Decentralized Sparse Matrix Multiplication Under Byzantine Attacks

Sara Ghasvarianjahromi, Yauhen Yakimenka, and Jörg Kliewer

***Abstract*—Distributed computations, such as distributed matrix multiplication, can be vulnerable to significant security issues, notably Byzantine attacks. These attacks may target either worker nodes or servers, potentially leading to faulty results that can significantly degrade the overall performance. Therefore, detecting Byzantine attackers and mitigating their effects are crucial in such systems. Motivated by the goal of establishing a secure decentralized matrix-multiplication system, we first introduce a verification method named Common Tag, inspired by the well-known Freivalds' algorithm, able to verify the multiplication results independent of their associated input matrices. Then, we propose two schemes for sparse matrix multiplication where a group of nodes collaboratively performs a computation task over a logical ring. We consider a subset of Byzantine nodes in the system that may arbitrarily corrupt either their result or any other result passing through them. In Scheme I considering the highly sparse nature of input matrices, we assume that each node has sufficient capacity to store the entire input matrices, and the nodes forward the read-only versions of their computed blocks so that other nodes cannot corrupt them. In Scheme II, we relax the above assumptions, firstly, by considering a limited storage capacity for each node. Secondly, we introduce more powerful adversaries capable of corrupting other nodes' results by relaxing the read-only assumption. The results demonstrate the feasibility of both schemes and show a significant improvement in terms of distortion over the case where no detection happens. The results also provide a trade-off between the computational complexity required at each node and the reconstruction distortion in both schemes.**

***Index Terms*—Distributed matrix multiplication, decentralized computation, sparse matrices, Byzantine attacks.**

I. Introduction

A. Motivation and Problem Definition

RECENTLY, tensor operations such as matrix multiplication have emerged as an important ingredient in numerous 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, encodes them, and offloads them 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, decoded and the multiplication result is obtained [2], [3].

As these worker nodes are often cloud-based and thus may not be trusted, there exists the threat of Byzantine attackers interfering arbitrarily with the computation of these worker nodes [4], [5], [6]. This represents an important security bottleneck in distributed computation systems. Additionally, the centralized architecture poses some scalability limitations. For example, scaling the network size loads extra computational complexity on the server to verify the results and to identify the attacked worker nodes. Moreover, the server, often perceived as a trusted entity by the worker nodes in the system, can be a single point of failure or be maliciously attacked as well [7].

In certain networks, such as IoT networks, individual nodes may only possess partial observations of the input data. Additionally, due to the increasingly large size of data generated from various sources, nodes may lack sufficient storage capacity to retain the entire dataset [8]. Consequently, centralizing the data distribution becomes impractical within these networks. Another notable example of such networks arises in data-driven machine learning applications, such as recommender systems, where the maximum utilization of extensive data is crucial for accurate predictions and future event determination [9]. Therefore, to provide more coherent recommendations to the users, decentralized recommender systems were proposed to use information distributed across a network of nodes, with each node possessing only a partial observation of the data [10].

As a result, a promising alternative to the centralized setting in such networks is to consider fully decentralized settings where the nodes exchange their partial computation results to derive the complete result collectively. By "fully decentralized," we mean a system with no central server, ensuring that all computations and communications are performed by the participating nodes rather than relying on a single controlling entity. As in general, such an environment is untrusted, an additional goal beyond computation is to detect and mitigate the impact of adversarial nodes.

Received 23 July 2024; revised 28 March 2025; accepted 5 June 2025. Date of publication 18 June 2025; date of current version 18 July 2025. This work was supported in part by U.S. NSF under Grant 1815322, Grant 1908756, and Grant 2107370. An earlier version of this paper was presented in part at the IEEE Global Communications Conference (GLOBECOM), Kuala Lumpur, Malaysia, in December 2023 [DOI: 10.1109/GLOBECOM54140.2023.10437858]. The associate editor coordinating the review of this article and approving it for publication was Prof. Maryline Laurent. *(Corresponding author: Sara Ghasvarianjahromi.)*

The authors are with the Helen and John C. Hartmann Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: sg273@njit.edu; yauhen.yakimenka@njit.edu; jkliewer@njit.edu).

Digital Object Identifier 10.1109/TIFS.2025.3581053

To this end, our problem formulation is inspired by several practical scenarios. One notable example is in IoT networks, where multiple devices possessing partial sensor data must collaboratively process information without relying on a central server [11], [12]. For example, in [12], the authors introduced a decentralized framework specifically tailored for multi-hop IoT networks, which mitigates the bottleneck associated with central aggregation by facilitating local model exchanges between immediate neighbors. In this framework, the optimization process is driven by stochastic gradient descent (SGD), which involves large-scale matrix multiplications during both the forward and backward passes. Another significant example is healthcare systems, where sensitive data is distributed across various medical institutions [13], [14]. In this scenario, Lian et al. [13] proposed a hierarchical decentralized federated learning architecture, where local models trained on mobile healthcare devices are initially aggregated at institutional edge servers, which subsequently form a ring for model exchange, eliminating a single point of failure. Additionally, [14] proposed a decentralized federated learning network utilizing a ring of trusted and untrusted nodes for deep generative models in healthcare systems. Deep generative models rely heavily on large-scale matrix multiplications to perform the intensive linear transformations necessary during both training and inference. Distributing these matrix operations across a decentralized network can leverage parallel computation across nodes, enhancing scalability and robustness. However, this approach also introduces critical real-world challenges, such as effectively managing adversarial nodes without relying on global knowledge and ensuring consistent verification of results without relying on a centralized server.

Although fully decentralized computation has been proposed in the context of decentralized learning (see, e.g., [15], [16], [17], [18], [19]), to the best of our knowledge, a fully decentralized matrix multiplication approach was explored for the first time in our initial work [20]. Hence, inspired by the results from decentralized learning in this paper, we propose a decentralized sparse matrix multiplication approach on a logical ring under certain restrictions. We also consider the potential presence of Byzantine nodes to address security concerns in such a setting.

B. Related Work

To put our contribution in perspective, we briefly review prior related works. 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 [21]. 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], [22], [23], [24], [25], [26], [27]).

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 [28] achieves a sub-optimal recovery threshold. In [29], the authors consider a convolutional coding scheme and low-complexity peeling decoder for sparse distributed matrix-vector multiplication. The papers [30], [31] 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 [32] is based on a secret sharing scheme and trades privacy guarantees with sparsity. The authors in [33] propose a lower bound on the encoding weight (i.e., number of submatrices to be combined) to preserve the input sparsity as much as possible.

Similarly, distributed matrix multiplication in the presence of Byzantine attackers has been addressed recently for architectures with a centralized server and dense input matrices. Specifically, in [34], [35], a distributed matrix-vector multiplication in the presence of a subset of Byzantine workers is considered. In [34] a probabilistic scheme based on group testing is proposed to identify the attacked workers with high probability. Further, [36] presents a coded distributed computing scheme to preserve data security and privacy by improving adversarial tolerance. A private and secure coded matrix-matrix multiplication scheme SRPM3 is proposed in [37], where Freivalds' algorithm is applied to detect the adversaries. In recent work, [38], a general framework for linear secure distributed matrix multiplication (SDMM) is introduced to treat the straggling and Byzantine worker nodes in a centralized setting. However, little attention has been devoted to decentralized adversarial matrix multiplication problems.

C. Main Contribution

In summary, our work presents several novel contributions beyond existing research in distributed matrix multiplication. We first extend the well-known setting with a trusted server to the fully decentralized case over a logical ring. Despite conventional distributed systems with a central server, all computation, verification, and task re-computations are carried out by the individual nodes in our proposed schemes. This design eliminates a single point of failure and enhances robustness against Byzantine attacks. Furthermore, the ring-based approach ensures that all nodes share equal responsibility for verifying computations and reassigning tasks, even if each node only holds partial or systematically redundant data.

Also, our proposed algorithm employs an uncoded scheme to preserve computational efficiency at the worker nodes for sparse input matrices. In this process, we introduce a novel verification method called Common Tag, which is a modified version of the well-known Freivalds' algorithm [39], and allows for result verification in the absence of blocks of the two input matrices to be multiplied.

Furthermore, we propose two new schemes for decentralized sparse matrix multiplication, capable of detecting and mitigating the effects of Byzantine attacks. Unlike centralized systems, our approach relies on local consensus among all nodes. Specifically, every node actively participates in verifying computations, detecting misbehavior, and rerouting data.

Consequently, tasks initially assigned to a flagged adversarial node are reallocated to benign nodes that hold the corresponding input data. This collaborative, fully decentralized mechanism eliminates any single point of failure and enhances the network's overall robustness.

In Scheme I, we assume that all nodes have enough capacity to store the entire input matrices, which allows to use Freivalds' algorithm for result verification. Additionally, we assume that each node forwards a read-only version of its computed result to prevent corruption by Byzantine nodes. Next, we introduce Scheme II which relaxes the previous assumptions regarding storage capacity and read-only files. This scheme is more storage efficient, making it suitable even for dense input matrices. The Byzantine nodes are also able to corrupt any blocks arbitrarily in Scheme II. The proposed Common Tag method is employed for result verification in this scheme. We apply our proposed schemes to both deterministic and probabilistic Byzantine adversaries, allowing 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. Finally, we analyze and compare both schemes in terms of reconstruction distortion and illustrate the trade-off between computational complexity at each worker node and its reconstruction distortion.

In general, Scheme I offers the advantage of lower communication costs due to the direct application of Freivalds' algorithm for result verification, provided that each node has sufficient memory capacity to store the entire input matrices. However, this requirement for large memory capacity can be a significant drawback, especially in resource-constrained environments. Additionally, Scheme I restricts Byzantine nodes to corrupt only their own blocks, which may limit its effectiveness against more sophisticated adversaries. In contrast, Scheme II reduces the memory burden on each node, making it more suitable for systems with limited storage capacity. Note that this scheme also enables result verification without the presence of blocks of the two input matrices associated with the result. However, the trade-off is a higher total communication cost of the scheme, as nodes need to exchange more information to ensure accurate verification. Ultimately, the choice between Scheme I and Scheme II depends on the specific system requirements and constraints. Scheme I is ideal for scenarios where communication efficiency is paramount and memory capacity is not a limiting factor. On the other hand, Scheme II is better suited for environments with limited storage resources and where robust verification mechanisms are needed, despite the increased communication overhead.

D. Organization

The rest of this paper is organized as follows. In Section II, we describe the basic definitions and the system model. In Section III we describe the Freivalds' algorithm and introduce our verification method Common Tag, which is able to verify the block multiplication results without the requirement of existing the input blocks associated with the result. Section IV presents our proposed schemes for decentralized sparse matrix multiplication which can detect and mitigate the effect of Byzantine nodes. Section V analyzes the reconstruction distortion for two sparsity models of input matrices: with constant column weight, and with i.i.d. entries. Section VI summarizes the overall computational complexity of the schemes. In Section VII, we present numerical results to illustrate our analytical findings. Finally, the paper is concluded in Section VIII.

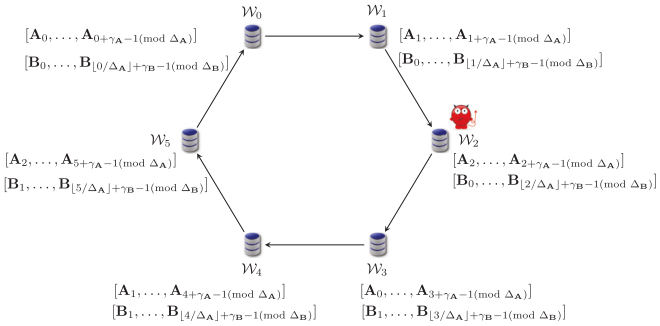


Fig. 1. System model and data distribution in the network of $N = 6$ nodes, with $\Delta_\mathbf{A} = 3, \Delta_\mathbf{B} = 2$, and an initial communication path (logical ring).

II. Problem Statement

A. Notation

We denote by $\mathbb{F}_q$ the finite field of size q. The set $[n]$ is defined as $\{0, 1, \ldots, n-1\}$. Matrices and vectors are denoted in boldface: $\mathbf{A}$ and $\mathbf{a}$, respectively. A zero vector is denoted by $\mathbf{0}$. The transposition of matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$. We denote i-th column of $\mathbf{A}$ by $\mathrm{col}_i(\mathbf{A})$. 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 $\mathbb{P}[A]$. We write $x_1, x_2, \ldots, x_n \sim \mathcal{U}(S)$ to denote that the objects $x_1, x_2, \ldots, x_n$ are drawn uniformly and independently at random from the set S.

B. System Model

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

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

Each node $\mathcal{W}_n$ has partially observed and stored the input data based on its accessibility and storage capacity. 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. Thus, some links are preferred for communication, and nodes may follow a Hamiltonian cycle with the minimum total communication cost (i.e., logical ring) with the node path sequence $(\mathcal{W}_0, \mathcal{W}_1, \ldots, \mathcal{W}_{N-1}, \mathcal{W}_0, \ldots)$, as depicted in Fig. 1. The ring topology eases the information exchange between the nodes

and also accelerates the process by leveraging parallel transmissions in a pipelined fashion. However, while we use a ring topology in this work for the sake of clarity, the core ideas, such as local verification, re-computation of corrupted blocks, and link reconfiguration remain applicable to more general network topologies with minor modifications. We also assume that the worker nodes are also allowed to broadcast small-sized vectors (much smaller than the sizes of the matrix blocks) at negligible cost.

To this end, the input matrices $\mathbf{A}$ and $\mathbf{B}$ are partitioned into $\Delta_{\mathbf{A}}$ and $\Delta_{\mathbf{B}}$ block-columns,[1] respectively:

$$\mathbf{A} = \left[\mathbf{A}_0, \ldots, \mathbf{A}_{\Delta_{\mathbf{A}}-1}\right], \quad \mathbf{B} = \left[\mathbf{B}_0, \ldots, \mathbf{B}_{\Delta_{\mathbf{B}}-1}\right], \tag{2}$$

where each block column $\mathbf{A}_i$ is of size $P \times S'$, $S' = S/\Delta_{\mathbf{A}}$, and $\mathbf{B}_j$ is of size $P \times D'$, $D' = D/\Delta_{\mathbf{B}}$. We imply that S is divisible by $\Delta_{\mathbf{A}}$ and that D is divisible by $\Delta_{\mathbf{B}}$, respectively, 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 multiplication of two block-columns $\mathbf{A}_i^T \mathbf{B}_j$, $i \in [\Delta_{\mathbf{A}}]$, $j \in [\Delta_{\mathbf{B}}]$, to be one multiplication task. Hence, the total number of multiplication tasks is $\Delta_{\mathbf{A}}\Delta_{\mathbf{B}} \triangleq \Delta$. We use both double indexing $(i, j) \in [\Delta_{\mathbf{A}}] \times [\Delta_{\mathbf{B}}]$ and single indexing $n \in [\Delta]$ interchangeably, when it is not ambiguous.

Suppose that each node $\mathcal{W}_n$ has access to a subset of size $\gamma_{\mathbf{A}} \leq \Delta_{\mathbf{A}}$ of block-columns from the input matrix $\mathbf{A}$, namely $\mathbf{A}_{n+l \pmod{\Delta_{\mathbf{A}}}}$, $l = \{0, \ldots, \gamma_{\mathbf{A}} - 1\}$, and a subset of size $\gamma_{\mathbf{B}} \leq \Delta_{\mathbf{B}}$ of block-columns from the input matrix $\mathbf{B}$, namely $\mathbf{B}_{\lfloor n/\Delta_{\mathbf{A}} \rfloor + l' \pmod{\Delta_{\mathbf{B}}}}$, $l' = \{0, \ldots, \gamma_{\mathbf{B}} - 1\}$, respectively. We assume that each node is potentially capable of performing $d \leq \gamma_{\mathbf{A}}\gamma_{\mathbf{B}}$ multiplication tasks, where d denotes the computational complexity constraint of each node. However, it is important to note that each node $\mathcal{W}_n$ is solely responsible for performing exactly one multiplication task $\mathbf{C}_{ij} = \mathbf{A}_i^T \mathbf{B}_j$, which implies $N = \Delta$.

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. However, minor delays are tolerable as long as the order of messages is preserved and no data is permanently lost. In such cases, each node can buffer incoming data and proceed to the next round only once all required messages have been received.

As mentioned, 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 $|\mathcal{A}| = z$ ($N - z$ benign nodes remain benign during the whole process). The adversaries want to maliciously corrupt the final result by arbitrarily sending incorrect results $\mathbf{Z}_{ij}$ to the next neighboring worker node. The assumption of non-colluding adversarial nodes is made to simplify both the design and analysis of our scheme, establishing a baseline for decentralized verification and re-computation without relying on a central server.

We model the existence of adversaries probabilistically. More precisely, each node can be adversarial with probability α, independently of others. The number of adversaries z is then a random variable (RV) distributed according to binomial distribution with parameters N and α. Moreover, the node responsible for computing $\mathbf{C}_{ij}$ produces

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

To verify the results and detect the Byzantine worker nodes, we apply one of the verification methods presented in the following section.

III. Adversarial Attack Detection

Verifying the correctness of the results and also identifying the adversarial worker nodes are critical design issues in distributed computation systems. In the following, we present a well-known matrix multiplication verification method, Freivalds' algorithm [39], and introduce our proposed method, named Common Tag. The latter is a modified version of Freivalds' algorithm, which does not require the input matrices associated with the result for verification.

A. Freivalds' Algorithm

Freivalds' algorithm is a well-known probabilistic method to verify the correctness of a matrix multiplication more efficiently than 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 $\mathbf{v} \sim \mathcal{U}\left(\mathbb{F}_q^{D'}\right)$, the probability that $\mathbf{A}_i^T \mathbf{B}_j \mathbf{v} = \mathbf{C}_{ij}\mathbf{v}$ is small. At the same time, this verification requires only three matrix-vector multiplications instead of one matrix-matrix multiplication. Note that the time complexity of a matrix-vector multiplication is much smaller than the complexity of a matrix-matrix multiplication. The probability of misdetection (i.e., the probability of not detecting the incorrect block) for Freivalds' algorithm is given in the following lemma [37].

Lemma 1 ([37]): Assume $\mathcal{W}_n$ is an adversary and, thus, $\tilde{\mathbf{C}}_{ij} = \mathbf{Z}_{ij} \neq \mathbf{A}_i^T \mathbf{B}_j$. Having ζ i.i.d. vectors $\mathbf{v}_{t_0}, \mathbf{v}_{t_1}, \ldots, \mathbf{v}_{t_{\zeta-1}} \sim \mathcal{U}\left(\mathbb{F}_q^{D'}\right)$, the probability of misdetection is at most

$$P_m \triangleq \mathbb{P}\left[\mathbf{A}_i \mathbf{B}_j \mathbf{v}_{t_k} = \mathbf{Z}_{ij}\mathbf{v}_{t_k}, \forall k \in [\zeta]\right] \leq \left(\frac{1}{q}\right)^{\zeta}.$$

While the original Freivalds' algorithm relies on the presence of input matrices for verification, we propose a modified version that eliminates this dependency. Our proposed verification method has a complexity comparable to the original algorithm (see Section VI). In the adapted version described below, we introduce tags for input blocks from both matrices $\mathbf{A}$ and $\mathbf{B}$. These tags are specifically designed to verify the result of each worker node, $\mathbf{C}_{ij}$ without requiring the presence of the associated input blocks, $\mathbf{A}_i$ and $\mathbf{B}_j$.

[1] Our ring-based verification can be extended to inner [40] or grid (2D) [41] partitioning as long as the sub-matrices are uniformly sized. This may change how sub-blocks are assigned, but the overall protocol, such as forwarding, verification, and re-computation upon detecting corruption, remains valid.

B. Common Tag

In this verification method, denoted as Common Tag, at iteration t, after each node $\mathcal{W}_n$ forwards the result to its downstream neighboring node, it generates two vectors $\mathbf{v}_{\mathbf{A}_{n,t}} \sim \mathcal{U}\left(\mathbb{F}_q^{S'}\right)$ and $\mathbf{v}_{\mathbf{B}_{n,t}} \sim \mathcal{U}\left(\mathbb{F}_q^{D'}\right)$ and broadcasts them to all other nodes in the ring. Note that the required communication cost to do this is much smaller than communicating the computed block $\mathbf{C}_{ij}$ downstream along the ring. Then, all the nodes aggregate all the vectors to create common vectors $\mathbf{v}_{\mathbf{A}_t} = \sum_{n=0}^{N-1} \mathbf{v}_{\mathbf{A}_{n,t}}$ and $\mathbf{v}_{\mathbf{B}_t} = \sum_{n=0}^{N-1} \mathbf{v}_{\mathbf{B}_{n,t}}$. Subsequently, each node $\mathcal{W}_n$ computes the tags $\mathcal{T}_{n,t}^{\mathbf{A}_i}$, and $\mathcal{T}_{n,t}^{\mathbf{B}_j}$ for each of its stored block-columns, $\mathbf{A}_i \in \{\mathbf{A}_{n+l \pmod{\Delta_\mathbf{A}}} \mid l \in [\gamma_\mathbf{A}]\}$ and $\mathbf{B}_j \in \{\mathbf{B}_{\lfloor n/\Delta_\mathbf{A} \rfloor + l' \pmod{\Delta_\mathbf{B}}} \mid l' \in [\gamma_\mathbf{B}]\}$ as outlined below[2]:

$$\mathcal{T}_{n,t}^{\mathbf{A}_i} = \mathbf{v}_{\mathbf{A}_t}^T \mathbf{A}_i^T \tag{4}$$

$$\mathcal{T}_{n,t}^{\mathbf{B}_j} = \mathbf{B}_j \mathbf{v}_{\mathbf{B}_t} \tag{5}$$

Since all the nodes use the common vectors $\mathbf{v}_{\mathbf{A}_t}$ and $\mathbf{v}_{\mathbf{B}_t}$, all the tags $\mathcal{T}_{n,t}^{\mathbf{A}_i}$ and $\mathcal{T}_{n,t}^{\mathbf{B}_j}$ computed by different nodes for the same input blocks must be the same. In other words, using the same $\mathbf{v}_{\mathbf{A}_t}$ and $\mathbf{v}_{\mathbf{B}_t}$ ensures that all the tags belonging to the same input block-column but computed by different nodes are equal. Therefore, a Byzantine node cannot broadcast an arbitrary tag; otherwise, it will be detected by benign nodes right away. For simplicity, we revise the notation as $\mathcal{T}_t^{\mathbf{A}_i} = \mathcal{T}_{n,t}^{\mathbf{A}_i}$ and $\mathcal{T}_t^{\mathbf{B}_j} = \mathcal{T}_{n,t}^{\mathbf{B}_j}$. Consequently, all the nodes broadcast their computed tags to every other node in the ring, incurring a negligible cost compared to communicating computed blocks downstream in the ring. Under the assumption of non-colluding adversaries, this scheme requires at least two equal tags for each input block-column $\mathbf{A}_i$ and $\mathbf{B}_j$. This ensures that other nodes can trust the generated tags and consider nodes with different tags as adversaries.

Next, each node $\mathcal{W}_n$ utilizes the tags $\mathcal{T}_t^{\mathbf{A}_u}$ and $\mathcal{T}_t^{\mathbf{B}_v}$ to verify its received block $\tilde{\mathbf{C}}_{uv}$, possibly without having the associated input matrices $\mathbf{A}_u$ and/or $\mathbf{B}_v$. The result of the verification test can be improved by repeating the whole process ζ times. The verification process is carried out as follows:

$$\mathcal{T}_t^{\mathbf{A}_u} \mathcal{T}_t^{\mathbf{B}_v} \stackrel{?}{=} \mathbf{v}_{\mathbf{A}_t}^T \tilde{\mathbf{C}}_{uv} \mathbf{v}_{\mathbf{B}_t} \tag{6}$$

If the equality in (6) does not hold, it indicates the result $\tilde{\mathbf{C}}_{uv}$ is corrupted with probability 1. On the contrary, if the aforementioned equality is satisfied, there is a high probability that the result is correct. The probability of misdetection for Common Tag is given in the following lemma.

Lemma 2: Assume $\mathcal{W}_n$ is an adversary and, thus, $\tilde{\mathbf{C}}_{ij} = \mathbf{Z}_{ij} \neq \mathbf{A}_i^T \mathbf{B}_j$. Having ζ pairs of vectors $\mathbf{v}_{\mathbf{A}_{t_0}}, \mathbf{v}_{\mathbf{A}_{t_1}}, \ldots, \mathbf{v}_{\mathbf{A}_{t_{\zeta-1}}} \sim \mathcal{U}\left(\mathbb{F}_q^{S'}\right)$ and $\mathbf{v}_{\mathbf{B}_{t_0}}, \mathbf{v}_{\mathbf{B}_{t_1}}, \ldots, \mathbf{v}_{\mathbf{B}_{t_{\zeta-1}}} \sim \mathcal{U}\left(F_q^{D'}\right)$, all vectors being independent, the probability of misdetection is at most

$$P_m \triangleq \mathbb{P}\left[\mathcal{T}_{t_k}^{\mathbf{A}_i} \mathcal{T}_{t_k}^{\mathbf{B}_j} = \mathbf{v}_{\mathbf{A}_{t_k}}^T \mathbf{Z}_{ij} \mathbf{v}_{\mathbf{B}_{t_k}}, \forall k \in [\zeta]\right] \leq \left(1 - \left(1 - \frac{1}{q}\right)^2\right)^{\zeta}.$$

[2]Over-the-air computation can be used to efficiently compute these tags [42].

Proof: The proof of the lemma follows along the lines of the original result by Freivalds [39]. Since all the vectors $\mathbf{v}_{\mathbf{A}_{t_k}}$ and $\mathbf{v}_{\mathbf{B}_{t_k}}$ are independent, we have

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

where $\mathbf{v}_{\mathbf{A}_{t_k}} \sim \mathcal{U}\left(\mathbb{F}_q^{S'}\right)$ and $\mathbf{v}_{\mathbf{B}_{t_k}} \sim \mathcal{U}\left(\mathbb{F}_q^{D'}\right)$ are independent. Let $\mathbf{Y} = \mathbf{A}_i^T \mathbf{B}_j - \mathbf{Z}_{ij} \neq \mathbf{0}$. For each fixed non-zero vector $\mathbf{y} \in \mathbb{F}_q^{D'}$, there are exactly $q^{D'-1}$ vectors $\mathbf{v}_{\mathbf{B}_{t_k}} \in \mathbb{F}_q^{D'}$, such that $\mathbf{y}^T \mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}$.[3] Since $\mathbf{Y} \neq \mathbf{0}$, at least one of its rows is a non-zero row vector. Therefore, there are at most $q^{D'-1}$ vectors $\mathbf{v}_{\mathbf{B}_{t_k}} \in \mathbb{F}_q^{D'}$ such that $\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}$, and

$$\mathbb{P}\left[\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}\right] \leq \frac{q^{D'-1}}{\left|\mathbb{F}_q^{D'}\right|} = \frac{1}{q}.$$

On the other hand, obviously, $\mathbb{P}\left[\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}\right] \geq 1/q^{D'}$, as $\mathbf{Y0} = \mathbf{0}$. Now, using the law of total probability,

$$\begin{aligned}
&\mathbb{P}\left[\mathbf{v}_{\mathbf{A}_{t_k}}^T \mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = 0\right] \\
&= \mathbb{P}\left[\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}\right] \mathbb{P}\left[\mathbf{v}_{\mathbf{A}_{t_k}}^T \mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = 0 \mid \mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}\right] \\
&\quad + \mathbb{P}\left[\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} \neq \mathbf{0}\right] \mathbb{P}\left[\mathbf{v}_{\mathbf{A}_{t_k}}^T \mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = 0 \mid \mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} \neq \mathbf{0}\right] \\
&= \mathbb{P}\left[\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}\right] \cdot 1 + \left(1 - \mathbb{P}\left[\mathbf{Y}\mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}\right]\right) \cdot \frac{1}{q} \\
&\leq \max_{\frac{1}{q^{D'}} \leq x \leq \frac{1}{q}} \left(x + (1-x)\frac{1}{q}\right) = \frac{1}{q} + \frac{q-1}{q^2}
\end{aligned}$$

and the statement of the lemma follows immediately. ∎

Algorithm 1 Common Tag

Input: $\mathcal{W}_n$, $n \in [N]$, $\Delta_\mathbf{A}$, $\Delta_\mathbf{B}$, $\gamma_\mathbf{A}$, $\gamma_\mathbf{B}$, round t
Output: CORRECT(0) or INCORRECT(1)

receive $\tilde{\mathbf{C}}_{uv}$
generate and broadcast $\mathbf{v}_{\mathbf{A}_{n,t}} \leftarrow \mathcal{U}\left(F_q^{S'}\right)$
generate and broadcast $\mathbf{v}_{\mathbf{B}_{n,t}} \leftarrow \mathcal{U}\left(F_q^{D'}\right)$
compute collaboratively $\mathbf{v}_{\mathbf{A}_t} = \sum_{m=0}^{N-1} \mathbf{v}_{\mathbf{A}_{m,t}}$, and $\mathbf{v}_{\mathbf{B}_t} = \sum_{m=0}^{N-1} \mathbf{v}_{\mathbf{B}_{m,t}}$
compute and broadcast $\mathcal{T}_t^{\mathbf{A}_i} = \mathbf{v}_{\mathbf{A}_t}^T \mathbf{A}_i^T$ **for** all $i \in \{\mathbf{A}_{n+l \pmod{\Delta_\mathbf{A}}} \mid l \in [\gamma_\mathbf{A}]\}$
compute and broadcast $\mathcal{T}_t^{\mathbf{B}_j} = \mathbf{B}_j \mathbf{v}_{\mathbf{B}_t}$ **for** all $j \in \{\mathbf{B}_{\lfloor n/\Delta_\mathbf{A} \rfloor + l' \pmod{\Delta_\mathbf{B}}} \mid l' \in [\gamma_\mathbf{B}]\}$
receive $\mathcal{T}_t^{\mathbf{A}_u}$ and $\mathcal{T}_t^{\mathbf{B}_v}$
compute $\mathbf{v}_{\mathbf{A}_t}^T \tilde{\mathbf{C}}_{uv} \mathbf{v}_{\mathbf{B}_t}$
if $\mathcal{T}_t^{\mathbf{A}_u} \mathcal{T}_t^{\mathbf{B}_v} \neq \mathbf{v}_{\mathbf{A}_t}^T \tilde{\mathbf{C}}_{uv} \mathbf{v}_{\mathbf{B}_t}$ **then**
  return INCORRECT ("1")
else
  return CORRECT ("0")

As seen from the lemma, the probability of misdetection decreases with the increase of the field size q. Note that this verification test requires only a single comparison in $\mathbb{F}_q$, as opposed to S' comparisons in Freivalds' algorithm. The details of this method are presented in Algorithm 1.

[3]This is the number of solutions of the linear system $\mathbf{y}^T \mathbf{v}_{\mathbf{B}_{t_k}} = \mathbf{0}$ which has rank 1 and D' variables.

TABLE I

COMPARISON OF COMMON TAG WITH FREIVALDS' ALGORITHM FOR A SINGLE NODE IN ROUND t (UNIT: FIELD ELEMENT)

	Common Tag	**Freivalds**
Communication complexity (broadcasting)	$S' + D' + P(\gamma_{\mathbf{A}} + \gamma_{\mathbf{B}})$	0
# of comparisons	1	S'
$\mathbf{P_m}$ **(upper bound)**	$\frac{1}{q} + \frac{q-1}{q^2}$	$\frac{1}{q}$

C. Comparison

We compare the Freivald's algorithm and Common Tag in Table I. The communication complexity for Common Tag in Table I is given in field elements, specifically, the random vectors $\mathbf{v}_{\mathbf{A},n,t} \in \mathbb{F}_q^{S'}$ and $\mathbf{v}_{\mathbf{B},n,t} \in \mathbb{F}_q^{D'}$, along with the tags for $\gamma_{\mathbf{A}}$ and $\gamma_{\mathbf{B}}$ input blocks. In terms of bits, this overhead is at worst $\big((S'+D')+(\gamma_{\mathbf{A}}+\gamma_{\mathbf{B}})P\big)\log_2(q)$ per node per round, since each element of $\mathbb{F}_q$ can be represented using $\log_2(q)$ bits. Next, the number of comparisons is the number of field elements that need to be compared for the result's verification. Finally, the third row of Table I shows the upper bound on the probability of misdetection P_m for each method. As can be seen, P_m for Common Tag has an additional term $\frac{q-1}{q^2}$ resulting from the multiplication of two random vectors by $\mathbf{C}_{ij}$, as opposed to the multiplication by one random vector in Freivalds' algorithm. However, this term is negligible for sufficiently large field size q.

Remark 1: To implement Common Tag more efficiently in terms of communication, each worker node $\mathcal{W}_n$ can broadcast two random seeds $R_{\mathbf{A}_{n,t}} \in \mathbb{F}_q$ and $R_{\mathbf{B}_{n,t}} \in \mathbb{F}_q$, respectively, instead of broadcasting the entire vectors $\mathbf{v}_{\mathbf{A}_{n,t}}$ and $\mathbf{v}_{\mathbf{B}_{n,t}}$. Each node then computes the average of all seeds as $R_{\mathbf{A}_t} = \sum_{n=0}^{N-1} R_{\mathbf{A}_{n,t}}$ and $R_{\mathbf{B}_t} = \sum_{n=0}^{N-1} R_{\mathbf{B}_{n,t}}$. Finally, the same $\mathbf{v}_{\mathbf{A}_t}$ and $\mathbf{v}_{\mathbf{B}_t}$ can be generated at each node using a pseudo random number generator [43].

IV. PROPOSED SCHEMES

We now describe our proposed schemes for sparse matrix multiplication in a fully decentralized setting under Byzantine attacks. In Scheme I, we assume that all the worker nodes have enough memory capacity to store the entire input matrices $\mathbf{A}$ and $\mathbf{B}$[4], i.e., $\gamma_{\mathbf{A}} = \Delta_{\mathbf{A}}$ and $\gamma_{\mathbf{B}} = \Delta_{\mathbf{B}}$. This assumption is made for simplicity and to demonstrate the feasibility of our approach in an idealized setting, serving as a baseline for performance evaluation before introducing more practical constraints. Nevertheless, we recognize that real-world distributed systems often have limited storage capacity [45], [46]. Therefore, we further relax this assumption in Scheme II, by considering a limited storage capacity per node, while preserving robustness against Byzantine attacks. Another important assumption is that the Byzantine nodes can only corrupt their own computed blocks. In other words, each worker node sends the read-only version of its computed multiplication task to its downstream neighboring node in the ring. The read-only requirement can be implemented by cryptographic tools [47].

[4]Possibly in a compressed form. Compressed sparse row (CSR) and compressed sparse column (CSC) are the most common compression techniques for sparse matrix storage [44].

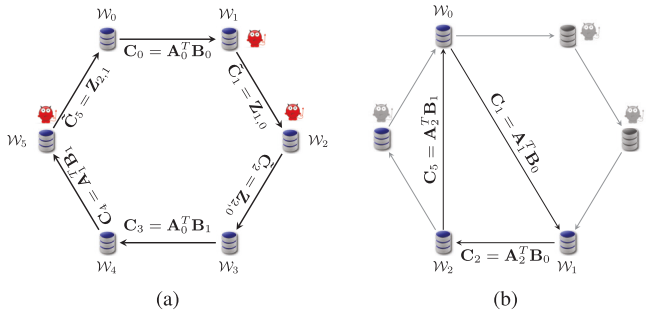


Fig. 2. Scheme I: decentralized sparse matrix multiplication in the presence of Byzantine nodes with $N = \Delta = 6$, $\gamma_{\mathbf{A}} = \Delta_{\mathbf{A}} = 3$ and $\gamma_{\mathbf{B}} = \Delta_{\mathbf{B}} = 2$. (a) Each node computes its task and forwards it to its downstream neighboring node. (b) The subset of benign nodes, $\mathcal{B}$, forms a smaller ring, recomputes the z remaining tasks, and communicates them over the new ring.

In Scheme II, we relax the assumptions of read-only files and the need to store the entire input matrices $\mathbf{A}$ and $\mathbf{B}$ at each node, making this scheme practically applicable even to dense input matrices. In this scheme, we consider a limited storage capacity for each node, i.e., $\gamma_{\mathbf{A}} < \Delta_{\mathbf{A}}$ and $\gamma_{\mathbf{B}} < \Delta_{\mathbf{B}}$. We also consider Byzantine nodes able to arbitrarily corrupt any blocks. Both schemes can produce either perfect or imperfect reconstruction of the result. The latter is the case when perfect reconstruction is not possible, and we aim to minimize the distortion of such reconstruction. We denote the reconstruction of the product $\mathbf{C}$ by $\hat{\mathbf{C}}$.

A. Scheme I

In Scheme I, each node computes its assigned multiplication task and shares the results over the ring in a sequence of parallel transmissions. Each node is responsible for the verification of the calculation results from its upstream neighbors by employing the verification 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 (see Fig. 2(a)). The block multiplications that were assigned to the adversaries are re-assigned among the remaining nodes, and they proceed similarly (see Fig. 2(b)). If the number of remaining multiplication tasks is larger than the number of benign nodes, this procedure may be repeated several times.

1) Perfect Reconstruction: Perfect reconstruction of the product $\mathbf{C}$ is possible if the total number of multiplication tasks all the benign nodes can perform is larger than the number of the tasks, i.e., $|\mathcal{B}|d \geq \Delta$. This scheme has multiple steps including computation, communication, and verification. Given that each node has complete access to the entire input matrices $\mathbf{A}$ and $\mathbf{B}$ in this scheme, every node can independently verify any block multiplication result using Freivalds' algorithm (see Lemma 1).

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

Computation: Each worker node computes the assigned task and forwards it to its downstream neighboring node (see Fig. 2(a)).

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 verification 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 $\mathcal{P}$ for further computation;
4) forwards the received block further downstream.

After that, every node 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.

Algorithm 2 Scheme I

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 $\mathcal{B} \leftarrow [N]$, $\Delta \leftarrow \Delta_A\Delta_B$, $\mathcal{P} \leftarrow [0 : \Delta - 1]$
3 **for** $t = 1 : d$ **do**
4 $m_0, m_1, \ldots, m_{N-1} \leftarrow$ first N elements[5] of $\mathcal{P}$.
5 $\mathcal{P} \leftarrow \mathcal{P} \setminus \{m_n\}$
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' algorithm 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 $\mathcal{P} \leftarrow \mathcal{P} \setminus \{m_{n-i \pmod N}\}$
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** $\mathcal{P} = \emptyset$ **then**
21 goto Line 24
22 **if** $\mathcal{P} \neq \emptyset$ **then**
23 Replace $\gamma = \Delta - (N-z)d$ blocks with all-zero blocks
24 **return** $\hat{\mathbf{C}}$

If there are still tasks in $\mathcal{P}$ left to compute and each benign node has performed less than d computations, the protocol execution goes back to the task assignment step. The same algorithm runs on every node in parallel, but each node $\mathcal{W}_n$ is aware of its index n in the system. See a detailed description of Scheme I in Algorithm 2.

Proposition 1: Algorithm 2 is resilient to at most $N - \lceil \Delta/d \rceil$ adversaries. In other words, if there are no more than $N - \lceil \Delta/d \rceil$ adversaries, the algorithm provides a perfect reconstruction of the matrix $\mathbf{C}$.

Proof: Based on our construction, the total number of tasks that benign worker nodes can 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 verification 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.

2) Imperfect Reconstruction: If $|\mathcal{B}|d < \Delta$, perfect reconstruction of $\mathbf{C}$ is not possible. In this case, $\gamma = \Delta - (N-z)d$ tasks cannot be reconstructed since this exceeds the computational capabilities of the benign worker nodes. Due to the sparse nature of the matrices, these γ blocks can be substituted with all-zero blocks (cf. Line 23 of Algorithm 2). This substitution is the element-wise maximum likelihood estimate of the correct blocks and it imposes a relatively small error/distortion on the result.

B. Scheme II

In Scheme II shown in Fig. 3, we relax the assumption of read-only files and the requirement for sufficient memory capacity to store the entire input matrices on each node. In this scheme, each node stores $\gamma_{\mathbf{A}} < \Delta_{\mathbf{A}}$ and $\gamma_{\mathbf{B}} < \Delta_{\mathbf{B}}$ block-columns from the input matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. We assume that adversarial nodes themselves never provide a correct result (i.e., they always corrupt their own output). A correct re-computation can be obtained from a benign node holding the same input blocks, albeit at the cost of higher computational overhead. This assumption ensures that our protocol remains robust even under worst-case adversarial behavior. However, if an adversary does occasionally produce a valid result, the ring accepts and forwards it, thereby reducing the re-computation overhead.

Similar to Scheme I, each node $\mathcal{W}_n$ computes its multiplication task $\mathbf{C}_n$ and forwards it to its downstream neighbor over the ring. Each node verifies the correctness of its received calculation result from the upstream node using the Common Tag method, presented in Section III-B. At the end of the final round, the nodes associated with missing blocks are identified as adversaries, and thus the remaining nodes form a smaller ring to recompute the missing blocks.

1) Perfect Reconstruction: We first describe the case of reconstructing the product $\mathbf{C}$ perfectly. This is possible if the total number of multiplication tasks Δ can be done by the set of benign nodes $\mathcal{B}$ based on their availability of input blocks. Since each node has partially observed the input matrices $\mathbf{A}$ ($\gamma_{\mathbf{A}}$ blocks from $\mathbf{A}$), and $\mathbf{B}$ ($\gamma_{\mathbf{B}}$ blocks from $\mathbf{B}$) in this

[5]If there are fewer tasks than in $\mathcal{P}$ the remaining benign nodes, some of them do not perform computations but still participate in verification and distribution.

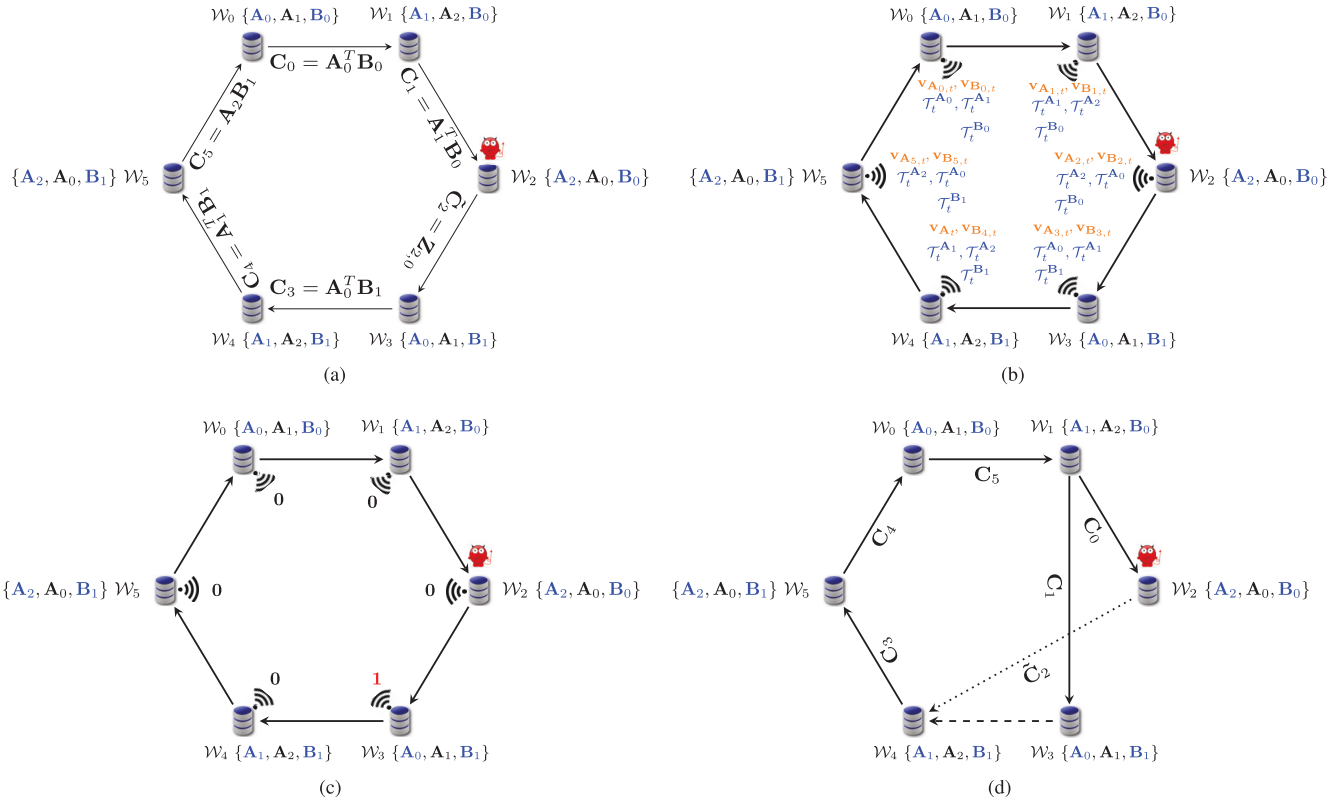


Fig. 3. Scheme II: decentralized sparse matrix multiplication in the presence of Byzantine nodes with $N = \Delta = 6$, $\Delta_{\mathbf{A}} = 3$, $\Delta_{\mathbf{B}} = 2$, $\gamma_{\mathbf{A}} = 2$ and $\gamma_{\mathbf{B}} = 1$. Each node (a) computes its task and forwards it to its downstream neighboring node, (b) first broadcasts random vectors and then broadcasts the computed tags, (c) performs the test and broadcasts the result, (d) continues forwarding the blocks via the former link (solid lines) and the temporarily activated link (dotted line). The link between $\mathcal{W}_3$ and $\mathcal{W}_4$ is temporarily deactivated (dashed line) since $\mathcal{W}_3$ discarded its received block from $\mathcal{W}_2$ and has no verified block to forward in this round.

scheme, the Common Tag method presented in Section III is utilized for verification. This scheme has multiple steps including computation, communication, and verification.

Task assignment: The worker nodes form a pool of computational tasks $\mathcal{P}$, and each node is assigned one task from $\mathcal{P}$ based on its observed input block-columns from the input matrices $\mathbf{A}$ and $\mathbf{B}$.

Computation and Communication: Each worker node computes its assigned task and forwards it to its downstream neighboring node (see Fig. 3(a)).

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. At step t, $t = 1, 2, \ldots, N-1$,

1) node $\mathcal{W}_n$ receives a new block forwarded by the upstream node, $\mathcal{W}_{n-1 \bmod N}$ (Fig. 3(a));
2) $\mathcal{W}_n$ generates $\mathbf{v}_{\mathbf{A}_{n,t}} \sim \mathcal{U}\left(\mathbb{F}_q^{S'}\right)$ and $\mathbf{v}_{\mathbf{B}_{n,t}} \sim \mathcal{U}\left(\mathbb{F}_q^{D'}\right)$ and broadcast them;
3) $\mathcal{W}_n$ gather all the vectors $\mathbf{v}_{\mathbf{A}_{m,t}}$ and $\mathbf{v}_{\mathbf{B}_{m,t}}$, $m \in [N] \setminus n$ to compute the common vectors $\mathbf{v}_{\mathbf{A}_t}$ and $\mathbf{v}_{\mathbf{B}_t}$, respectively, and subsequently computes the tags and broadcasts them (Fig. 3(b));
4) $\mathcal{W}_n$ performs verification tests on the received block from $\mathcal{W}_{n-1 \bmod N}$ using the received tags;
5) If verification fails, the following sequence of actions occur:

a) **Initial Actions upon Detected Corruption:**
- $\mathcal{W}_n$ discards the corrupted block immediately.
- $\mathcal{W}_n$ broadcasts "1" ("incorrect") to notify all nodes that node $\mathcal{W}_{n-1 \bmod N}$ is adversarial (see Fig. 3(c)).

b) **Local Link Adjustment (at node $\mathcal{W}_n$):**
- Node $\mathcal{W}_n$ now deactivates the link from the adversarial upstream node $\mathcal{W}_{n-1 \bmod N}$. In other words, node $\mathcal{W}_n$ itself refuses further communication with node $\mathcal{W}_{n-1 \bmod N}$ (see Fig. 3(d), deactivated link between $\mathcal{W}_2$ and $\mathcal{W}_3$).
- Instead, node $\mathcal{W}_n$ activates an alternative link with the next upstream node $\mathcal{W}_{n-2 \bmod N}$ (see Fig. 3(d), activated link between $\mathcal{W}_3$ and $\mathcal{W}_1$). After this reconfiguration, node $\mathcal{W}_n$ bypasses $\mathcal{W}_{n-1 \bmod N}$ and its new upstream neighbor becomes $\mathcal{W}_{n-2 \bmod N}$.

c) **Downstream Adjustment (to maintain ring connectivity):**
- Because node $\mathcal{W}_n$ discarded the corrupted block from $\mathcal{W}_{n-1 \bmod N}$, $\mathcal{W}_n$ now has no block to forward downstream in the next round.
- To preserve the connectivity of the ring and maintain a steady flow of the blocks, node $\mathcal{W}_{n-1 \bmod N}$'s temporarily activates the link to its next downstream neighbor $\mathcal{W}_{n+1 \bmod N}$

and directly forwards its computed block to $\mathcal{W}_{n+1 \bmod N}$ (see Fig. 3(d), activated link between $\mathcal{W}_2$ and $\mathcal{W}_4$).

- The direct link between $\mathcal{W}_{n-1 \bmod N}$ and $\mathcal{W}_{n+1 \bmod N}$ is activated until either node $\mathcal{W}_{n+1 \bmod N}$ independently detects $\mathcal{W}_{n-1 \bmod N}$ as adversarial or node $\mathcal{W}_n$ receives a correct block to resume forwarding.

Even though node $\mathcal{W}_n$ has flagged node $\mathcal{W}_{n-1 \bmod N}$ as adversarial, other nodes do *not* immediately trust $\mathcal{W}_n$'s accusation. Instead, they use $\mathcal{W}_n$'s broadcast ("1") only as a trigger for link reconfiguration. Each node *independently* verifies the correctness of the results from node $\mathcal{W}_{n-1 \bmod N}$ using its own verification processes and does not classify $\mathcal{W}_{n-1 \bmod N}$ as adversarial until corruption is confirmed. This ensures robustness against the situations in which node $\mathcal{W}_n$ itself is adversarial and might falsely flag its upstream node $\mathcal{W}_{n-1 \bmod N}$ to mislead other nodes. A node is universally recognized as adversarial only after multiple independent detections, representing a consensus among benign nodes. Thus, the scheme achieves robust decentralized verification based on consensus among nodes.

6) if the correctness of the result is verified, $\mathcal{W}_n$ forwards the received block further downstream.

Considering that adversaries always corrupt their blocks, the above communication and verification steps will repeat for $N-1$ rounds. After that, the nodes associated with missing blocks are identified and marked as adversaries for all other nodes (universally). Now, the remaining nodes are renumbered in increasing order, skip all the identified adversaries, thus forming a smaller ring and recompute the missing blocks. The detailed description of the scheme is presented in Algorithm 3.

Algorithm 3 Scheme II

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

$\mathcal{B} \leftarrow [N]$, $\Delta \leftarrow \Delta_A \Delta_B$, $\mathcal{P} \leftarrow [0 : \Delta - 1]$
for $t = 1 : \gamma_{\mathbf{A}}\gamma_{\mathbf{B}}$ **do**
  $m_0, m_1, \ldots, m_{N-1} \leftarrow$ first N elements of $\mathcal{P}$
  $\mathbf{A}_{m_n} \in \{\mathbf{A}_{n+l \pmod{\Delta_{\mathbf{A}}}} \mid l \in [\gamma_{\mathbf{A}}]\}$
  $\mathbf{B}_{m_n} \in \{\mathbf{B}_{\lfloor n/\Delta_{\mathbf{A}} \rfloor + l' \pmod{\Delta_{\mathbf{B}}}} \mid l' \in [\gamma_{\mathbf{B}}]\}$
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{m_n\}$
  Compute $\mathbf{C}_{m_n} = \mathbf{A}_{m_n \pmod{\Delta_A}} \mathbf{B}_{\lfloor m_n/\Delta_A \rfloor \pmod{\Delta_{\mathbf{B}}}}$
  Send $\mathbf{C}_{m_n}$ to the node downstream
  for $i = 1 : N-1$ **do**
    Receive $\mathbf{C}_{m_{n-i \pmod N}}$ from the node upstream $\mathcal{W}_{n-1 \pmod N}$
    Run Common Tag on $\mathbf{C}_{m_{n-i \pmod N}}$
    if *test succeeds* **then**
      $\hat{\mathbf{C}}_{m_{n-i \pmod N}} \leftarrow \mathbf{C}_{m_{n-i \pmod N}}$
      $\mathcal{P} \leftarrow \mathcal{P} \setminus \{m_{n-i \pmod N}\}$
      Forward $\mathbf{C}_{m_{n-i \pmod N}}$ to the node downstream
    else
      discards $\mathbf{C}_{m_{n-i \pmod N}}$
      $\mathcal{B} \leftarrow \mathcal{B} \setminus \{n - i \pmod N\}$
      $\mathcal{W}_{n-1 \pmod N} \leftarrow \mathcal{W}_{n-2 \pmod N}$
  $N \leftarrow |\mathcal{B}|$, $n \leftarrow$ index of n in $\mathcal{B}$
  Renumber benign nodes: $\mathcal{W}_0, \ldots, \mathcal{W}_{N-1} \leftarrow \{\mathcal{W}_m \mid m \in \mathcal{B}\}$
  $\mathcal{B} \leftarrow [0 : N-1]$
  if $\mathcal{P} = \emptyset$ **then**
    goto Line 27
if $\mathcal{P} \neq \emptyset$ **then**
  Replace γ blocks with all-zero blocks
return $\hat{\mathbf{C}}$

Proposition 2: Algorithm 3 is resilient to at least $\gamma_{\mathbf{A}}\gamma_{\mathbf{B}} - 1$ adversaries.

Proof: According to the construction, each worker node observes $\gamma_{\mathbf{A}}$ and $\gamma_{\mathbf{B}}$ block-columns from the input matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. Based on the system model, the nodes are organized into $N/\Delta_{\mathbf{A}} = \Delta_{\mathbf{B}}$ groups, ensuring that nodes within each group observe the same set of block-columns from $\mathbf{B}$. Therefore, each $\mathbf{B}_j$ is observed by all the nodes in $\frac{\gamma_{\mathbf{B}} N}{\Delta_{\mathbf{A}}\Delta_{\mathbf{B}}}$ different groups. Given that $N = \Delta = \Delta_{\mathbf{A}}\Delta_{\mathbf{B}}$, each B_j appears exactly in $\gamma_{\mathbf{B}}$ groups. Also, each $\mathbf{A}_i$ is observed by $\gamma_{\mathbf{A}}$ different nodes within each group. Hence, each task can be executed by $\gamma_{\mathbf{A}}\gamma_{\mathbf{B}}$ different nodes in the entire network. Therefore, to ensure at least one correct version of a specific computed task exists, this configuration can tolerate $\gamma_{\mathbf{A}}\gamma_{\mathbf{B}} - 1$ adversaries, regardless of their position in the ring, which proves the proposition. ∎

As explained in Proposition 2, perfect reconstruction of the result can be always obtained by the partial results from any $N - \gamma_{\mathbf{A}}\gamma_{\mathbf{B}} + 1$ nodes in the ring. However, depending on the specific positions of adversaries in the ring, the system might be able to tolerate more adversaries. In other words, it is possible to obtain perfect reconstruction of the result by having the partial results from different sets of benign nodes including a certain number of nodes from each group of the same $\mathbf{B}_j$, $j \in \{0, 1, \cdots, \Delta_{\mathbf{B}} - 1\}$, as shown in the following example.

Example: An illustrative scenario of Scheme II is shown in Fig. 3 with $N = 6$, $\Delta_{\mathbf{A}} = 3$, $\Delta_{\mathbf{B}} = 2$, $\gamma_{\mathbf{A}} = 2$, and $\gamma_{\mathbf{B}} = 1$.

We consider six nodes, $\{\mathcal{W}_0, \ldots, \mathcal{W}_5\}$, and partition $\mathbf{A}$ into blocks $\{\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2\}$ and $\mathbf{B}$ into $\{\mathbf{B}_0, \mathbf{B}_1\}$, resulting in a total of $\Delta = \Delta_{\mathbf{A}} \times \Delta_{\mathbf{B}} = 6$ multiplication tasks $\{\mathbf{C}_0, \ldots, \mathbf{C}_5\}$. Each node stores two blocks from $\mathbf{A}$ and one block from $\mathbf{B}$. For example, $\mathcal{W}_0$ holds $\{\mathbf{A}_0, \mathbf{A}_1, \mathbf{B}_0\}$ and $\mathcal{W}_1$ stores $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_0$.

- **Round 1:** Each node computes one assigned multiplication task $\mathbf{C}_n = \mathbf{A}^T_{n \bmod 3} \mathbf{B}_{\lfloor n/3 \rfloor}$, then forwards it to its downstream neighbor. Specifically, node $\mathcal{W}_0$ computes $\mathbf{C}_0 = \mathbf{A}_0^T \mathbf{B}_0$, node $\mathcal{W}_1$ computes $\mathbf{C}_1 = \mathbf{A}_1^T \mathbf{B}_0$, and so forth. Each node forwards its computed block to the next node in the ring (i.e., $\mathcal{W}_0 \rightarrow \mathcal{W}_1$, $\mathcal{W}_1 \rightarrow \mathcal{W}_2$, etc.). Upon receiving a block, each node immediately verifies it using the Common Tag method. For example, node $\mathcal{W}_3$ receives and verifies $\tilde{\mathbf{C}}_2$ from node $\mathcal{W}_2$. If verification fails (indicating corruption), the receiving node discards the

block, flags the sender as adversarial, and broadcasts "1" ("incorrect"). For instance, as depicted in Fig. 3(c), node $\mathcal{W}_3$ discards the corrupted block from node $\mathcal{W}_2$, flags $\mathcal{W}_2$ as adversarial, and broadcasts "1". This action triggers the network to reconfigure links, temporarily bypassing the adversarial node, as clearly illustrated in Fig. 3(d). If verification passes, the node continues forwarding the block downstream.

- **Rounds 2–5:** Over the next $N-2=4$ rounds, each block circulates through the ring, allowing independent verification by all nodes. This iterative verification ensures robust detection of corrupted outputs. If an adversarial node is detected (for example, node $\mathcal{W}_2$ in Fig. 3), alternate communication links are activated to maintain ring connectivity. For instance, after node $\mathcal{W}_3$ detects node $\mathcal{W}_2$ as adversarial, it establishes a direct link to node $\mathcal{W}_1$ (bypassing node $\mathcal{W}_2$) to receive future verified blocks. Also, $\mathcal{W}_2$ establishes a direct link to its next downstream node $\mathcal{W}_4$ and sends its computed block $\tilde{\mathbf{C}}_2$.

If a corrupted task is detected and discarded by all the nodes (universally), another benign node that stores the same corresponding input blocks $(\mathbf{A}_i, \mathbf{B}_j)$ recomputes the task after the final round, ensuring correctness and enabling recovery of the corrupted results.

In this scheme, perfect reconstruction of the result is achievable with the contributions from any $N-\gamma_{\mathbf{A}}\gamma_{\mathbf{B}}+1=5$ worker nodes, implying resilience to $\gamma_{\mathbf{A}}\gamma_{\mathbf{B}}-1=1$ adversary. However, perfect reconstruction also holds if any two nodes per $\mathbf{B}_j$ group are benign (e.g., $\{\mathcal{W}_0, \mathcal{W}_1, \mathcal{W}_3, \mathcal{W}_4\}$), showing resilience to 2 adversaries. Thus, the exact tolerance depends on adversary placement within the ring.

2) Imperfect Reconstruction: When the set of benign nodes $\mathcal{B}$ is unable to compute all Δ tasks due to the lack of corresponding input blocks, perfect reconstruction of the multiplication result is not possible. In this case, γ remaining tasks cannot be reconstructed. These γ blocks can be substituted with all-zero blocks similar to Scheme I (see Section IV-A2). In this scheme, γ is determined numerically as it depends on the position of Byzantine nodes in the ring. This substitution is the element-wise maximum likelihood estimate, and it imposes a relatively small error/distortion to the result.

V. Reconstruction Distortion

In this section, we present the results of the average reconstruction distortion for both schemes. We consider two sparsity models for the input 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 fraction of positions where $\mathbf{X}$ and $\mathbf{Y}$ differ:

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

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 matrix $\mathbf{A}$ share identical marginal distributions, and likewise for the columns of matrix $\mathbf{B}$ (though the distributions between the columns of A and B may differ), Scheme I and II achieve the following upper bound for expected distortion:

$$\mathbb{E}\left[\mathrm{d_H}(\mathbf{C}, \hat{\mathbf{C}})\right] \leq \frac{1}{\Delta}\sum_{z=0}^{N}\binom{N}{z}\alpha^z(1-\alpha)^{N-z} \times \left[(1-\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right])\gamma + P_m z\right] \quad (7)$$

Proof: Distortion can occur when a corrupted block is either detected but not recomputed and therefore, substituted with all zeros, or left undetected. Since P_m is close to zero for large field size q, the upper bound for this term occurs when all z corrupted blocks are detected. Therefore, for an average of $(1-P_m)z$ detected adversaries (first term of summation in (7)), γ blocks cannot be recomputed and must instead be substituted with all zero blocks. If every column in matrices $\mathbf{A}$ and $\mathbf{B}$ shares 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 $\mathbb{P}\left[c_{ij}=0\right] = \mathbb{E}\left[\mathcal{L}(\mathbf{C})\right]$. Therefore, the upper bound for the average number of non-zero elements in these blocks together is $\gamma S'D'(1-\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right])$, which also represents the average number of positions where γ blocks of $\mathbf{C}$ and $\hat{\mathbf{C}}$ differ.

For undetected blocks (the second term of summation in (7)), an average of $P_m z$ undetected adversaries result in $P_m z$ corrupted blocks. Additionally, the maximum distortion in this case occurs when $\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right] = 0$. Therefore, the upper bound for the average number of non-zero elements in these blocks together will be $P_m z S'D' \cdot 1 = P_m z S'D'$. In total, the upper bound for the expected normalized Hamming distance (distortion) between $\mathbf{C}$ and $\hat{\mathbf{C}}$ can be obtained by summing up the aforementioned upper bounds for distortion and normalizing it over the matrix size as

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

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 $\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right]$ 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 constant 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 independent and uniform over the multiplicative group of the field $\mathbb{F}_q$.

Lemma 3: For random matrices $\mathbf{A}$ and $\mathbf{B}$ with constant column weights w_A and w_B, respectively, the expected sparsity level of $\mathbf{C} = \mathbf{A}^T\mathbf{B}$ is

$$\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right] = \frac{\binom{P-w_A}{w_B}}{\binom{P}{w_B}} + \frac{1}{q-1}\sum_{\ell=2}^{\min(w_A,w_B)} \frac{\binom{w_A}{\ell}\binom{P-w_A}{w_B-\ell}}{\binom{P}{w_B}}. \tag{8}$$

Proof: The element in i-th row and j-th column of $\mathbf{C}$ is the dot product of $\mathrm{col}_i(\mathbf{A})$ 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 $\ell = 0$ (the 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 $2 \leq \ell \leq \min(w_A, w_B)$, 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 $\mathrm{col}_i(\mathbf{A})$, 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 $\mathrm{col}_j(\mathbf{B})$. On the other hand, the total number of choices for the independent positions of non-zeros in $\mathrm{col}_i(\mathbf{A})$ and $\mathrm{col}_j(\mathbf{B})$ is $\binom{P}{w_A}\binom{P}{w_B}$. 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 sparsity levels are as follows:

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

Lemma 4: For random matrices $\mathbf{A}$ and $\mathbf{B}$ with independent entries, the expected sparsity level of $\mathbf{C} = \mathbf{A}^T\mathbf{B}$ is

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

where $\lambda = \lambda_a\lambda_b$.

Proof: The proof of the lemma follows along the same lines as the proof of Lemma 3. The only difference is the distribution of the support intersection size ℓ. Also note that now ℓ can be as large as the whole column, i.e., P. For the very first position in both columns, the probability that both values are non-zero is $\lambda = \lambda_a\lambda_b$. Observe that this holds also for any position. In other words, it follows a binomial distribution with parameters P and λ, i.e., the size of the support intersection set equals ℓ with probability

$$\binom{P}{\ell}\lambda^\ell(1-\lambda)^{P-\ell}.$$

Finally,

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right] &= (1-\lambda)^P + \frac{1}{q-1}\sum_{\ell=2}^{P}\binom{P}{\ell}\lambda^\ell(1-\lambda)^{P-\ell} \\ &= \left(1-\frac{1}{q-1}\right)(1-\lambda)^P + \frac{1-P\lambda(1-\lambda)^{P-1}}{q-1}.\end{aligned}$$

■

VI. Computational Complexity

In this section, we outline the computational complexity of the proposed schemes. We consider $\mathcal{L}(\mathbf{A})$, $\mathcal{L}(\mathbf{B})$ and $\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right]$ as the sparsity levels of the input matrices $\mathbf{A}$ and $\mathbf{B}$ and the expected sparsity level of the resulting matrix $\mathbf{C}$, respectively. We provide a breakdown of the main components as follows:

- **Freivalds' algorithm:** The computational complexity of one verification test per node to verify a block column using Freivalds' algorithm is at most $T_F = \mathcal{O}(PD'(1-\mathcal{L}(\mathbf{B}))+S'P(1-\mathcal{L}(\mathbf{A}))+S'D'(1-\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right]))$. The first term represents the cost of computing $\mathbf{B}_j\mathbf{v}$, the second term accounts for the cost of computing $\mathbf{A}_i\mathbf{B}_j\mathbf{v}$, and finally, the last term reflects the complexity of multiplying $\mathbf{C}_ij\mathbf{v}$. All terms consider the maximum overlap of nonzero entries. Note that $\mathbf{v}$ is dense.
- **Common Tag:** The computational complexity of computing tags for the input blocks are at most $\mathcal{O}\left(\gamma_{\mathbf{A}}S'P(1-\mathcal{L}(\mathbf{A}))\right)$ for $\gamma_{\mathbf{A}}$ block columns of $\mathbf{A}$, and $\mathcal{O}\left(\gamma_{\mathbf{B}}PD'(1-\mathcal{L}(\mathbf{B}))\right)$ for $\gamma_{\mathbf{B}}$ block columns of $\mathbf{B}$, respectively, for each node. Also, the complexity of multiplying two tags of size P is at most $\mathcal{O}(P)$. Finally, the complexity of computing $\mathbf{v}_{\mathbf{A}}^T\mathbf{C}_{ij}\mathbf{v}_{\mathbf{B}}$ is at most $\mathcal{O}\left(S'D'(1-\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right])+S'\right)$. Therefore, the total complexity of Common Tag is at most $T_C = \mathcal{O}(\gamma_{\mathbf{A}}S'P(1-\mathcal{L}(\mathbf{A}))+\gamma_{\mathbf{B}}PD'(1-\mathcal{L}(\mathbf{B}))+P(1-\mathcal{L}_2)+S'D'(1-\mathbb{E}\left[\mathcal{L}(\mathbf{C})\right])+S')$.
- **Matrix multiplication:** The computational complexity required at each worker node to compute a matrix-matrix multiplication $\mathbf{C}_{ij} = \mathbf{A}_i\mathbf{B}_j$ is at most $T_M = \mathcal{O}\left(S'PD'(1-\mathcal{L}_1)\right)$, where $\mathcal{L}_1 = \max\left(\mathcal{L}(\mathbf{A}), \mathcal{L}(\mathbf{B})\right)$.
- **Scheme I:** Based on the computational complexity of both, Freivalds' algorithm (T_F) and matrix multiplication (T_M), the cost for Scheme I is computed as follows:
  1) The $N-z$ benign nodes compute their assigned tasks, contributing a cost of $(N-z)T_M$.
  2) These nodes also verify blocks over $N-1$ rounds, incurring a cost of $(N-z)(N-1)\zeta T_F$, where ζ is the number of verification repetitions per round.
  3) Additionally, for the $z-\gamma$ corrupted blocks (which require re-computation), the cost for re-computing and verifying each block over $N-z-1$ rounds is $T_M + (N-z-1)\zeta T_F$, resulting in a total cost of $(z-\gamma)(T_M + (N-z-1)\zeta T_F)$.

  Thus, the overall computational complexity per benign node is $\mathcal{O}([(N-z)(T_M+(N-1)\zeta T_F)+(z-\gamma)(T_M+(N-$

$z-1)\zeta T_F)]/(N-z)$). Since the overall complexity depends on the number of adversaries z (which follows a binomial distribution), we average the cost over this distribution.

- **Scheme II:** Similarly to Scheme I, we compute the overall cost for Scheme II based on the computational complexities of Common Tag (T_C) and matrix multiplication (T_M). The overall computational complexity per benign node for Scheme II is $\mathcal{O}([(N-z)(T_M+(N-1)\zeta(T_C))+(z-\gamma)(T_M+(N-z-1)\zeta(T_C))]/(N-z))$. where N is the total number of nodes, z denotes the number of adversaries, γ is the number of tasks requiring re-computation, and ζ is the number of verification repetitions per round. This cost is averaged over the binomial distribution of adversaries.

Note that these complexities are 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. In other words, $\text{supp}\left(\text{col}_i(\mathbf{A})\cap\text{col}_j(\mathbf{B})\right) \ll (1-\mathcal{L}_1)P$. It is important to note that our uncoded sparse matrix multiplication approach exhibits a linear computational complexity, compared to conventional coded matrix multiplication frameworks (such as Polynomial [23], PolyDot [22], and MatDot [22] coding), which typically incur polynomial complexity of degree greaterr than 1 [2], [22]. The key reason is that dense coded approaches inherently lose the sparsity present in the original matrices $\mathbf{A}$ and $\mathbf{B}$; the coding process often results in coded submatrices that are significantly denser.

VII. Performance Evaluation

In this section, we evaluate the performance of our proposed schemes based on the analytical results derived in Section V. We compare these schemes with the "no-detection" scenario, where the worker nodes do not verify their neighboring nodes' results. Our evaluation is conducted within a network comprising $N = 100$ nodes. The input matrices $\mathbf{A}$ and $\mathbf{B}$ are equally partitioned into $\Delta_\mathbf{A} = \Delta_\mathbf{B} = 10$, thus resulting in $\Delta = N = \Delta_\mathbf{A}\Delta_\mathbf{B} = 100$. Moreover, we consider the adversarial nodes, denoted as z. The distribution of these adversarial nodes follows a binomial distribution, with parameters N and α, where α is the probability of each node being adversarial. We evaluate two sparsity levels $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B})$ of 0.98 and 0.99 for input matrices $\mathbf{A}$ and $\mathbf{B}$, both 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. We utilize a large-size prime field $\mathbb{F}_q = \mathbb{F}_{10^6-17}$ for these implementations. This field size was selected exemplarily to balance misdetection probability and computational efficiency. In the following figures, we use solid lines for the sparsity level of $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{A}) = 0.98$ and dash-dotted lines for the sparsity level of $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{A}) = 0.99$, while different colors specify the relevant parameters.

Fig. 4 illustrates the expected normalized distortion assuming the probabilistic model for the adversarial attack.

This figure illustrates the results for different numbers of tasks that each worker node can perform, i.e., $d = 1, 2, 3$. As depicted in this figure, the adversarial tolerance increases significantly with the number of tasks that each worker node can perform. For instance, with $d = 1$, the scheme cannot

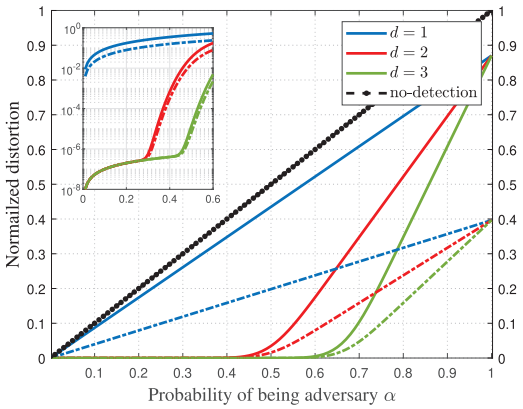


Fig. 4. Expected normalized distortion of Scheme I with $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B})$ of 0.98 (solid lines) and 0.99 (dash-dotted lines).

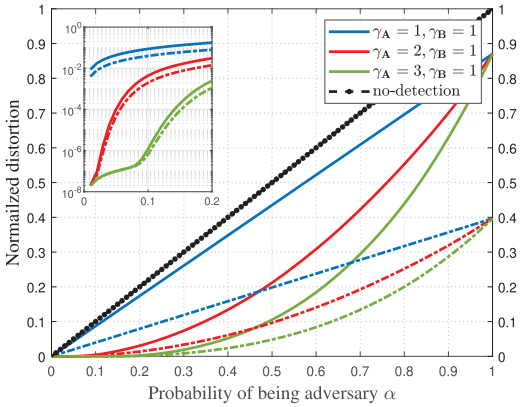
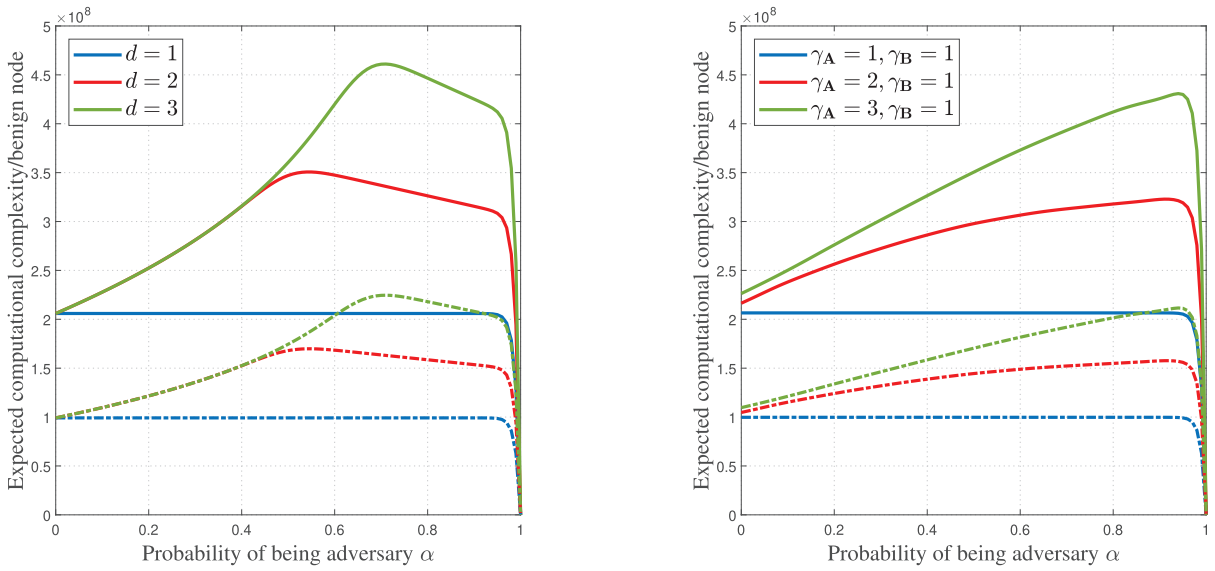


Fig. 5. Expected normalized distortion of Scheme II with $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B})$ of 0.98 (solid lines) and 0.99 (dash-dotted lines).

tolerate any adversaries, whereas, with $d = 2$, it can withstand roughly up to $z = N/2$ adversaries with negligible values of distortion shown in the logarithmic scale of the plot in Fig. 4. The small distortions observed in the ranges of $\alpha \in [0, 0.45]$ and $\alpha \in [0, 0.6]$ for $d = 2$ and $d = 3$, respectively, are attributed to verification errors of Freivalds' algorithm. As explained in Theorem 1, this error results from undetected adversaries and therefore, it depends on the probability of misdetection P_m. The impact of sparsity on distortion is also evident in Fig. 4.

Fig. 5 shows the distortion results for Scheme II, highlighting the worst-case scenario where the adversaries always corrupt their own computed result (the correct versions of Byzantine nodes' tasks do not exist). We evaluate different memory capacity sizes of $\gamma_\mathbf{A}\gamma_\mathbf{B} = 1$, $\gamma_\mathbf{A}\gamma_\mathbf{B} = 2$, and $\gamma_\mathbf{A}\gamma_\mathbf{B} = 3$ in this figure, to be comparable with the result presented in Fig. 4. We numerically determine the average value of corrupted blocks, γ, employing 1000 simulation rounds with adversaries randomly positioned in the ring. As observed in Fig. 5, Scheme II can roughly tolerate 10% and 20% of the nodes to be adversaries for $\gamma_\mathbf{A}\gamma_\mathbf{B} = 2$, $\gamma_\mathbf{A}\gamma_\mathbf{B} = 3$, respectively, with negligible values of distortion shown in the logarithmic scale of the plot. The small distortions observed in the ranges of $\alpha \in [0, 0.1]$ and $\alpha \in [0, 0.2]$ for $\gamma_\mathbf{A}\gamma_\mathbf{B} = 2$ and $\gamma_\mathbf{A}\gamma_\mathbf{B} = 3$, respectively, are attributed to verification (Common Tag) errors. As a result, Scheme II trades off the ability to tolerate a large number of adversaries for a significant reduction in required memory capacity at each node.



(a) (b)

Fig. 6. Expected computation complexity per benign node in (a) Scheme I, and (b) Scheme II, with $\zeta = 1$ for sparsity levels $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B}) = 0.98$ (solid line), and $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B}) = 0.99$ (dash-dotted line).

Similar observations can be made for the i.i.d. entry case in Section V-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 $\mathbb{E}[\mathcal{L}(\mathbf{C})]$. Therefore the i.i.d. entry case shows qualitatively the same behavior as the constant column-weight case in Fig. 4 and Fig. 5. In Fig. 6, we presented the expected computational complexity per benign node versus the probability of being adversary α, analyzed in Section VI. Given the relatively small probability of misdetection P_m in both schemes, attributed to the large field size q, we set the number of repetitions for the verification tests to $\zeta = 1$. As depicted in Fig. 6, the expected computational complexity per benign node for both schemes increases with α. This increase is attributable to higher α values indicating a larger number of adversaries. Consequently, benign nodes must recompute and verify the corrupted blocks, leading to higher computational complexity. However, beyond the region of negligible distortion in Scheme I, as illustrated in Fig. 6(a), the complexity begins to decrease. This occurs because the number of benign nodes decreases, resulting in fewer block recomputations and verifications, or in other words, the number of $z-\gamma$ blocks that can be recomputed by benign nodes starts to decrease in this regime. However, in Scheme II shown in Fig. 6(b), since the benign nodes compute the tags for all their existing blocks in each round, the curves preserve their increasing behavior. It is also evident from Fig. 6 that the computational complexity versus α decreases with increasing sparsity level in both schemes.

VIII. Conclusion

In this work, we considered the problem of decentralized sparse matrix multiplication $\mathbf{C} = \mathbf{A}^T\mathbf{B}$. We also considered the presence of Byzantine nodes to address the security concerns in such a setting. The main goal was to design a fully decentralized setting, detect the adversaries, and mitigate their effects. In this regard, we proposed two new schemes demonstrating the feasibility of sparse matrix multiplication in a fully decentralized setting. In Scheme I, we relied on two key assumptions: a) that all nodes possessed sufficient memory capacity to store the entire input matrices, and b) that they transmitted read-only versions of the results to constrain Byzantine nodes, allowing them only to corrupt their respective tasks or matrix blocks. Scheme II relaxed both assumptions by considering a limited storage capacity for each node and allowing Byzantine nodes to arbitrarily corrupt any blocks. In this scheme, the nodes were able to verify any block using our proposed Common Tag verification method. The Common Tag method was introduced as a novel modification of the well-known Freivalds' algorithm and it allowed the nodes to verify the results without requiring the presence of input block matrices associated with the results. Our proposed schemes were applied to both deterministic and probabilistic Byzantine adversaries for perfect and imperfect reconstruction of the matrix product. The results not only demonstrated the feasibility of the proposed schemes but also showed a substantial performance improvement compared to the no-detection case. The results for both approaches further revealed a trade-off between computational complexity at each node and the reconstruction distortion. Scheme I exhibited a higher resilience to adversaries at the expense of requiring storage for the complete input matrices at each node. Conversely, Scheme II required significantly less storage, rendering it suitable for dense input matrices, albeit with a reduced tolerance for adversaries.

References

[1] X. Luo et al., "An efficient second-order approach to factorize sparse matrices in recommender systems," *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 946–956, Aug. 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 Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[3] T. Jahani-Nezhad and M. A. Maddah-Ali, "CodedSketch: A coding scheme for distributed computation of approximated matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 4185–4196, Jun. 2021.

[4] S. Hong, H. Yang, and J. Lee, "Byzantine attack identification in distributed matrix multiplication via locally testable codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 560–565.

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

[6] S. Hong, H. Yang, Y. Yoon, and J. Lee, "Group-wise verifiable coded computing under Byzantine attacks and stragglers," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 4344–4357, 2024.

[7] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4289–4301, Apr. 2023.

[8] W. Liu, L. Chen, and W. Zhang, "Decentralized federated learning: Balancing communication and computing costs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 131–143, 2022.

[9] B. Tan, B. Liu, V. Zheng, and Q. Yang, "A federated recommender system for online services," in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 579–581.

[10] D. Defiebre, D. Sacharidis, and P. Germanakos, "A decentralized recommendation engine in the social Internet of Things," in *Proc. Adjunct Publication 28th ACM Conf. User Model., Adaptation Personalization*, Jul. 2020, pp. 77–82.

[11] E. T. Beltrán et al., "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2983–3013, 4th Quart., 2023.

[12] P. Pinyoanuntapong, W. H. Huff, M. Lee, C. Chen, and P. Wang, "Toward scalable and robust AIoT via decentralized federated learning," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 30–35, Mar. 2022.

[13] Z. Lian et al., "DEEP-FEL: Decentralized, efficient and privacy-enhanced federated edge learning for healthcare cyber physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3558–3569, Sep. 2022.

[14] Z. Wang, Y. Hu, S. Yan, Z. Wang, R. Hou, and C. Wu, "Efficient ring-topology decentralized federated learning with deep generative models for medical data in eHealthcare systems," *Electronics*, vol. 11, no. 10, p. 1548, May 2022.

[15] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Nneural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 5330–5340.

[16] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: Does topology matter?," in *Proc. Int. Conf. Artif. Intell. Statist.*, Jan. 2020, pp. 2348–2358.

[17] E. Cyffers and A. Bellet, "Privacy amplification by decentralization," in *Proc. Int. Conf. Artif. Intell. Statist.*, Jan. 2020, pp. 5334–5353.

[18] Z. Yang and W. U. Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 611–627, Dec. 2019.

[19] A. R. Elkordy, S. Prakash, and S. Avestimehr, "Basil: A fast and Byzantine-resilient approach for decentralized training," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 9, pp. 2694–2716, Sep. 2022.

[20] S. Ghasvarianjahromi, Y. Yakimenka, and J. Kliewer, "Decentralized sparse matrix multiplication under Byzantine attacks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2023, pp. 1723–1728.

[21] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2418–2422.

[22] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2017, pp. 1264–1270.

[23] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 4406–4416.

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

[25] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the 'Cubic' barrier," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 245–250.

[26] S. Hong, H. Yang, Y. Yoon, and J. Lee, "Straggler-exploiting fully private distributed matrix multiplication with Chebyshev polynomials," *IEEE Trans. Commun.*, vol. 71, no. 3, pp. 1579–1594, Mar. 2023.

[27] J. Li and C. Hollanti, "Private and secure distributed matrix multiplication schemes for replicated or MDS-coded servers," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 659–669, 2022.

[28] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5152–5160.

[29] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 3022–3026.

[30] A. B. Das and A. Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 4156–4181, Jun. 2022.

[31] A. B. Das and A. Ramamoorthy, "An integrated method to deal with partial stragglers and sparse matrices in distributed computations," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 1010–1015.

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

[33] A. B. Das, A. Ramamoorthy, D. J. Love, and C. G. Brinton, "Preserving sparsity and privacy in straggler-resilient distributed matrix computations," in *Proc. 59th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2023, pp. 1–8.

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

[35] S. Jain, M. Cardone, and S. Mohajer, "Identifying reliable machines for distributed matrix-vector multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 820–825.

[36] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-decodable coded computing: Breaking the adversarial toleration barrier," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 3, pp. 867–878, Sep. 2021.

[37] C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the singleton bound," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 275–285, Jun. 2022.

[38] O. Makkonen and C. Hollanti, "General framework for linear secure distributed matrix multiplication with Byzantine servers," *IEEE Trans. Inf. Theory*, vol. 70, no. 6, pp. 3864–3877, Jun. 2024.

[39] R. Freivalds, "Fast probabilistic algorithms," in *Proc. Int. Symp. Math. Found. Comput. Sci.* Berlin, Germany: Springer, 1979, pp. 57–69.

[40] R. A. Machado, G. L. Matthews, and W. Santos, "HerA scheme: Secure distributed matrix multiplication via Hermitian codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2023, pp. 1729–1734.

[41] R. A. Machado and F. Manganiello, "Root of unity for secure distributed matrix multiplication: Grid partition case," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2022, pp. 155–159.

[42] A. Şahin and R. Yang, "A survey on over-the-air computation," *IEEE Commun. Surveys & Tuts.*, vol. 25, no. 3, pp. 1877–1908, 3rd Quart., 2023.

[43] B. Barak and S. Halevi, "A model and architecture for Pseudo-random generation with applications to /dev/random," in *Proc. 12th ACM Conf. Comput. Commun. Secur.*, Nov. 2005, pp. 203–212.

[44] N. Srivastava, H. Jin, J. Liu, D. Albonesi, and Z. Zhang, "MatRaptor: A sparse-sparse matrix multiplication accelerator based on row-wise product," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 766–780.

[45] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[46] R. Gu et al., "Towards efficient large-scale interprocedural program static analysis on distributed data-parallel computation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 4, pp. 867–883, Apr. 2021.

[47] K. Fu, M. F. Kaashoek, and D. Mazières, "Fast and secure distributed read-only file system," *ACM Trans. Comput. Syst.*, vol. 20, no. 1, pp. 1–24, Feb. 2002.