregation:

- 2: Worker n calculates the sum of the messages received in the last step.
 - 3: End.
-

VIII. THE PROPOSED ALGORITHM

As we mentioned before, in Section VII, at the end of all procedures 1-5, the output is in the form of *polynomial sharing*. This allows us to calculate any polynomial function of the inputs by concatenating these procedures accordingly. We explain this in detail in Algorithm 6.

In the proposed algorithm, we use the arithmetic representation of the function described in Appendix C. In the following theorem we claim that this algorithm satisfies constraints (3), (4), and (5).

Algorithm 6 Proposed Algorithm

Assume that $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$ are chosen independently and uniformly at random in \mathbb{F} , and are available everywhere.

Phase 1- Secret Sharing: Every source node $\gamma \in [\Gamma]$ takes the following steps.

- 1: Calculates $\mathbf{F}_{\mathbf{X}^{[\gamma]},1,t,k}(x)$.
- 2: Sends $\mathbf{F}_{\mathbf{X}^{[\gamma]},1,t,k}(\alpha_n)$ to worker $n \in [N]$.

Phase 2- Computation and communication: All of the workers consider the arithmetic representation of the function according to the rules in Appendix C. Worker $n \in [N]$ takes the following steps.

- 3: If it assesses all of the gates, it goes to the next phase, otherwise it considers the first non-assessed gate. Call that gate g .
- 4: If g is an addition gate, it follows Procedure 1 and go to Step 3.
- 5: If g is a multiplication by constant gate, it follows Procedure 2 and go to Step 3.
- 6: If g is a multiplication of two matrices, call the output of the gate \mathbf{C} , and assume we have $\mathbf{C} = \mathbf{A}^T \mathbf{B}$, where \mathbf{A}, \mathbf{B} are inputs of the gate.
- 7: If it has access to $\mathbf{F}_{\mathbf{B}^T,1,t,k}(\alpha_n)$, it follows Procedure 4 to access $\mathbf{F}_{\mathbf{B},1,t,k}(\alpha_n)$, and goes to Step 8.
- 8: If it has access to $\mathbf{F}_{\mathbf{B},1,t,k}(\alpha_n)$, it follows Procedure 5 to access $\mathbf{F}_{\mathbf{B},k,t,k}(\alpha_n)$, and goes to Step 9.
- 9: If it has access to $\mathbf{F}_{\mathbf{A}^T,1,t,k}(\alpha_n)$, it follows Procedure 4 to access $\mathbf{F}_{\mathbf{A},1,t,k}(\alpha_n)$.
- 10: Follows Procedure 3 to access $\mathbf{F}_{\mathbf{C},1,t,k}(\alpha_n)$ and it goes to Step 3.

Phase 3- Reconstruction: Worker $n \in [N]$ takes the following steps.

- 11: Stores the output of the arithmetic representation of the function in \mathbf{O}_n .
 - 12: Sends \mathbf{O}_n to the master.
 - 13: The master uses the method in [35] to reconstruct the result using matrices $\mathbf{O}_n, n \in [N]$.
 - 14: End.
-

Theorem 3. Algorithm 6 satisfies constraints (3), (4), and (5), for any $\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]} \in \mathbb{F}^{m \times m}$, and polynomial function $\mathbf{G} : (\mathbb{F}^{m \times m})^\Gamma \rightarrow \mathbb{F}^{m \times m}$.

Proof. To see the proof that conditions (4) and (5) are satisfied, see Appendix B. For constraint (3), note that the master receives the polynomial sharing of the result from all of the workers, and according to Theorem 2, these shares are enough to reconstruct the result. \square

A. Computation and communication complexity

In this section, we evaluate the computation and communication complexity of all the procedures and phases of Algorithm 6, explained throughout the paper.

- Sharing phase: In this phase each source node $\gamma \in [\Gamma]$ has to evaluate $\mathbf{F}_{\mathbf{X}^{[\gamma]},1,t,k}(x)$, for N distinct values. According to (19) we have

$$\mathbf{F}_{\mathbf{X}^{[\gamma]},1,t,k}(x) = \sum_{j=1}^k \mathbf{X}_j^{[\gamma]} x^{j-1} + x^{k^2} \sum_{j=1}^{t-1} \mathbf{R}_j x^{j-1}. \quad (47)$$

Using Horner's method [36], calculating a polynomial of degree n can be done in $\mathcal{O}(n)$. Therefore, calculating $\sum_{j=1}^k \mathbf{X}_j^{[\gamma]} x^{j-1}$ and $x^{k^2} \sum_{j=1}^{t-1} \mathbf{R}_j x^{j-1}$ can be done in $\mathcal{O}(k(m \times \frac{m}{k}))$ and $\mathcal{O}((t-1)(m \times \frac{m}{k}) +$

$2 \log k$), respectively. Therefore, evaluating $\mathbf{F}_{\mathbf{x}^{[n]}, 1, t, k}(x)$ at any point can be done in $\mathcal{O}((k+t)(\frac{m^2}{k}))$. Thus, sharing phase can be done with the computation complexity of $\mathcal{O}((k+t)(\frac{m^2}{k})(N))$.

Also it has to send the resulting $m \times \frac{m}{k}$ matrices to each worker. Therefore, there is an aggregated communication complexity of $\mathcal{O}((N\Gamma)(\frac{m^2}{k}))$.

- **Addition procedure:** In this procedure, each worker computes only the addition of two matrices of dimensions $m \times \frac{m}{k}$ with the computation complexity of $\mathcal{O}(\frac{m^2}{k})$. Also, there is no communication between the workers in this procedure.
- **Multiplication by constant procedure:** In this procedure, each worker computes only the multiplication of a matrix of dimension $m \times \frac{m}{k}$ and a constant number with the computation complexity of $\mathcal{O}(\frac{m^2}{k})$. Also, there is no communication between the workers in this procedure.
- **Multiplication of two matrices:** In this procedure, first of all, each worker computes only the multiplication of two matrices of dimensions $\frac{m}{k} \times m$ and $m \times \frac{m}{k}$. If it is done conventionally, it can be executed with the computation complexity of $\mathcal{O}(\frac{m^3}{k^2})$. Then, each worker n has to evaluate polynomial sharing of $\mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(x)$ at N points, which has a complexity of $\mathcal{O}((k+t)(\frac{m^2}{k})(N))$. Finally, each worker has to sum up N matrices of dimensions $m \times \frac{m}{k}$ with the computation complexity of $\mathcal{O}((\frac{m^2}{k})(N))$. Therefore, per node computation complexity of this step is $\mathcal{O}(\frac{m^3}{k^2} + (k+t)(\frac{m^2}{k})(N) + (\frac{m^2}{k})(N)) = \mathcal{O}(\frac{m^3}{k^2} + (k+t)(\frac{m^2}{k})(N))$. Also there is a communication of $\mathcal{O}(N^2)$ of $m \times \frac{m}{k}$ matrices between the workers, with the aggregated communication complexity of $\mathcal{O}((N^2)(\frac{m^2}{k}))$.
- **Transposing:** In this procedure, each worker n has to evaluate polynomial function of $\mathbf{F}_{\mathbf{H}^{(n)}, b, t, k}(x)$ at N points, which has complexity of $\mathcal{O}((k+t)(\frac{m^2}{k})(N))$. Then, each worker has to sum up N matrices of dimensions $m \times \frac{m}{k}$ with the computation complexity of $\mathcal{O}((\frac{m^2}{k})(N))$. Therefore, the per node computation complexity of this step is $\mathcal{O}((k+t)(\frac{m^2}{k})(N) + (\frac{m^2}{k})(N)) = \mathcal{O}((k+t)(\frac{m^2}{k})(N))$. Also there is a communication of $\mathcal{O}(N^2)$ of $m \times \frac{m}{k}$ matrices between the workers, which implies that there is an aggregated communication complexity of $\mathcal{O}((N^2)(\frac{m^2}{k}))$.
- **Changing the parameter of sharing:** In this procedure, each worker n has to evaluate polynomial sharing of $\mathbf{F}_{\mathbf{H}^{(n)}, b', t, k}(x)$ at N points, which has complexity of $\mathcal{O}((k+t)(\frac{m^2}{k})(N))$. Then, each worker has to sum up N matrices of dimensions $m \times \frac{m}{k}$ with the computation complexity of $\mathcal{O}((\frac{m^2}{k})(N))$. Therefore, the per node computation complexity of this step is $\mathcal{O}((k+t)(\frac{m^2}{k})(N) + (\frac{m^2}{k})(N)) = \mathcal{O}((k+t)(\frac{m^2}{k})(N))$. Also there is a communication of $\mathcal{O}(N^2)$ of $m \times \frac{m}{k}$ matrices between the workers, with the aggregated communication complexity of $\mathcal{O}((N^2)(\frac{m^2}{k}))$.
- **Reconstruction phase:** In this phase, the master has to reconstruct the value of the result, using the messages it has received from the workers. Since the result is in the form of polynomial sharing at the master, we have to interpolate a polynomial of degree $k^2 + t - 2$, which can be done with the complexity of $\mathcal{O}((k^2 + t - 2) \log^2(k^2 + t - 2) \log \log(k^2 + t - 2))$, according to [35]. Also, there is a communication of N of $m \times \frac{m}{k}$ matrices between the workers and master, with the aggregated communication complexity of $N \frac{m^2}{k}$.

According to the above calculations, multiplication of two matrices has the most computation complexity among the all of the procedures described throughout the paper. Now let us calculate the computation complexity of the function G . Assume that v is the number of monomial terms of \mathbf{G} , and $d = \deg \mathbf{G}$. Therefore, the per node computation complexity of the function \mathbf{G} is at most $\mathcal{O}((vd)(\frac{m^3}{k^2} + (k+t)(\frac{m^2}{k})N))$. Also the aggregated communication complexity of the function \mathbf{G} is at most $\mathcal{O}(N\Gamma(\frac{m^2}{k}) + dN^2(\frac{m^2}{k}) + N(\frac{m^2}{k})) = \mathcal{O}(N\frac{m^2}{k}(dN + \Gamma))$.

In the context of MPC, there are some solutions that decrease the communication complexity, which are not information theoretic [6], [37]–[40]. One idea that is worth exploring is to combine the idea of this work and those solutions.

IX. EXTENSION

In the proposed scheme in Section VIII, in order to share the matrix according to the *polynomial sharing* scheme, we partition it column-wise. This model of partitioning and sharing can be extended. In general, we can partition the matrix into some blocks and share the matrix according to this configuration. This approach is inspired and motivated by *entangled polynomial code* [14], or *MatDot code* [15].

Definition 3. Let $\mathbf{A} \in \mathbb{F}^{z \times v}$, for some $z, v \in \mathbb{N}$, be

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} & \mathbf{A}_{0,2} & \cdots & \mathbf{A}_{0,m-1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,m-1} \\ \mathbf{A}_{2,0} & \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{p-1,0} & \mathbf{A}_{p-1,1} & \mathbf{A}_{p-1,2} & \cdots & \mathbf{A}_{p-1,m-1} \end{bmatrix},$$

where $\mathbf{A}_{i,j} \in \mathbb{F}^{k \times k'}$, for some $k, k' \in \mathbb{N}$, $k|z$, and $k'|v$.

We define the entangled polynomial function $\mathbf{F}_{\mathbf{A},b,p,m,n,t,s}(x)$, for some $p, m, n, t \in \mathbb{N}$, as

$$\begin{aligned} \mathbf{F}_{\mathbf{A},b,p,m,n,t,+}(x) &\triangleq \sum_{j=0}^{p-1} \sum_{k=0}^{m-1} \mathbf{A}_{j,k} x^{j+bkp} + \sum_{q=0}^{t-2} \mathbf{R}'_q x^{npm+q}, \\ \mathbf{F}_{\mathbf{A},b,p,m,n,t,-}(x) &\triangleq \sum_{j=0}^{p-1} \sum_{k=0}^{m-1} \mathbf{A}_{j,k} x^{(p-1-j)+bkp} + \sum_{q=0}^{t-2} \mathbf{R}'_q x^{npm+q}, \end{aligned}$$

where $\mathbf{R}'_q, q \in \{0, 1, \dots, t-2\}$, are chosen independently and uniformly at random from $\mathbb{F}^{k \times k'}$.

Now with this method of sharing we can do more general models of polynomial calculation (see [23]).

X. CONCLUSION AND DISCUSSION

In this paper, we developed a new secure multiparty computation for massive input data. The proposed solution offers significant gains compared to schemes based on splitting the data into smaller pieces and applying conventional multiparty computation. In this work, we assumed that some of the nodes are semi-honest. The next step is to consider the case where nodes are adversarial, which has been addressed in [41]. There are many open problems in this direction. This includes exploring communication efficiency, the tradeoff between the number of servers and communication load, having a network of heterogeneous servers, various network topologies, and the cases where some communication links are eavesdropped. In addition, investigating the case where the sources are collocated and can be encoded together would be interesting. Here we assume that we want to calculate $\mathbf{G}(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]})$, where inputs are massive. One interesting direction is to consider the case, where the goal is to calculate $\mathbf{G}(\mathbf{X}_i^{[1]}, \mathbf{X}_i^{[2]}, \dots, \mathbf{X}_i^{[\Gamma]})$, for $i = 1, \dots, k$, for some integer k . This would be in the intersection of this work and [19]. Exploiting the sparsity of the input data in this calculation would also be of great interest (see [20]).

REFERENCES

- [1] H. A. Nodehi and M. A. Maddah-Ali, "Limited-sharing multi-party computation for massive matrix operations," in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pp. 1231–1235, 2018.
- [2] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," in *In Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, pp. 503–513, 1990.
- [3] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified vss and fast-track multiparty computations with applications to threshold cryptography," in *In Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing*, pp. 101–111, 1998.
- [4] 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*, pp. 1–10, 1988.
- [5] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," in *In Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pp. 11–19, 1988.
- [6] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Annual International Cryptology Conference*, pp. 420–432, Springer, 1991.

- [7] D. Evans, V. Kolesnikov, and M. Rosulek, “A pragmatic introduction to secure multi-party computation,” *Foundations and Trends in Privacy and Security*, vol. 2, no. 2-3, pp. 70–246, 2018.
- [8] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [9] J. Saia and M. Zamani, “Recent results in scalable multi-party computation,” in *Proceedings of International Conference on Current Trends in Theory and Practice of Informatics*, pp. 24–44, 2015.
- [10] J. Dean and L. A. Barroso, “The tail at scale,” *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.
- [11] 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, 2018.
- [12] K. Lee, C. Suh, and K. Ramchandran, “High-dimensional coded matrix multiplication,” in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pp. 2418–2422, 2017.
- [13] Q. Yu, M. Maddah-Ali, and S. Avestimehr, “Polynomial codes: an optimal design for high-dimensional coded matrix multiplication,” in *Advances in Neural Information Processing Systems*, pp. 4403–4413, 2017.
- [14] Q. Yu, M. Maddah-Ali, and A. Avestimehr, “Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding,” in *Proceedings of IEEE International Symposium on Information Theory*, pp. 2022–2026, 2018.
- [15] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” in *Proceedings of 55th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1264–1270, 2018.
- [16] S. Dutta, V. Cadambe, and P. Grover, “Short-dot: Computing large linear transforms distributedly using coded short dot products,” in *Advances in Neural Information Processing Systems*, pp. 2092–2100, 2016.
- [17] V. Gupta, S. Wang, T. Courtade, and K. Ramchandran, “Oversketch: Approximate matrix multiplication for the cloud,” pp. 298–304, 2018.
- [18] S. Wang, J. Liu, and N. Shroff, “Coded sparse matrix multiplication,” in *Proceedings of 35th International Conference on Machine Learning (ICML)*, vol. 12, pp. 8176–8193, 2018.
- [19] 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,” pp. 1215–1225, 2019.
- [20] T. Jahani-Nezhad and M. A. Maddah-Ali, “Codedsketch: A coding scheme for distributed computation of approximated matrix multiplication,” *arXiv preprint arXiv:1812.10460*, 2018.
- [21] A. Lapets, F. Jansen, K. D. Albab, R. Issa, L. Qin, M. Varia, and A. Bestavros, “Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities,” in *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies, COMPASS '18*, 2018.
- [22] C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.
- [23] H. A. Nodehi, S. R. H. Najarkolaie, and M. A. Maddah-Ali, “Entangled polynomial coding in limited-sharing multi-party computation,” in *Proceedings of IEEE Information Theory Workshop*, 2018.
- [24] Q. Yu, N. Raviv, and A. S. Avestimehr, “Coding for private and secure multiparty computing,” in *2018 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2018.
- [25] 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.
- [26] W.-T. Chang and R. Tandon, “On the capacity of secure distributed matrix multiplication,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.
- [27] J. Kakar, S. Ebadifar, and A. Sezgin, “On the capacity and straggler-robustness of distributed secure matrix multiplication,” *IEEE Access*, vol. 7, pp. 45783–45799, 2019.
- [28] R. G. DOLiveira, S. El Rouayheb, and D. Karpuk, “Gasp codes for secure distributed matrix multiplication,” *IEEE Transactions on Information Theory*, 2020.
- [29] Z. Jia and S. A. Jafar, “On the capacity of secure distributed matrix multiplication,” *arXiv preprint arXiv:1908.06957*, 2019.
- [30] M. Kim and J. Lee, “Private secure coded computation,” in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1097–1101, IEEE, 2019.
- [31] 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.
- [32] B. Tahmasebi and M. A. Maddah-Ali, “Private sequential function computation,” in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1667–1671, 2019.
- [33] B. Tahmasebi and M. A. Maddah-Ali, “Private function computation,” in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 1118–1123, 2020.
- [34] N. S. Bakhvalov, “Numerical methods: analysis, algebra, ordinary differential equations,” 1977.
- [35] K. S. Kedlaya and C. Umans, “Fast polynomial factorization and modular composition,” *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1767–1802, 2011.
- [36] Wikipedia contributors, “Horner’s method — Wikipedia, the free encyclopedia,” 2020. [Online; accessed 5-June-2020].
- [37] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, “Practical covertly secure mpc for dishonest majority—or: breaking the spdz limits,” in *European Symposium on Research in Computer Security*, pp. 1–18, Springer, 2013.
- [38] G. Couteau, “A note on the communication complexity of multiparty computation in the correlated randomness model,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 473–503, Springer, 2019.
- [39] M. Yoshida and S. Obana, “On the (in) efficiency of non-interactive secure multiparty computation,” *Designs, Codes and Cryptography*, vol. 86, no. 8, pp. 1793–1805, 2018.
- [40] A. Choudhury and A. Patra, “An efficient framework for unconditionally secure multiparty computation,” *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 428–468, 2016.

- [41] M. A. M.-A. Seyed Reza Hoseini and M. R. Aref, "Secure coded multi-party computation for massive matrices with adversarial nodes," in *International Conference on Machine Learning (ICML)*, 2019.
- [42] G. Sobczyk, "Generalized vandermonde determinants and applications," *Aportaciones Matematicas, Serie Comunicaciones*, vol. 30, pp. 203–213, 2002.
- [43] T. Kitamoto, "On the computation of the determinant of a generalized vandermonde matrix," pp. 242–255, 2014.
- [44] Wikipedia contributors, "Schur polynomial — Wikipedia, the free encyclopedia," 2020. [Online; accessed 29-May-2020].
- [45] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 782–795, 2003.
- [46] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.
- [47] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *Symbolic and algebraic computation*, pp. 216–226, Springer, 1979.

APPENDIX A
PROOF OF THEOREM 2

In order to prove Theorem 2, we first prove the following lemma.

Lemma 4. Let $\mathbf{A}, \mathbf{B} \in \mathbb{F}^{m \times m}$, and

$$\begin{aligned}\mathbf{A} &= [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k], \\ \mathbf{B} &= [\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_k],\end{aligned}$$

where $\mathbf{A}_i, \mathbf{B}_i \in \mathbb{F}^{m \times \frac{m}{k}}$, for $i \in [k]$ and $k|m$. In addition, assume that these matrices are shared among N workers using polynomial functions $\mathbf{F}_{\mathbf{A},1,t,k}(x)$ and $\mathbf{F}_{\mathbf{B},k,t,k}(x)$, respectively. Let us define

$$\mathbf{H}(x) = \sum_{n=0}^{2(k^2+t-2)} \mathbf{H}_n x^n \triangleq \mathbf{F}_{\mathbf{A},1,t,k}^T(x) \mathbf{F}_{\mathbf{B},k,t,k}(x). \quad (48)$$

The number of nonzero coefficients of $\mathbf{H}(x)$ is equal to

$$\min\{2k^2 + 2t - 3, k^2 + kt + t - 2\} = \begin{cases} 2k^2 + 2t - 3 & \text{if } k < t \\ k^2 + kt + t - 2 & \text{if } k \geq t \end{cases}.$$

Proof. We have

$$\begin{aligned}\mathbf{F}_{\mathbf{A},1,t,k}^T(x) &= \sum_{n=1}^k \mathbf{A}_n^T x^{n-1} + x^{k^2} \sum_{n=1}^{t-1} \bar{\mathbf{R}}_n^T x^{n-1}, \\ \mathbf{F}_{\mathbf{B},k,t,k}(x) &= \sum_{n=1}^k \mathbf{B}_n x^{k(n-1)} + x^{k^2} \sum_{n=1}^{t-1} \hat{\mathbf{R}}_n x^{n-1}.\end{aligned}$$

- The power of x with nonzero coefficients in $(\sum_{n=1}^k \mathbf{A}_n^T x^{n-1})(\sum_{n=1}^k \mathbf{B}_n x^{k(n-1)})$, are $\mathcal{S}_1 \triangleq \{0, 1, 2, \dots, k^2 - 1\}$.
- The power of x with nonzero coefficients in $(x^{k^2} \sum_{n=1}^k \mathbf{A}_n^T x^{n-1})(\sum_{n=1}^{t-1} \hat{\mathbf{R}}_n x^{n-1})$, are $\mathcal{S}_2 \triangleq \{k^2, k^2 + 1, k^2 + 2, \dots, k^2 + k - 1 + t - 2\}$.
- The power of x with nonzero coefficients in $(x^{k^2} \sum_{n=1}^{t-1} \bar{\mathbf{R}}_n^T x^{n-1})(\sum_{n=1}^k \mathbf{B}_n x^{k(n-1)})$, are $\mathcal{S}_3 \triangleq \{k^2 + ik + j, i \in [0, k-1], j \in [0, t-2]\}$.
- The power of x with nonzero coefficients in $(x^{2k^2} \sum_{n=1}^{t-1} \bar{\mathbf{R}}_n^T x^{n-1})(\sum_{n=1}^{t-1} \hat{\mathbf{R}}_n x^{n-1})$, are $\mathcal{S}_4 \triangleq \{2k^2, 2k^2 + 1, 2k^2 + 2, \dots, 2k^2 + 2t - 4\}$.

The degree of the polynomial $\mathbf{H}(x)$ is $2k^2 + 2t - 4$. Therefore, there are $2k^2 + 2t - 3$ coefficients from 0 to $2k^2 + 2t - 4$, where some of them are zero. One can see that the set of zero coefficients of $\mathbf{H}(x)$ is equal to

$$[0, 2k^2 + 2t - 4] - (\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4). \quad (49)$$

We consider the following two cases.

- 1) Case 1, $k - 1 \leq t - 2$: In this case one can see that we have

$$(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4) = [0, 2k^2 + 2t - 4].$$

Therefore, in this case non of the coefficients of $\mathbf{H}(x)$, is equal to zero. Thus, the number of nonzero coefficients of $\mathbf{H}(x)$ is $2k^2 + 2t - 3$.

- 2) Case 2, $k-1 > t-2$: In this case, counting the number of non-zero coefficients is more complicated, specially because the intersection $\mathcal{S}_2 \cap \mathcal{S}_3$ is not zero. In this case we claim that the number of zero coefficients of $\mathbf{H}(x)$ is $(k-t+1)(k-1)$, thus the number of nonzero coefficients is $(2k^2 + 2t - 3) - (k-t+1)(k-1) = k^2 + kt + t - 2$.

Let us define

$$\begin{aligned}\mathcal{S}_{21} &= \{k^2, k^2 + 1, k^2 + 2, \dots, k^2 + k - 1\}, \\ \mathcal{S}_{22} &= \mathcal{S}_2 - \mathcal{S}_{21}, \\ \mathcal{S}_{3i} &= \{k^2 + ik + j, j \in [0, t-2]\},\end{aligned}$$

for $i \in [0, k-1]$.

In this case ($k-1 > t-2$), one can see that we have

$$\mathcal{S}_2 = \mathcal{S}_{21} \cup \mathcal{S}_{22}, \quad (50)$$

$$\mathcal{S}_3 = \bigcup_{i=0}^{k-1} \mathcal{S}_{3i},$$

$$\mathcal{S}_{30} \subset \mathcal{S}_{21}, \quad (51)$$

$$\mathcal{S}_3 \cap \mathcal{S}_{21} = \mathcal{S}_{30},$$

$$\mathcal{S}_{22} \subset \mathcal{S}_{31},$$

$$\mathcal{S}_1 \cup \mathcal{S}_{21} = [0, k^2 + k - 1],$$

$$\mathcal{S}_1 \cup \mathcal{S}_{21} \cup \mathcal{S}_4 = [0, k^2 + k - 1] \cup [2k^2, 2k^2 + 2t - 4],$$

$$(\mathcal{S}_1 \cup \mathcal{S}_{21} \cup \mathcal{S}_4) \cap (\mathcal{S}_{22} \cup (\mathcal{S}_3 - \mathcal{S}_{30})) = \emptyset, \quad (52)$$

$$\mathcal{S}_{3i} \cap \mathcal{S}_{3j} = \emptyset,$$

for distinct $i, j \in [0, k-1]$.

It is important to note that

$$\begin{aligned}(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4) &\stackrel{(a)}{=} (\mathcal{S}_1 \cup (\mathcal{S}_{21} \cup \mathcal{S}_{22}) \cup \mathcal{S}_3 \cup \mathcal{S}_4) \\ &= ((\mathcal{S}_1 \cup \mathcal{S}_{21} \cup \mathcal{S}_4) \cup (\mathcal{S}_3 \cup \mathcal{S}_{22})) \\ &= ((\mathcal{S}_1 \cup \mathcal{S}_{21} \cup \mathcal{S}_4) \cup ((\mathcal{S}_3 - \mathcal{S}_{30}) \cup \mathcal{S}_{30} \cup \mathcal{S}_{22})) \\ &= ((\mathcal{S}_1 \cup (\mathcal{S}_{21} \cup \mathcal{S}_{30}) \cup \mathcal{S}_4) \cup ((\mathcal{S}_3 - \mathcal{S}_{30}) \cup \mathcal{S}_{22})) \\ &\stackrel{(b)}{=} ((\mathcal{S}_1 \cup \mathcal{S}_{21} \cup \mathcal{S}_4) \cup (\mathcal{S}_3 - \mathcal{S}_{30})),\end{aligned} \quad (53)$$

where (a) follows from (50) and (b) follows from (51) and the fact that $\mathcal{S}_{22} \subset \mathcal{S}_{31} \subset (\mathcal{S}_3 - \mathcal{S}_{30})$. Also note that

$$[0, 2k^2 + 2t - 4] - (\mathcal{S}_1 \cup \mathcal{S}_{21} \cup \mathcal{S}_4) = \{k^2 + k, k^2 + k + 1, \dots, 2k^2 - 1\}. \quad (54)$$

Thus, according to (49), (52), and (53), in order to calculate the number of zero-coefficients, we must exclude $(\mathcal{S}_3 - \mathcal{S}_{30})$ from the set $\{k^2 + k, k^2 + k + 1, \dots, 2k^2 - 1\}$. Therefore, the number of zero-coefficients of $\mathbf{H}(x)$ is equal to

$$\begin{aligned}|\{k^2 + k, k^2 + k + 1, \dots, 2k^2 - 1\} - (\mathcal{S}_3 - \mathcal{S}_{30})| &\stackrel{(a)}{=} (k^2 - k) - (t-1)(k-1) \\ &= k(k-1) - (t-1)(k-1) \\ &= (k-t+1)(k-1)\end{aligned}$$

where (a) follows from $|\{k^2 + k, k^2 + k + 1, \dots, 2k^2 - 1\}| = k^2 - k$, $|\mathcal{S}_3 - \mathcal{S}_{30}| = (t-1)(k-1)$, and $(\mathcal{S}_3 - \mathcal{S}_{30} \subset \{k^2 + k, k^2 + k + 1, \dots, 2k^2 - 1\})$.

□

Now we prove Theorem 2. First note that based on Lemma 4, the number of nonzero coefficients of $\mathbf{H}(x)$ is

$$\min\{2k^2 + 2t - 3, k^2 + kt + t - 2\} = \begin{cases} 2k^2 + 2t - 3 & \text{if } k < t \\ k^2 + kt + t - 2 & \text{if } k \geq t \end{cases}.$$

Assume that $N = \min\{2k^2 + 2t - 3, k^2 + kt + t - 2\}$, and consider distinct values $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$. Let us define

$$\mathbf{H} \triangleq \begin{bmatrix} \alpha_1^{j_1} & \alpha_1^{j_2} & \dots & \alpha_1^{j_N} \\ \alpha_2^{j_1} & \alpha_2^{j_2} & \dots & \alpha_2^{j_N} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_N^{j_1} & \alpha_N^{j_2} & \dots & \alpha_N^{j_N} \end{bmatrix},$$

where $j_1, j_2, \dots, j_N \in \{0, 1, \dots, 2(k^2 + t - 2)\}$ are the distinct indexes of the nonzero coefficients of $\mathbf{H}(x)$. Also, for any distinct $i_1 < i_2 < \dots < i_{k+t-1} \in [N]$, define

$$\mathbf{A}_{i_1, i_2, \dots, i_{k+t-1}} \triangleq \begin{bmatrix} \alpha_{i_1}^0 & \alpha_{i_1}^1 & \dots & \alpha_{i_1}^{k-1} & \alpha_{i_1}^{k^2} & \alpha_{i_1}^{k^2+1} & \dots & \alpha_{i_1}^{k^2+t-2} \\ \alpha_{i_2}^0 & \alpha_{i_2}^1 & \dots & \alpha_{i_2}^{k-1} & \alpha_{i_2}^{k^2} & \alpha_{i_2}^{k^2+1} & \dots & \alpha_{i_2}^{k^2+t-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{i_{k+t-1}}^0 & \alpha_{i_{k+t-1}}^1 & \dots & \alpha_{i_{k+t-1}}^{k-1} & \alpha_{i_{k+t-1}}^{k^2} & \alpha_{i_{k+t-1}}^{k^2+1} & \dots & \alpha_{i_{k+t-1}}^{k^2+t-2} \end{bmatrix},$$

$$\mathbf{B}_{i_1, i_2, \dots, i_{k+t-1}} \triangleq \begin{bmatrix} \alpha_{i_1}^0 & \alpha_{i_1}^k & \dots & \alpha_{i_1}^{(k-1)k} & \alpha_{i_1}^{k^2} & \alpha_{i_1}^{k^2+1} & \dots & \alpha_{i_1}^{k^2+t-2} \\ \alpha_{i_2}^0 & \alpha_{i_2}^k & \dots & \alpha_{i_2}^{(k-1)k} & \alpha_{i_2}^{k^2} & \alpha_{i_2}^{k^2+1} & \dots & \alpha_{i_2}^{k^2+t-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{i_{k+t-1}}^0 & \alpha_{i_{k+t-1}}^k & \dots & \alpha_{i_{k+t-1}}^{(k-1)k} & \alpha_{i_{k+t-1}}^{k^2} & \alpha_{i_{k+t-1}}^{k^2+1} & \dots & \alpha_{i_{k+t-1}}^{k^2+t-2} \end{bmatrix}.$$

These three matrices are called Generalized Vandermonde matrix [42], [43], which has been extensively studied in the literatures. In [43] it has been shown that the determinant of the generalized Vandermonde matrix $\mathbf{A}_{i_1, i_2, \dots, i_{k+t-1}}$ is

$$\det \mathbf{A}_{i_1, i_2, \dots, i_{k+t-1}} = g(\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_{k+t-1}}) \prod_{i_j > i_l} (\alpha_{i_j} - \alpha_{i_l}),$$

where the function $g(\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_{k+t-1}})$ is called a Schur polynomial [44]. Unfortunately, there is no guarantee that $g(\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_{k+t-1}})$ is not equal to zero, given that α_j 's are distinct. Thus we cannot say that $\mathbf{A}_{i_1, i_2, \dots, i_{k+t-1}}$ is full rank.

To prove Theorem 2, its enough to show that for large enough $|\mathbb{F}|$, there exist distinct values $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, such that for any distinct $i_1 < i_2 < \dots < i_{k+t-1} \in [N]$, we have

$$\det \mathbf{A}_{i_1, i_2, \dots, i_{k+t-1}} \neq 0, \quad (55)$$

$$\det \mathbf{B}_{i_1, i_2, \dots, i_{k+t-1}} \neq 0, \quad (56)$$

$$\det \mathbf{H} \neq 0, \quad (57)$$

and if we choose $\alpha_1, \alpha_2, \dots, \alpha_N$, independently and uniformly at random in \mathbb{F} , the probability that (55), (56), and (57) hold, approaches to one, as $|\mathbb{F}| \rightarrow \infty$.

Let us define

$$f(\alpha_1, \alpha_2, \dots, \alpha_N) \triangleq \left(\prod_{i_1, i_2, \dots, i_{k+t-1} \in [N]} \det \mathbf{A}_{i_1, i_2, \dots, i_{k+t-1}} \det \mathbf{B}_{i_1, i_2, \dots, i_{k+t-1}} \right) \det \mathbf{H}.$$

This polynomial is not equal to zero polynomial. Thus, according to [45], for large enough $|\mathbb{F}|$, there exist distinct $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{F}$, such that $f(\alpha_1, \alpha_2, \dots, \alpha_N) \neq 0$. Also, based on Schwartz-Zippel Lemma [46], [47], if we choose $\alpha_1, \alpha_2, \dots, \alpha_N$, independently and uniformly at random in \mathbb{F} , the probability that (55), (56), and (57) hold, approaches to one, as $|\mathbb{F}| \rightarrow \infty$.

APPENDIX B
PROOF OF PRIVACY IN THEOREM 3

Recall that in this algorithm, to share any information, we always add some random matrices to it. We claim that this protocol satisfies privacy constraints (4) and (5). In order to formally prove that, we use the following two lemmas.

Lemma 5. Consider the polynomial $r(x): \mathbb{F} \rightarrow \mathbb{F}$

$$r(x) = \sum_{n=1}^{t-1} a_n x^{n-1},$$

where a_1, a_2, \dots, a_{t-1} , are chosen independently and uniformly at random in \mathbb{F} . Define

$$\tilde{\mathbf{r}} \triangleq (r(\alpha_1), r(\alpha_2), \dots, r(\alpha_{t-1})),$$

for some distinct values $\alpha_1, \alpha_2, \dots, \alpha_{t-1} \in \mathbb{F}$. Then $\tilde{\mathbf{r}}$ has a uniform distribution over \mathbb{F}^{t-1} .

Proof. Assume that r_1, r_2, \dots, r_{t-1} are chosen independently and uniformly at random in \mathbb{F} . According to Lagrange interpolation rule [34], we know that the following set of equations

$$\begin{aligned} r(\alpha_1) &= r_1, \\ r(\alpha_2) &= r_2, \\ &\vdots \\ r(\alpha_{t-1}) &= r_{t-1}, \end{aligned}$$

has a unique answer. It means that if we know the values of r_1, r_2, \dots, r_{t-1} , we can uniquely determine the values of a_i , for $i \in [t-1]$. Also it is obvious that if we know the values of a_1, a_2, \dots, a_{t-1} , we can uniquely determine the values of $r(\alpha_1), r(\alpha_2), \dots, r(\alpha_{t-1})$. Therefore, there is a one to one mapping between $\mathbf{a} = (a_1, a_2, \dots, a_{t-1})$ and $\tilde{\mathbf{r}} = (r(\alpha_1), r(\alpha_2), \dots, r(\alpha_{t-1}))$. Note that $a_i, i = 1, 2, \dots, t-1$, are chosen independently and uniformly at random in \mathbb{F} , thus \mathbf{a} has uniform distribution over \mathbb{F}^{t-1} . Therefore, $\tilde{\mathbf{r}}$ has uniform distribution over \mathbb{F}^{t-1} , too. \square

Corollary 6. Assume that $\mathbf{R}(x): \mathbb{F} \rightarrow \mathbb{F}^{p \times q}$ is a polynomial function of degree $\max(0, t-2)$, $t \in \mathbb{N}$, where the $t-1$ coefficients are chosen uniformly at random in $\mathbb{F}^{p \times q}$. Define $\tilde{\mathbf{R}}$ as

$$\tilde{\mathbf{R}} \triangleq (\mathbf{R}(\alpha_1), \mathbf{R}(\alpha_2), \dots, \mathbf{R}(\alpha_{t-1})),$$

where the $\alpha_1, \alpha_2, \dots, \alpha_{t-1} \in \mathbb{F}$ are distinct values. Then $\tilde{\mathbf{R}}$ has a uniform distribution over $\mathbb{F}^{p \times (t-1)q}$.

Proof. The proof directly follows from Lemma 5. \square

Lemma 7. Consider m polynomials $\mathbf{U}_1(x), \mathbf{U}_2(x), \dots, \mathbf{U}_m(x)$ of degree at most $n-1$ where their coefficients are chosen with arbitrary joint distribution from $\mathbb{F}^{p \times q}$. Let \mathcal{A} denotes the order set of those coefficients. Consider the polynomials

$$\begin{aligned} \mathbf{T}_1(x) &= \mathbf{U}_1(x) + x^n \mathbf{R}_1(x), \\ \mathbf{T}_2(x) &= \mathbf{U}_2(x) + x^n \mathbf{R}_2(x), \\ &\vdots \\ \mathbf{T}_m(x) &= \mathbf{U}_m(x) + x^n \mathbf{R}_m(x), \end{aligned} \tag{58}$$

where for $1 \leq i \leq m$, $\mathbf{R}_i(x): \mathbb{F} \rightarrow \mathbb{F}^{p \times q}$ is a polynomial of degree $\max(0, t-2)$, where the $t-1$ coefficients are chosen independently and uniformly at random from $\mathbb{F}^{p \times q}$. Then, $I(\mathcal{A}; \tilde{\mathbf{T}}) = 0$, where $\tilde{\mathbf{T}}$ defined as

$$\tilde{\mathbf{T}} \triangleq \begin{bmatrix} \mathbf{T}_1(\beta_1) & \mathbf{T}_1(\beta_2) & \dots & \mathbf{T}_1(\beta_{t-1}) \\ \mathbf{T}_2(\beta_1) & \mathbf{T}_2(\beta_2) & \dots & \mathbf{T}_2(\beta_{t-1}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{T}_m(\beta_1) & \mathbf{T}_m(\beta_2) & \dots & \mathbf{T}_m(\beta_{t-1}) \end{bmatrix}, \quad (59)$$

$$(60)$$

for some arbitrary values $\beta_1, \beta_2, \dots, \beta_{t-1} \in \mathbb{F}$.

Proof. Let us define

$$\begin{aligned} \tilde{\mathbf{U}} &\triangleq \begin{bmatrix} \mathbf{U}_1(\beta_1) & \mathbf{U}_1(\beta_2) & \dots & \mathbf{U}_1(\beta_{t-1}) \\ \mathbf{U}_2(\beta_1) & \mathbf{U}_2(\beta_2) & \dots & \mathbf{U}_2(\beta_{t-1}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{U}_m(\beta_1) & \mathbf{U}_m(\beta_2) & \dots & \mathbf{U}_m(\beta_{t-1}) \end{bmatrix}, \\ \tilde{\mathbf{R}} &\triangleq \begin{bmatrix} \mathbf{R}_1(\beta_1) & \mathbf{R}_1(\beta_2) & \dots & \mathbf{R}_1(\beta_{t-1}) \\ \mathbf{R}_2(\beta_1) & \mathbf{R}_2(\beta_2) & \dots & \mathbf{R}_2(\beta_{t-1}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{R}_m(\beta_1) & \mathbf{R}_m(\beta_2) & \dots & \mathbf{R}_m(\beta_{t-1}) \end{bmatrix}. \end{aligned}$$

For any $\mathbf{T}, \mathbf{U} \in \mathbb{F}^{mp \times (t-1)q}$ we have

$$\begin{aligned} \Pr(\tilde{\mathbf{U}} = \mathbf{U} | \tilde{\mathbf{T}} = \mathbf{T}) &\stackrel{(a)}{=} \frac{\Pr(\tilde{\mathbf{T}} = \mathbf{T} | \tilde{\mathbf{U}} = \mathbf{U}) \Pr(\tilde{\mathbf{U}} = \mathbf{U})}{\sum_{\mathbf{U}_j \in \mathbb{F}^{mp \times (t-1)q}} \Pr(\tilde{\mathbf{T}} = \mathbf{T} | \tilde{\mathbf{U}} = \mathbf{U}_j) \Pr(\tilde{\mathbf{U}} = \mathbf{U}_j)} \\ &= \frac{\Pr(\tilde{\mathbf{R}} = \mathbf{T} - \mathbf{U} | \tilde{\mathbf{U}} = \mathbf{U}) \Pr(\tilde{\mathbf{U}} = \mathbf{U})}{\sum_{\mathbf{U}_j \in \mathbb{F}^{mp \times (t-1)q}} \Pr(\tilde{\mathbf{R}} = \mathbf{T} - \mathbf{U}_j | \tilde{\mathbf{U}} = \mathbf{U}_j) \Pr(\tilde{\mathbf{U}} = \mathbf{U}_j)} \\ &\stackrel{(b)}{=} \frac{\Pr(\tilde{\mathbf{R}} = \mathbf{T} - \mathbf{U}) \Pr(\tilde{\mathbf{U}} = \mathbf{U})}{\sum_{\mathbf{U}_j \in \mathbb{F}^{mp \times (t-1)q}} \Pr(\tilde{\mathbf{R}} = \mathbf{T} - \mathbf{U}_j) \Pr(\tilde{\mathbf{U}} = \mathbf{U}_j)} \\ &= \frac{\Pr(\tilde{\mathbf{U}} = \mathbf{U})}{\sum_{\mathbf{U}_j \in \mathbb{F}^{mp \times (t-1)q}} \Pr(\tilde{\mathbf{U}} = \mathbf{U}_j)} \\ &= \Pr(\tilde{\mathbf{U}} = \mathbf{U}), \end{aligned}$$

where (a) follows from Bayesian Rule, (b) follows from the fact that according to Corollary 6, each row of matrix $\tilde{\mathbf{R}}$ has a uniform distribution over $\mathbb{F}^{p \times (t-1)q}$, thus $\tilde{\mathbf{R}}$ has a uniform distribution over $\mathbb{F}^{mp \times (t-1)q}$, which implies that $\Pr(\tilde{\mathbf{R}} = \mathbf{T} - \mathbf{U}) = \Pr(\tilde{\mathbf{R}} = \mathbf{T} - \mathbf{U}_j) = \frac{1}{|\mathbb{F}|^{mp(t-1)q}}$. Thus, we have $H(\tilde{\mathbf{U}} | \tilde{\mathbf{T}}) = H(\tilde{\mathbf{U}})$. Therefore $I(\tilde{\mathbf{U}}; \tilde{\mathbf{T}}) = 0$.

On the other hand, from the definition of \mathcal{A} , one can see that $\mathcal{A} \rightarrow \tilde{\mathbf{U}} \rightarrow \tilde{\mathbf{T}}$ is a Markov chain. Thus, according to data processing inequality, we have

$$I(\mathcal{A}; \tilde{\mathbf{T}}) \leq I(\tilde{\mathbf{U}}; \tilde{\mathbf{T}}) = 0.$$

Therefore, if the adversaries know the elements of $\tilde{\mathbf{T}}$, they are not able to gain any information about the elements of \mathcal{A} . \square

Corollary 8. Assume that

$$\mathbf{Y} \triangleq [\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k],$$

where $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k \in \mathbb{F}^{m \times \frac{m}{k}}$. Assume that \mathbf{Y} is shared using polynomial function

$$\mathbf{F}_{\mathbf{Y}, b, t, k}(x) = \sum_{j=1}^k \mathbf{Y}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1},$$

where $\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1}$ are chosen independently and uniformly at random in $\mathbb{F}^{m \times \frac{m}{k}}$, with N workers, i.e., $\mathbf{F}_{\mathbf{Y}, b, t, k}(\alpha_n)$ is delivered to worker n , $n \in [N]$. For any subset $\mathcal{S} \subset [N]$, $|\mathcal{S}| = t-1$, we have

$$\begin{aligned} H(\mathbf{Y} | \mathbf{F}_{\mathbf{Y}, b, t, k}(\alpha_i), i \in \mathcal{S}) &= H(\mathbf{Y}), \\ H(\mathbf{F}_{\mathbf{Y}, b, t, k}(\alpha_i), i \in \mathcal{S} | \mathbf{Y}) &= H(\mathbf{F}_{\mathbf{Y}, b, t, k}(\alpha_i), i \in \mathcal{S}). \end{aligned}$$

Proof. The proof follows directly from Lemma 7. \square

Intuitively, we can explain the privacy constraints as follows. If we consider any subset of $t-1$ workers, each share that they receive from the sources or any other workers includes contribution $t-1$ random matrices, excluding the original data itself. Thus, if we ignore the original data, the number of equations and the number of variables are the same. However, because of original data, the number of equations is always less than the number of the variables, no matter how data is involved in these equations. Thus, the adversaries cannot gain any information about the private inputs.

Now we formally prove the privacy constraints (4) and (5), for Algorithm (6). Assume that the semi-honest workers are $i_1, i_2, \dots, i_{t-1} \in [N]$. In order to prove constraint (4), for Algorithm 6, we must show that for any $\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]} \in \mathbb{F}^{m \times m}$, and polynomial function $\mathbf{G} : (\mathbb{F}^{m \times m})^\Gamma \rightarrow \mathbb{F}^{m \times m}$ we have

$$H(\mathbf{X}^{[j]}, j \in [\Gamma] | \bigcup_{n \in \mathcal{S}} \{\mathcal{M}_{n' \rightarrow n}, n' \in [N]\}, \tilde{\mathbf{X}}_{\gamma n}, \gamma \in [\Gamma], n \in \mathcal{S}) = H(\mathbf{X}^{[j]}, j \in [\Gamma]),$$

where $\mathcal{S} = \{i_1, i_2, \dots, i_{t-1}\}$. Assume that the calculations is done through R rounds. Let us define $\mathcal{M}_{n' \rightarrow n}^{(r)}$ as the messages sent from worker n' to worker n , in round $r \in [R]$. Thus

$$\mathcal{M}_{n' \rightarrow n} = \bigcup_{r=1}^R \mathcal{M}_{n' \rightarrow n}^r.$$

Also define $\mathcal{R}_{n' \rightarrow n}^{(r)}$ as the set of all random matrices that worker n' uses for sending a message to worker n , in round r . Let us define

$$\begin{aligned} \mathcal{M}_S^{(r)} &\triangleq \bigcup_{n' \in [\Gamma], n \in \mathcal{S}} \mathcal{M}_{n' \rightarrow n}^{(r)}, \\ \mathcal{R}_S^{(r)} &\triangleq \bigcup_{n' \in [\Gamma], n \in \mathcal{S}} \mathcal{R}_{n' \rightarrow n}^{(r)}, \\ \mathcal{X} &\triangleq \{\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]}\}, \\ \tilde{\mathcal{X}}_S &\triangleq \{\tilde{\mathbf{X}}_{\gamma n}, \gamma \in [\Gamma], n \in \mathcal{S}\}. \end{aligned}$$

From the definition, we have

$$\begin{aligned} H(\mathbf{X}^{[j]}, j \in [\Gamma] | \bigcup_{n \in \mathcal{S}} \{\mathcal{M}_{n' \rightarrow n}, n' \in [N]\}, \tilde{\mathbf{X}}_{\gamma n}, \gamma \in [\Gamma], n \in \mathcal{S}) &= \\ H(\mathcal{X} | \tilde{\mathcal{X}}_S, \mathcal{M}_S^{(r)}, r \in [R]). \end{aligned}$$

Therefore, to prove the privacy constraint (4), it is sufficient to show that

$$I(\tilde{\mathcal{X}}_S, \mathcal{M}_S^{(1)}, \mathcal{M}_S^{(2)}, \dots, \mathcal{M}_S^{(R)}; \mathcal{X}) = 0.$$

One can see that

$$\begin{aligned}
H(\tilde{\mathcal{X}}_S, \mathcal{M}_S^{(1)}, \mathcal{M}_S^{(2)}, \dots, \mathcal{M}_S^{(R)} | \mathcal{X}) &= H(\tilde{\mathcal{X}}_S | \mathcal{X}) + H(\mathcal{M}_S^{(1)} | \tilde{\mathcal{X}}_S, \mathcal{X}) \\
&\quad + H(\mathcal{M}_S^{(2)} | \mathcal{M}_S^{(1)}, \tilde{\mathcal{X}}_S, \mathcal{X}) \\
&\quad \vdots \\
&\quad + H(\mathcal{M}_S^{(R)} | \mathcal{M}_S^{(R-1)}, \dots, \mathcal{M}_S^{(1)}, \tilde{\mathcal{X}}_S, \mathcal{X}).
\end{aligned} \tag{61}$$

According to Corollary 8, we have $H(\tilde{\mathcal{X}}_S | \mathcal{X}) = H(\tilde{\mathcal{X}}_S)$. In addition, since in each round $r \in [R]$, $\mathcal{M}_S^{(r)}$ is a function of $\tilde{\mathcal{X}}_S$, $\mathcal{M}_S^{(1)}$, $\mathcal{M}_S^{(2)}$, $\mathcal{M}_S^{(r-1)}$, and $\mathcal{R}_S^{(r)}$, then

$$\begin{aligned}
H(\mathcal{M}_S^{(r)} | \mathcal{M}_S^{(r-1)}, \dots, \mathcal{M}_S^{(1)}, \tilde{\mathcal{X}}_S, \mathcal{X}) &= H(\mathcal{R}_S^{(r)} | \mathcal{M}_S^{(r-1)}, \dots, \mathcal{M}_S^{(1)}, \tilde{\mathcal{X}}_S, \mathcal{X}) \\
&= H(\mathcal{R}_S^{(r)}) \stackrel{(a)}{\geq} H(\mathcal{M}_S^{(r)}),
\end{aligned} \tag{62}$$

where (a) follows from the fact that $H(\mathcal{R}_S^{(r)})$ and $H(\mathcal{M}_S^{(r)})$ have the same size and $\mathcal{R}_S^{(r)}$ has a uniform distribution. According to (61) and (62) we have

$$\begin{aligned}
H(\tilde{\mathcal{X}}_S, \mathcal{M}_S^{(1)}, \mathcal{M}_S^{(2)}, \dots, \mathcal{M}_S^{(R)} | \mathcal{X}) &\geq H(\tilde{\mathcal{X}}_S) + H(\mathcal{M}_S^{(1)}) + H(\mathcal{M}_S^{(2)}) + \dots + H(\mathcal{M}_S^{(R)}) \\
&\geq H(\tilde{\mathcal{X}}_S, \mathcal{M}_S^{(1)}, \mathcal{M}_S^{(2)}, \dots, \mathcal{M}_S^{(R)}).
\end{aligned}$$

Therefore,

$$I(\tilde{\mathcal{X}}_S, \mathcal{M}_S^{(1)}, \mathcal{M}_S^{(2)}, \dots, \mathcal{M}_S^{(R)}; \mathcal{X}) \leq 0.$$

Thus

$$I(\tilde{\mathcal{X}}_S, \mathcal{M}_S^{(1)}, \mathcal{M}_S^{(2)}, \dots, \mathcal{M}_S^{(R)}; \mathcal{X}) = 0,$$

which proves the privacy constraint at the workers (4).

Now we prove constraint (5). Assume that

$$\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k],$$

where $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k \in \mathbb{F}^{m \times \frac{m}{k}}$. Since the result at the master is in the form of *polynomial sharing*, according to (19) there exist a polynomial function $\mathbf{F}_{\mathbf{Y}, b, t, k}(x)$, where $\mathbf{F}_{\mathbf{Y}, b, t, k}(\alpha_i) = \mathbf{O}_i, i \in [N]$. More precisely, according to (19), there are $\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1}$ with independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$, where

$$\begin{aligned}
\mathbf{F}_{\mathbf{Y}, b, t, k}(x) &= \sum_{j=1}^k \mathbf{Y}_j x^{b(j-1)} + \sum_{j=1}^{t-1} \mathbf{R}_j x^{k^2+j-1}, \\
\mathbf{F}_{\mathbf{Y}, b, t, k}(\alpha_n) &= \mathbf{O}_n.
\end{aligned} \tag{63}$$

One can see that according to Theorem 2, we have

$$H(\mathbf{Y} | \mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N) = 0, \tag{64}$$

$$H(\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1} | \mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N) = 0, \tag{65}$$

$$H(\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N | \mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1}) = 0. \tag{66}$$

Thus, we have

$$\begin{aligned}
H(\mathcal{X}|\mathbf{Y}, \mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N) &\stackrel{(a)}{=} H(\mathcal{X}|\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N) \\
&\stackrel{(b)}{=} H(\mathcal{X}|\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_N, \mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1}) \\
&\stackrel{(c)}{=} H(\mathcal{X}|\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1}) \\
&\stackrel{(d)}{=} H(\mathcal{X}|\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_k) \\
&= H(\mathcal{X}|\mathbf{Y})
\end{aligned}$$

where (a) follows from (64), (b) and (c) follow from (63), (65), and (66), and (d) follows from the fact that $\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_{t-1}$ have independent and uniform distribution in $\mathbb{F}^{m \times \frac{m}{k}}$ and independent from \mathcal{X} .

APPENDIX C

ARITHMETIC CIRCUITS

In this part, we describe some rules to create the arithmetic circuit corresponding to a function $G(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]})$ and a specific order of computation. We know that each polynomial function can be written as a sum of production terms

$$G(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]}) = \sum_{j=1}^M G_j(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \dots, \mathbf{X}^{[\Gamma]}), \quad (67)$$

where M is the number of the monomial terms, and G_j 's are monomial functions, for $j \in [M]$.

Example 7. Assume that the desired function G is

$$G(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \mathbf{X}^{[3]}, \mathbf{X}^{[4]}) = (\mathbf{X}^{[2]})^T (\mathbf{X}^{[1]})^2 \mathbf{X}^{[3]} + \mathbf{X}^{[2]} \mathbf{X}^{[4]} (\mathbf{X}^{[3]})^T.$$

According to the notations we have

$$\begin{aligned} G_1(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \mathbf{X}^{[3]}, \mathbf{X}^{[4]}) &= (\mathbf{X}^{[2]})^T (\mathbf{X}^{[1]})^2 \mathbf{X}^{[3]}, \\ G_2(\mathbf{X}^{[1]}, \mathbf{X}^{[2]}, \mathbf{X}^{[3]}, \mathbf{X}^{[4]}) &= \mathbf{X}^{[2]} \mathbf{X}^{[4]} (\mathbf{X}^{[3]})^T. \end{aligned}$$

◇

There are some rules to represent G by multiplication and addition gates. The rules are described in the following. For simplification we show a multiplication gate by an AND gate, and an addition gate by an OR gate.

1) *Rule 1:*

Assume that the function is in the form of $\prod_{j=1}^{\Gamma} \mathbf{X}^{[j]}$. In order to represent this function, first we multiply the last two matrices $(\mathbf{X}^{[\Gamma-1]}, \mathbf{X}^{[\Gamma]})$, then we multiply $\mathbf{X}^{[\Gamma-2]}$ to the result of the previous operation and so on. The order of computation is shown in Fig. 2.

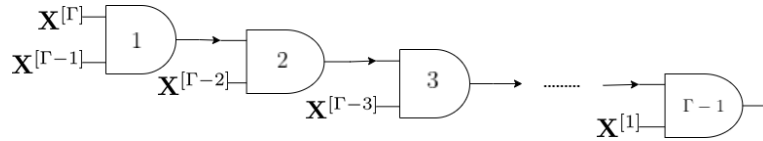


Fig. 2. The circuit representing the order of operations in calculating the function $G = \prod_{j=1}^{\Gamma} \mathbf{X}^{[j]}$.

2) *Rule 2:*

Assume that the function is in the form of $\sum_{j=1}^{\Gamma} \mathbf{X}^{[j]}$. In order to represent this function, first we add the last two matrices $(\mathbf{X}^{[\Gamma-1]} + \mathbf{X}^{[\Gamma]})$, then we add $\mathbf{X}^{[\Gamma-2]}$ to the result of the previous operation, and so on. The order of computation is shown in Fig. 3.

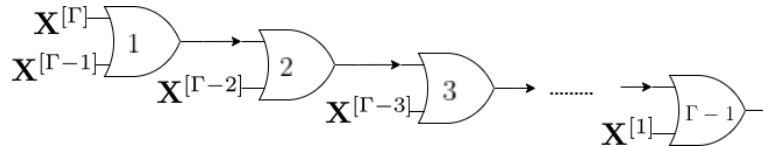


Fig. 3. The circuit representing the order of operations in calculating the function $G = \sum_{j=1}^{\Gamma} \mathbf{X}^{[j]}$.

3) *Rule 3:* To represent a general function (67), and assign a specific order to the computations, we first compute G_M based on Rule 1, keep the result, and compute G_{M-1} based on Rule 1, and add up the result based on rule 2, and then compute G_{M-2} based on Rule 1, and so on. The representation and order of computation are shown for an example in Fig.4.

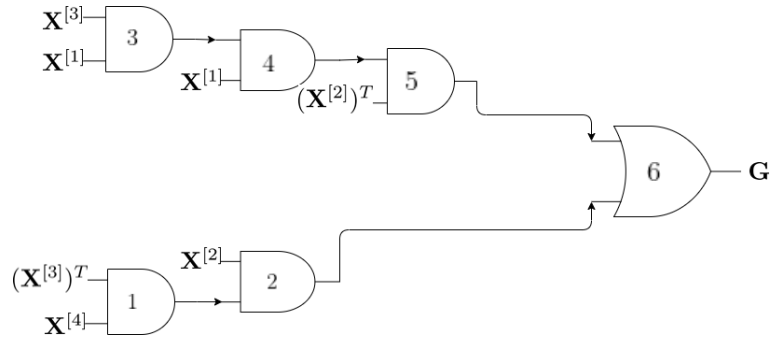


Fig. 4. The circuit representing the order of computation for function $G(X^{[1]}, X^{[2]}, X^{[3]}, X^{[4]}) = (X^{[2]})^T (X^{[1]})^2 X^{[3]} + X^{[2]} X^{[4]} (X^{[3]})^T$.