Decentralized Sparse Matrix Multiplication Under Byzantine Attacks

Sara Ghasvarianjahromi, Yauhen Yakimenka, Jörg Kliewer
Helen and John C. Hartmann Department of Electrical and Computer Engineering
New Jersey Institute of Technology, Newark, New Jersey, 07102, USA
Email: {sg273, yauhen.yakimenka, jkliewer}@njit.edu

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437858

***Abstract*—In this paper, we propose a sparse matrix multiplication in a decentralized setting, where a set of worker nodes wishes to compute a task collaboratively over a logical ring. We consider a subset of Byzantine nodes in the system who want to maliciously corrupt the result by corrupting their own computed blocks. In particular, the main focus of this paper is to compute the result with the least possible distortion by identifying the Byzantine nodes and re-assigning their tasks to the benign nodes. Our results demonstrate the feasibility of our proposed decentralized scheme and provide a trade-off between the computational complexity required at each worker node and the reconstruction distortion.**

***Index Terms*—Distributed matrix multiplication, decentralized computation, sparse matrices, byzantine attack**

I. Introduction

Recently, tensor operations such as matrix multiplication have emerged as an important ingredient of many signal processing and machine learning applications. These operations are often complex due to the large size of the associated matrices, even if these matrices in many cases are sparse, as, e.g., in recommender systems [1]. Thus, due to limited memory size and restricted computational capabilities, a server often is not able to perform these computations on its own. Therefore, the server typically partitions the input matrices into submatrices and offloads those to a set of worker nodes in the cloud. In such a system, the worker nodes compute their assigned task(s) in parallel and return it to the server, where the results are aggregated to obtain the multiplication result [2].

As these worker nodes are cloud-based and thus may not be trusted, there exists the threat of Byzantine attackers interfering arbitrarily with the computation of these worker nodes. This represents an important security bottleneck in distributed computation systems [3]. Also, verification of the results and identification of the attacked workers impose extra computational complexity on the server. In addition, the server, considered as a trusted entity by the worker nodes in the system, may be maliciously attacked as well [4].

An alternative is to consider a fully decentralized setting where the nodes exchange their partial computation results in such a way that each of them ends up with the complete result. An example of such a scenario is the internet-of-things (IoT) setting, where the IoT devices aim to complete the same computation individually, but rely on their neighbors to perform this computation due to their limited hardware capabilities. As in general such an environment is untrusted, the additional goal beyond computation is to detect and mitigate partial results provided by adversarial neighbors. While decentralized computation has been proposed in the context of decentralized learning (see, e.g., [5]–[9]), to the best of our knowledge, a fully decentralized matrix multiplication approach has not been addressed in the open literature. We aim to fill this void in this paper and, inspired by results from decentralized learning, propose a decentralized sparse matrix multiplication approach on a logical ring. Additionally, we take the potential presence of Byzantine nodes into account to address the security concerns in such a setting.

Note that in recent years the majority of the works in distributed matrix multiplication have focused on dense matrices. Specifically, it has been shown that encoding the input matrices via polynomial codes can improve the system performance in terms of latency and straggler tolerance [10]. Different coding schemes provide different trade-offs between the recovery threshold, i.e., the number of workers that have to complete their tasks before the server can recover the result, and the communication load, i.e., the amount of information to be downloaded from the workers (see, e.g., [2], [11]–[14]).

However, for sparse input matrices, encoding the submatrices in general decreases the sparsity of their coded representations sent to the worker nodes. This eliminates the complexity gains obtained by offloading the computation; for example, a coded sparse matrix multiplication scheme proposed in [15] achieves a sub-optimal recovery threshold. In [16], the authors consider a convolutional coding scheme and low-complexity peeling decoder for sparse distributed matrix-vector multiplication. The papers [17], [18] propose coding schemes for sparse matrix multiplication, which provides a trade-off between straggler resilience and worker computation speed. A coded scheme for distributed vector-matrix multiplication proposed in [19] is based on a secret sharing scheme and trades privacy guarantees with sparsity.

Similarly, distributed matrix multiplication in the presence of Byzantine attackers has been addressed recently for architectures with a centralized server. Specifically, in [20], [21], a distributed matrix-vector multiplication in presence of a subset of Byzantine workers is considered. In [20] a probabilistic

This work was in part supported by US NSF grants 1815322, 1908756, and 2107370.

scheme based on group testing is proposed to identify the attacked workers with high probability. Further, [22] presents a coded distributed computing scheme to preserve data security and privacy by improving the adversarial tolerance. A private and secure coded matrix-matrix multiplication scheme SRPM3 is proposed in [23], where Freivalds' test [24] is applied to detect the adversaries. However, little attention has been devoted to decentralized adversarial schemes.

In summary, our work has the following novel contributions beyond existing work in distributed matrix multiplication. (i) We extend the well-known setting with a trusted server to the fully decentralized case on a logical ring. Our scheme uses an uncoded scheme which preserves the computational savings at the workers for sparse input matrices. (ii) We propose a new algorithm to detect and mitigate Byzantine nodes in the fully decentralized setting under sparsity assumptions for the input matrices. The resulting algorithm is applied to both deterministic and probabilistic Byzantine adversaries and allows for both perfect recovery and an approximation of the correct matrix product at each node, depending on the number of active adversaries in the system.

Notation: We denote by $\mathbb{F}_q$ the finite field of size q. Matrix and vectors are denoted in boldface: $\mathbf{A}$ and $\mathbf{a}$. The transposition of matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$, and $\mathrm{col}_i(\mathbf{A})$ denotes the i-th column of $\mathbf{A}$. For a vector $\mathbf{a}$, we denote by $\mathrm{supp}(\mathbf{a})$ the set of indices of the non-zero entries of $\mathbf{a}$, and by $\mathrm{w_H}(\mathbf{a})$ its Hamming weight. The probability of an event A is denoted by $\mathbb{P}[A]$.

II. System model

Consider a computing system where N worker nodes $\mathcal{W} = \{\mathcal{W}_0, \ldots, \mathcal{W}_{N-1}\}$ need to compute the multiplication $\mathbf{C} = \mathbf{A}^T\mathbf{B}$ of two large sparse matrices $\mathbf{A} \in \mathbb{F}_q^{P\times S}$, and $\mathbf{B} \in \mathbb{F}_q^{P\times D}$. It is assumed that the sparsity levels of input matrices $\mathbf{A}$ and $\mathbf{B}$ are $\mathcal{L}(\mathbf{A})$ and $\mathcal{L}(\mathbf{B})$, respectively.

Definition 1. *The sparsity level of matrix $\mathbf{X} = (x_{ij}) \in \mathbb{F}_q^{m\times n}$, denoted by $\mathcal{L}(\mathbf{X})$, is the fraction of the number of zero elements with respect to the size of the matrix, i.e.,*

$$\mathcal{L}(\mathbf{X}) \triangleq \frac{|\{i,j : x_{ij} = 0\}|}{mn}. \tag{1}$$

We assume that all the worker nodes have enough memory capacity to store the entire input matrices $\mathbf{A}$ and $\mathbf{B}$, possibly in a compressed form[1]. As mentioned, the goal is to compute $\mathbf{C} = \mathbf{A}^T\mathbf{B}$, however, due to computational complexity constraints, each node $\mathcal{W}_n$ can compute only d_n multiplication tasks. This number in general depends on the size of the tasks. For the sake of simplicity, we assume that $d_n = d$ holds for all the nodes. However, the scheme can be easily adapted to the case of different d_n's. Therefore, the nodes employ a decentralized protocol where each node computes its corresponding task and sends it to its neighbor downstream, according to some predefined path. We generally assume that nodes can communicate on a complete graph where some links are associated with smaller communication costs than others, e.g., as in wireless networks. Therefore, some links are preferred for communication, and they may follow a logical ring as depicted in Fig. 1(a). Additionally, the ring topology eases the information exchange between the nodes to facilitate decentralized matrix multiplication.

To this end, the worker nodes partition the input matrices $\mathbf{A}$, $\mathbf{B}$ into Δ_A and Δ_B block-columns as

$$\mathbf{A} = \big[\mathbf{A}_0,\ \ldots, \mathbf{A}_{\Delta_A-1}\big],\ \mathbf{B} = \big[\mathbf{B}_0,\ \ldots,\ \mathbf{B}_{\Delta_B-1}\big], \tag{2}$$

where each block-column $\mathbf{A}_i$ is of size $P \times S'$, $S' = {}^{S}/_{\Delta_A}$, and $\mathbf{B}_j$ is of size $P \times D'$, $D' = {}^{D}/_{\Delta_B}$. We imply that S is divisible by Δ_A and that D is divisible by Δ_B, and thus both S' and D' are integers. With such a partitioning, the matrix $\mathbf{C} \in \mathbb{F}_q^{S\times D}$ consists of blocks $\mathbf{C}_{ij} = \mathbf{A}_i^T\mathbf{B}_j \in \mathbb{F}_q^{S'\times D'}$.

We consider a multiplication of two block-columns $\mathbf{A}_i^T\mathbf{B}_j$, $i \in [\Delta_A]$, $j \in [\Delta_B]$, to be one multiplication task. Hence, the total number of tasks is $\Delta_A\Delta_B = \Delta$. We use both double indexing $(i,j) \in [\Delta_A] \times [\Delta_B]$ and single indexing $n \in [\Delta]$ interchangeably, when it is not ambiguous.

We assume that the worker nodes can run their computations in parallel. Additionally, they are able to communicate in parallel, i.e., each node transmits and receives a bounded number of computed block-columns simultaneously within a fixed amount of time, which is synchronized across all the worker nodes.

Ideally, each node computes one block $\mathbf{C}_{ij} = \mathbf{A}_i^T\mathbf{B}_j$, and then the nodes exchange their results over the ring. However, there may exist an arbitrary subset of non-colluding Byzantine worker nodes (i.e., adversaries) $\{\mathcal{W}_n\}_{n\in\mathcal{A}}$ indexed by $\mathcal{A} \subset [N]$ with cardinality $|\mathcal{A}| = z$ in the system (the $N - z$ benign nodes remain benign during the whole process). The adversaries want to maliciously corrupt the result by sending wrong results $\tilde{\mathbf{C}}_{ij}$ to the next neighboring worker node. It is assumed that these Byzantine nodes can only manipulate their own computed blocks. This means that each worker node sends the read-only version of its computed task to the next worker node on the ring [26]. Note that the read-only requirement can be implemented by cryptographic tools.

We model the existence of adversaries via a probabilistic model. More precisely, each node $\mathcal{W}_n$ is adversarial with probability α, independently of others. The number of adversaries z is then a random variable (RV), distributed according to the binomial distribution with parameters N and α. Moreover,

$$\mathbf{C}_{ij} = \begin{cases} \mathbf{A}_i\mathbf{B}_j & \text{with probability } 1-\alpha, \\ \mathbf{Z}_{ij} & \text{with probability } \alpha. \end{cases} \tag{3}$$

To verify the results and also to detect the Byzantine worker nodes, we apply Freivalds' test [24] as outlined below.

A. Adversarial attack detection

Verifying the correctness of the results and also identifying the attacked worker nodes is a critical design issue in distributed computation systems. Freivalds' test [24] is a

[1]Compressed sparse row (CSR) and compressed sparse column (CSC) are the most common compression techniques for sparse matrix storage [25].

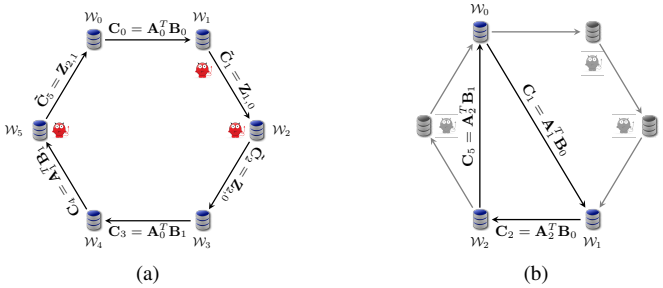

Fig. 1: Decentralized sparse matrix multiplication in the presence of Byzantine nodes.

well-known randomized method to verify the correctness of a matrix multiplication more efficiently than by recomputing the product. The algorithm is based on the following observation. If $\mathbf{A}_i^T\mathbf{B}_j \neq \mathbf{C}_{ij}$, then for a vector $\mathbf{v}$ drawn uniformly at random from $\mathbb{F}_q^D$, the probability that $\mathbf{A}_i^T\mathbf{B}_j\mathbf{v} = \mathbf{C}_{ij}\mathbf{v}$ is small. At the same time, computing this requires only three matrix-vector multiplications, instead of one matrix-matrix multiplication. Note that the time complexity of matrix-vector multiplication is much smaller than the complexity of matrix-matrix multiplication. The test can be amplified by drawing several independent vectors $\mathbf{v}$. The original paper [24] provides an upper bound on the failure probability (i.e., the misdetection probability) of the test for a general attack model. Since $\mathbf{Z}_{ij}$ is uniformly distributed in our model, we can improve on this probability as outlined in the following lemma. We also use random vectors $\mathbf{v}$ of a constant weight $w_H > 0$. The misdetection probability is the same for any fixed weight, and the multiplication by sparse vectors reduces complexity.

Lemma 1. *Assume $\mathcal{W}_n$ is an adversary. If we draw vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\zeta$ uniformly at random from $\{\mathbf{v} \in \mathbb{F}_q^{D'} : w_H(\mathbf{v}) = w_\mathbf{v}\}$, the probability of misdetection is*

$$P_m \triangleq \mathbb{P}\big[\mathbf{A}_i^T\mathbf{B}_j\mathbf{v}_k = \mathbf{Z}_{ij}\mathbf{v}_k, \forall k = 1, 2, \ldots, \zeta\big] = \frac{1}{q^{S'\zeta}}$$

Proof: The proof of the lemma follows along the lines of the original result by Freivalds [24] with the exploitation of the fact that the entries of matrix $\mathbf{Z}_{ij}$ are independent and uniform. Having ζ i.i.d vectors, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\zeta$, drawn uniformly at random from $\{\mathbf{v} \in \mathbb{F}_q^{D'} : w_H(\mathbf{v}) = w_\mathbf{v}\}$, and independent from the input matrices we have that

$$P_m = \big(\mathbb{P}\big[(\mathbf{A}_i^T\mathbf{B}_j - \mathbf{Z}_{ij})\mathbf{v}_1 = 0\big]\big)^\zeta,$$

Moreover, since $\mathbf{Z}_{ij}$ is uniform and independent of $\mathbf{A}_i$ and $\mathbf{B}_j$, the matrix $\mathbf{U} = \mathbf{A}_i^T\mathbf{B}_j - \mathbf{Z}_{ij}$ is uniform over $\mathbb{F}_q^{S'\times D'}$ ($\mathbf{Z}_{ij}$ acts as a one-time pad). For each fixed $\mathbf{v}$, there are precisely $q^{D'-1}$ vectors $\mathbf{u} \in \mathbb{F}_q^{D'}$, such that $\mathbf{u}^T\mathbf{v} = 0$ (as the number of solutions of the linear system $\mathbf{u}^T\mathbf{v} = 0$ which has rank 1 and D' variables). Therefore, there are $q^{(D'-1)S'}$ matrices $\mathbf{U}$ in $\mathbb{F}_q^{S'\times D'}$ such that $\mathbf{U}\mathbf{v} = 0$. In other words,

$$\mathbb{P}[\mathbf{U}\mathbf{v} = 0] = \frac{q^{(D'-1)S'}}{\left|\mathbb{F}_q^{S'\times D'}\right|} = \frac{1}{q^{S'}}.$$

and we obtain the statement of the lemma. ∎

Note that the misdetection probability decreases exponentially with the increase of the field size q.

III. Proposed Scheme

In this section, we present our proposed scheme in two variants: either perfect or imperfect reconstruction of the product $\mathbf{C}$, which we denote by $\hat{\mathbf{C}}$. In the proposed scheme, each node computes its assigned block multiplication task and shares the results over the ring in a sequence of parallel transmissions. Each node is responsible for verification of the calculation results from its upstream neighbors by employing Freivalds' test.

After the computed blocks have been distributed among all the nodes, the nodes skip all the identified adversaries, and thus the communication is performed over the smaller ring, albeit at a potentially higher communication cost.[2] The block multiplications that were assigned to the adversaries are reassigned between the remaining nodes, and they proceed in a similar manner. If the number of remaining multiplication tasks is larger than the number of benign nodes, this procedure may be repeated several times.

A. Perfect reconstruction

First, we describe the case of reconstructing the product $\mathbf{C}$ perfectly. This is possible if the total number of multiplication tasks the benign nodes can perform is larger than the number of the tasks, i.e., $|\mathcal{B}|d \geq \Delta$. The proposed scheme has multiple steps including computation, communication, and verification. Since every node has full access to $\mathbf{A}$ and $\mathbf{B}$, it can verify (via Freivands' test) any block multiplication result.

Task assignment: The workers form the pool of computational tasks $\mathcal{T}$, and each node is assigned one task from $\mathcal{T}$.

Computation: Each worker node computes the assigned task.

Verification and distribution: The steps below are performed in a synchronized fashion in parallel by all the nodes repeatedly until all the nodes receive all the blocks (exactly $N-1$ repetitions required). At step $t = 1, 2, \ldots, N-1$, a node

1) receives a new block forwarded by the upstream node;
2) performs Freivalds' test on the received block;
3) if the test fails, marks the node that produced this block as an adversary, and returns the corresponding computation task back to $\mathcal{T}$ for further computation;
4) forwards the received block further downstream.

After that, everybody has the same list of nodes marked adversarial since they all have tested the same computed blocks. Now, the benign nodes are renumbered in increasing order and skip all the adversaries, thus forming a smaller ring.

If there are still tasks left to compute and each benign node has performed less than d computations, the protocol execution goes back to the task assignment step, otherwise, it

[2]We do not consider communication cost in this work to show the feasibility of our proposed scheme and leave it for future work.

Algorithm 1: Proposed scheme

Input: matrices $\mathbf{A}$ and $\mathbf{B}$, Δ_A, Δ_B, node $\mathcal{W}_n$
Output: $\hat{\mathbf{C}}$

1 Partition $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B blocks, resp.
2 $\mathcal{A} \leftarrow \emptyset$, $\mathcal{B} \leftarrow [N]$, $\Delta \leftarrow \Delta_A\Delta_B$, $\mathcal{T} \leftarrow [0:\Delta-1]$
3 **for** $t = 1 : d$ **do**
4 $m_0, m_1, \ldots, m_{N-1} \leftarrow$ first N elements[3] of $\mathcal{T}$
5 $\mathcal{T} \leftarrow \mathcal{T} \setminus \{m_n\}$
6 Compute $\mathbf{C}_{m_n} = \mathbf{A}_{m_n \pmod{\Delta_A}}\mathbf{B}_{\lfloor m_n/\Delta_A \rfloor}$
7 Send $\mathbf{C}_{m_n}$ to the node downstream
8 **for** $i = 1 : N-1$ **do**
9 Receive $\mathbf{C}_{m_{n-i \pmod N}}$ from the node upstream
10 Run Freivalds' test on $\mathbf{C}_{m_{n-i \pmod N}}$
11 **if** *test succeeds* **then**
12 $\hat{\mathbf{C}}_{m_{n-i \pmod N}} \leftarrow \mathbf{C}_{m_{n-i \pmod N}}$
13 $\mathcal{T} \leftarrow \mathcal{T} \setminus \{m_{n-i \pmod N}\}$
14 **else**
15 $\mathcal{B} \leftarrow \mathcal{B} \setminus \{n - i \pmod N\}$
16 Forward $\mathbf{C}_{m_{n-i \pmod N}}$ to the node downstream
17 $N \leftarrow |\mathcal{B}|$, $n \leftarrow$ index of n in $\mathcal{B}$
18 Renumber benign nodes:
 $\mathcal{W}_0, \ldots, \mathcal{W}_{N-1} \leftarrow \{\mathcal{W}_m \mid m \in \mathcal{B}\}$
19 $\mathcal{B} \leftarrow [0 : N-1]$
20 **if** $\mathcal{T} = \emptyset$ **then**
21 goto Line 25
22 **if** $\mathcal{T} \neq \emptyset$ **then**
23 Replace $\gamma = \Delta - (N-z)d$ blocks with all-zero blocks
24 goto Line 25
25 **return** $\hat{\mathbf{C}}$

finishes. The detailed description of the scheme is presented in Algorithm 1. The same algorithm runs on every node, but each node $\mathcal{W}_n$ is aware of its index n in the system.

Proposition 1. *Algorithm 1 is resilient to at most $N - \lceil \Delta/d \rceil$ adversaries.*

Proof: Based on our construction, the total number of tasks that benign worker nodes are able to perform must be not smaller than the total number of tasks Δ, i.e., $(N-z)d \geq \Delta$, which proves the proposition. ∎

As a remark, once the node $\mathcal{W}_n$ detects $\mathcal{W}_m$ as an adversary, it can append the vector $\mathbf{v}$ to the corrupted block (i.e., $\mathbf{C}_m||\mathbf{v}$) and forward it along with the corrupted result to help the downstream nodes to run the Freivalds' test faster. Indeed, after such a vector $\mathbf{v}$ is found, it acts as a certificate that proves the incorrectness of the multiplication. We omit this optimization in the paper for the sake of clarity.

B. Imperfect reconstruction

For the case $|\mathcal{B}|d < \Delta$, perfect reconstruction of the multiplication result is not possible. In this case, $\gamma = \Delta - (N-z)d$ tasks cannot be reconstructed since this exceeds the computational capabilities of the worker nodes. Due to the sparse nature of the matrices, these γ blocks can be substituted with all-zero blocks in Line 23 of alg. 1. This substitution is the element-wise maximum likelihood estimate and it imposes a relatively small error/distortion to the result. It should be noted that unlike the scheme in Section III-A, this variant of the scheme never fails, even if $|\mathcal{B}|d < \Delta$.

IV. Reconstruction Distortion

In this section, we present the results on the average reconstruction distortion of the proposed scheme for random matrices $\mathbf{A}$ and $\mathbf{B}$, when the imperfect reconstruction variant outlined in Section III-B is used. We consider two sparsity models for the matrices $\mathbf{A}$ and $\mathbf{B}$: (i) with constant column weight and (ii) with i.i.d. entries distributed according to Bernoulli distribution. We measure the reconstruction distortion in terms of the normalized Hamming distance between the matrices.

Definition 2. *The normalized Hamming distance between two $m \times n$ matrices $\mathbf{X} = (x_{ij})$ and $\mathbf{Y} = (y_{ij})$ is defined as the number of positions where $\mathbf{X}$ and $\mathbf{Y}$ differ:*

$$\mathrm{d_H}(\mathbf{X}, \mathbf{Y}) \triangleq \frac{|\{i,j : x_{ij} \neq y_{ij}\}|}{mn}.$$

In both models below, the columns of $\mathbf{A}$ have the same (marginal) distributions, and this is also true for the columns of $\mathbf{B}$. In this case, the average distortion depends only on the expected sparsity level of the resulting matrix $\mathbf{C}$.

Theorem 1. *If the columns of $\mathbf{A}$ have the same marginal distributions (the same for the columns of $\mathbf{B}$), the proposed scheme in Algorithm 1 with imperfect reconstruction achieves the following expected distortion:*

$$\mathbb{E}\Big[\mathrm{d_H}(\mathbf{C}, \hat{\mathbf{C}})\Big] = \mathbb{E}[\mathcal{L}(\mathbf{C})] \times \sum_{z=N-\lceil \Delta/d \rceil}^{N} \binom{N}{z} \alpha^z (1-\alpha)^{N-z} \left(1 - \frac{(N-z)d}{\Delta}\right).$$

Proof: If every column in $\mathbf{A}$ and $\mathbf{B}$ has the same marginal distribution, then every element c_{ij} in matrix $\mathbf{C}$ has the same marginal distribution as the product of the i-th column in $\mathbf{A}$ and the j-th column in $\mathbf{B}$, for any choice of i and j, and $\mathbb{P}[c_{ij} = 0] = \mathbb{E}[\mathcal{L}(\mathbf{C})]$.

For a fixed number of adversaries $z > N - \lceil \Delta/d \rceil$, precisely $\gamma = \Delta - (N-z)d$ blocks of $\mathbf{C}$ will be not computed and they need to be substituted with all zeros. The average number of non-zero elements in these blocks is $\gamma S' D' \mathbb{E}[\mathcal{L}(\mathbf{C})]$ and this is also the average number of positions where $\mathbf{C}$ and $\tilde{\mathbf{C}}$ differ. Therefore, for a fixed number of adversaries z, the expected normalized Hamming distance between $\mathbf{C}$ and $\tilde{\mathbf{C}}$ is given as

$$\frac{\gamma S' D' \mathbb{E}[\mathcal{L}(\mathbf{C})]}{SD} = \mathbb{E}[\mathcal{L}(\mathbf{C})]\left(1 - \frac{(N-z)d}{\Delta}\right).$$

[3]If there are fewer tasks than remaining benign nodes, some of them do not perform computations but still participate in verification and distribution.

Finally, we need to average over the number of adversaries z, which is binomially distributed with parameters N and α. This yields the result of the theorem. ∎

Below, we derive the exact expressions for $\mathbb{E}[\mathcal{L}(\mathbf{C})]$ based on the two sparsity models for $\mathbf{A}$ and $\mathbf{B}$.

A. Constant column weight matrices

Let the matrix $\mathbf{A}$ be drawn uniformly at random from the set of all matrices in $\mathbb{F}_q^{P\times S}$ with constant column weight w_A and the matrix $\mathbf{B}$ be drawn from $\mathbb{F}_q^{P\times D}$ with column weight w_B, respectively. We assume that $\mathbf{A}$ and $\mathbf{B}$ are independent. With these distributions, the sparsity levels of the matrices are deterministic: $\mathcal{L}(\mathbf{A}) = 1 - {}^{w_A}/_{P}$ and $\mathcal{L}(\mathbf{B}) = 1 - {}^{w_B}/_{P}$. Note that the non-zero entries of the input matrices are i.i.d. over the multiplicative group of the field $\mathbb{F}_q$.

Lemma 2. *For random matrices $\mathbf{A}$ and $\mathbf{B}$ with constant column weights w_A and w_B, resp., the expected sparsity level of $\mathbf{C} = \mathbf{A}^T\mathbf{B}$ is*

$$\mathbb{E}[\mathcal{L}(\mathbf{C})] = \frac{\binom{P-w_A}{w_B}}{\binom{P}{w_B}} + \frac{1}{q-1}\sum_{\ell=2}^{\min(w_A,w_B)} \frac{\binom{w_A}{\ell}\binom{P-w_A}{w_B-\ell}}{\binom{P}{w_B}}. \quad (4)$$

Proof: The element in i-th row and j-th column of $\mathbf{C}$ is the dot product of $\mathrm{col}_i(\mathbf{A}^T)$ and $\mathrm{col}_j(\mathbf{B})$. For fixed i and j and random matrices, these columns are random vectors with fixed weights. Let ℓ be the size of the intersection of $\mathrm{supp}(\mathrm{col}_i(\mathbf{A}))$ and $\mathrm{supp}(\mathrm{col}_j(\mathbf{B}))$, i.e., the number of positions where both columns have non-zero values. Only these values are important for the result of the dot product.

If $\ell = 0$ (both supports have an empty intersection), the dot product is trivially 0. If $\ell = 1$, the dot product is never 0 as no product of two non-zero elements of $\mathbb{F}_q$ is 0. If $2 \le \ell \le \min(w_A, w_B)$, we have a dot product of two vectors of ℓ elements each, where all the elements are non-zero. Since these elements are drawn uniformly and independently from $\mathbb{F}_q \setminus \{0\}$, the probability of this product to be 0 is ${}^{1}/_{(q-1)}$.

The final ingredient of the proof is the distribution of ℓ. There are w_A non-zeros in $\mathrm{col}_i(\mathbf{A}^T)$, and $\binom{P}{w_A}$ ways to choose these positions. Among those w_A positions, there are $\binom{w_A}{\ell}$ ways to choose ℓ positions for the support intersection set. Finally, there are $\binom{P-w_A}{w_B-\ell}$ ways to choose the remaining non-zero positions of $\mathrm{col}_j(\mathbf{B})$. Altogether, the size of the support intersection set equals ℓ with probability

$$\frac{\binom{P}{w_A}\binom{w_A}{\ell}\binom{P-w_A}{w_B-\ell}}{\binom{P}{w_A}\binom{P}{w_B}} = \frac{\binom{w_A}{\ell}\binom{P-w_A}{w_B-\ell}}{\binom{P}{w_B}}.$$

∎

B. Matrices with i.i.d. entries

Again, we assume that $\mathbf{A}$ and $\mathbf{B}$ are drawn independently. Each element of $\mathbf{A}$ is 0 with probability $1 - \lambda_a$, and any other element of $\mathbb{F}_q$ with probability ${}^{\lambda_a}/_{(q-1)}$. Likewise, each element of $\mathbf{B}$ is 0 with probability $1-\lambda_b$ and any other element of $\mathbb{F}_q$ with probability ${}^{\lambda_b}/_{(q-1)}$. The expected sparsities are as follows:

$$\mathbb{E}[\mathcal{L}(\mathbf{A})] = 1 - \lambda_a, \quad \mathbb{E}[\mathcal{L}(\mathbf{B})] = 1 - \lambda_b.$$

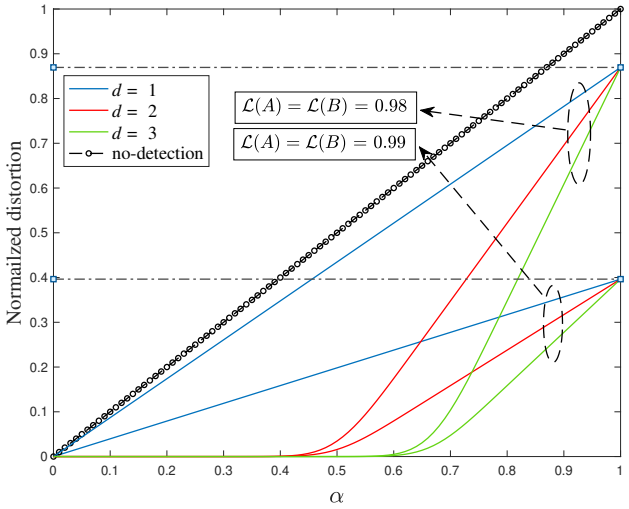

Fig. 2: Expected normalized distortion versus the probability that a node is adversarial

Lemma 3. *For random matrices $\mathbf{A}$ and $\mathbf{B}$ with independent entries, the expected sparsity level of $\mathbf{C} = \mathbf{A}^T\mathbf{B}$ is*

$$\mathbb{E}[\mathcal{L}(\mathbf{C})] = \left(1 - \frac{1}{q-1}\right)(1-\lambda)^P + \frac{1 - P\lambda(1-\lambda)^P}{q-1},$$

where $\lambda = \lambda_a\lambda_b$.

Proof: The proof of the lemma follows along the same lines as the proof of Lemma 2. The only difference is the distribution of the support intersection size ℓ: it follows a binomial distribution with parameters P and λ, ∎

V. Computational Complexity

The average computational complexity of the proposed scheme for each worker node can be summarized as follows. Considering $\mathcal{L}(\mathbf{A})$, $\mathcal{L}(\mathbf{B})$, and $\mathcal{L}(\mathbf{C})$ as the sparsity levels of the input matrices $\mathbf{A}$, $\mathbf{B}$, and the resulting matrix $\mathbf{C}$, resp., and by defining $\mathcal{L}_1 = \max(\mathcal{L}(A), \mathcal{L}(B))$, the computational complexity required at each worker node is at most $\mathcal{O}(S'(1-\mathcal{L}_1)PD')$ to compute its corresponding task. Note that this complexity is much lower in practice, due to the sparsity of the input matrices and also the uniform i.i.d. distribution of the non-zero elements in each column. Hence, we have $\mathrm{supp}(\mathrm{col}_i(\mathbf{A}) \cap \mathrm{col}_j(\mathbf{B})) \ll (1-\mathcal{L}_1)P$. The computational complexity of running one round of Freivalds' test for each node to verify one block-column is at most $\mathcal{O}(P(1-\mathcal{L}_2)D' + S'(1-\mathcal{L}_3)P + S'(1-\mathcal{L}_4)D')$, where $\mathcal{L}_2 = \max(\mathcal{L}(\mathbf{B}), \mathcal{L}(\mathbf{v}))$, $\mathcal{L}_3 = \max(\mathcal{L}(\mathbf{A}), \mathcal{L}_2)$, $\mathcal{L}_4 = \max(\mathcal{L}(\mathbf{C}), \mathcal{L}(\mathbf{v}))$. $\mathcal{L}(\mathbf{v}) = 1 - w_{\mathbf{v}}/D'$ denotes the sparsity level of vector $\mathbf{v}$.

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed scheme based on the analytical results derived in Section IV and compare them with the "no-detection" case, in which the environment is considered as fully trusted and the worker

nodes do not verify their neighboring nodes. The expected normalized distortion, considering the probabilistic model for an adversarial attack, is illustrated in Fig. 2. We consider z adversarial nodes, z being distributed according to a binomial distribution with parameters $N = 100$, the total number of nodes, and α, the probability of each node being adversarial. Therefore, the normalized distortion for the no-detection case is $\left(1-\frac{1}{q}\right)\alpha$.

Fig. 2 shows the results for different numbers of tasks that each worker node can perform, i.e., $d=1, 2, 3$, and two sparsity levels $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B})$ of 0.98 and 0.99 for input matrices $\mathbf{A}$ and $\mathbf{B}$ of size 5000×10000. The column weights are set to $w_A = w_B = 50$ and $w_A = w_B = 100$ for sparsity levels of 0.99 and 0.98, respectively. A moderate-size prime field $\mathbb{F}_q = \mathbb{F}_{2^{10}-3}$ is considered for this implementation. As can be seen from the figure, the adversarial tolerance increases significantly with the number of tasks that each worker node can perform. For example, for $d = 1$ no adversaries can be tolerated, but for $d = 2$, the system can tolerate roughly up to $z = {}^{N}/_{2}$ adversaries without any distortion in the final result. The effect of sparsity on the distortion can also be observed in Fig. 2. As shown, the straight dash-dotted lines in Fig. 2 depicts precisely the fraction of non-zero elements in the resulting matrix $\mathbf{C}$. Similar observations can be made for the i.i.d. entry case in Section IV-B. This is due to the fact that the reconstruction distortion is only a function of the sparsity level of $\mathbf{C}$. In fact, we can observe from Theorem 1 that the expected distortion only depends multiplicatively on $\mathcal{L}(\mathbf{C})$, and therefore the i.i.d. entry case shows qualitatively the same behavior as the constant column-weight case in Fig. 2.

VII. Conclusion

In this paper, we have proposed a new scheme to show the feasibility of sparse matrix multiplication in a decentralized manner. We have considered the potential presence of Byzantine nodes to address the security concerns in such a setting, and evaluated our proposed scheme by applying an adversarial detection method. Then the scheme was applied to both deterministic and probabilistic Byzantine adversaries for perfect and imperfect reconstruction of the matrix product. Our results demonstrated the feasibility of the proposed scheme and also show a significant performance improvement compared to the no-detection case.

References

[1] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, and H. Leung, "An efficient second-order approach to factorize sparse matrices in recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 946–956, 2015.

[2] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[3] S. Hong, H. Yang, and J. Lee, "Hierarchical group testing for Byzantine attack identification in distributed matrix multiplication," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 1013–1029, 2022.

[4] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[5] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," *Advances in Nneural Information Processing Systems*, vol. 30, 2017.

[6] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?" in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2348–2358.

[7] E. Cyffers and A. Bellet, "Privacy amplification by decentralization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 5334–5353.

[8] Z. Yang and W. U. Bajwa, "Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 611–627, 2019.

[9] A. R. Elkordy, S. Prakash, and S. Avestimehr, "Basil: A fast and Byzantine-resilient approach for decentralized training," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2694–2716, 2022.

[10] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2418–2422.

[11] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 1264–1270.

[12] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] 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.

[14] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the" cubic" barrier," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 245–250.

[15] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5152–5160.

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

[17] ——, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Transactions on Information Theory*, vol. 68, no. 6, pp. 4156–4181, 2022.

[18] ——, "An integrated method to deal with partial stragglers and sparse matrices in distributed computations," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 1010–1015.

[19] M. Xhemrishi, R. Bitar, and A. Wachter-Zeh, "Distributed matrix-vector multiplication with sparsity and privacy guarantees," *arXiv preprint arXiv:2203.01728*, 2022.

[20] A. Solanki, M. Cardone, and S. Mohajer, "Non-colluding attacks identification in distributed computing," in *2019 IEEE Information Theory Workshop (ITW)*. IEEE, 2019, pp. 1–5.

[21] S. Jain, M. Cardone, and S. Mohajer, "Identifying reliable machines for distributed matrix-vector multiplication," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 820–825.

[22] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-decodable coded computing: Breaking the adversarial toleration barrier," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 867–878, 2021.

[23] C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the Singleton bound," *IEEE Journal on Selected Areas in Information Theory*, 2022.

[24] R. Freivalds, "Fast probabilistic algorithms," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 1979, pp. 57–69.

[25] N. Srivastava, H. Jin, J. Liu, D. Albonesi, and Z. Zhang, "Matraptor: A sparse-sparse matrix multiplication accelerator based on row-wise product," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 766–780.

[26] K. Fu, M. F. Kaashoek, and D. Mazieres, "Fast and secure distributed read-only file system," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 1, pp. 1–24, 2002.