# Sparse Random Khatri-Rao Product Codes for Distributed Matrix Multiplication

Ruowan Ji, Anoosheh Heidarzadeh, and Krishna R. Narayanan

2022 IEEE Information Theory Workshop (ITW) | 978-1-6654-8341-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITW54588.2022.9965842

***Abstract*— We introduce two generalizations to the paradigm of using Random Khatri-Rao Product (RKRP) codes for distributed matrix multiplication. We first introduce a class of codes called Sparse Random Khatri-Rao Product (SRKRP) codes which have sparse generator matrices. SRKRP codes result in lower encoding, computation and communication costs than RKRP codes when the input matrices are sparse, while they exhibit similar numerical stability to other state of the art schemes. We empirically study the relationship between the probability of the generator matrix (restricted to the set of non-stragglers) of a randomly chosen SRKRP code being rank deficient and various parameters of the coding scheme including the degree of sparsity of the generator matrix and the number of non-stragglers. Secondly, we show that if the master node can perform a very small number of matrix product computations in addition to the computations performed by the workers, the failure probability can be substantially improved.**

## I. Introduction

Many machine learning applications require multiplication of two large matrices with real-valued entries. Such large-scale matrix multiplications cannot be simply performed on a single machine, and a natural solution is to parallelize the computation using the master-worker paradigm on distributed computing platforms. In classical distributed matrix multiplication schemes, the master splits each of the two input matrices into smaller blocks (submatrices), and requests each worker to compute and return the product of a pair of blocks—each belonging to one of the two input matrices. Upon receiving the computation results of all workers, the master recovers the product of the two input matrices. However, such systems are prone to stragglers (i.e., those workers that do not return their results as quickly as the rest of the workers) because the master must wait for all workers—including the stragglers—to finish their computations and return their results [1].

A promising approach to mitigate the effect of stragglers is to incorporate redundancy in the computations of the workers—using coding techniques—so that the master can recover the required product from the results of a subset of workers, instead of waiting for the results of all workers [2]. Inspired by the work of Lee *et al.* [1], several coding-based distributed matrix multiplication schemes have been recently proposed [3]–[15]. (Several variations of the coded distributed matrix multiplication problem—not closely related to our work—have also been studied in the literature, see, e.g., [16]–[41].) These schemes provide different trade-offs between several performance metrics including (i) recovery threshold, i.e., the minimum number of non-straggling workers required for successful recovery, (ii) communication cost, i.e., the average amount of information that needs to be transferred from the master to a worker, (iii) computation load, i.e., the average number of arithmetic operations performed by a worker, (iv) computational complexity of encoding and decoding processes, and (v) numerical stability in the presence of round-off and truncation errors.

Most of the existing codes for distributed matrix multiplication provide deterministic guarantees on the recovery threshold, i.e., the master can successfully decode from the results of *any* subset of workers of size no less than a certain threshold. Examples of such coding schemes are Polynomial codes [3] and MatDot codes [5]. While these codes have excellent performance in terms of recovery threshold, they are highly numerically unstable when the operations are performed over the field of real numbers. Motivated by this, in recent years, several numerically-stable coding schemes with deterministic guarantees were proposed in [7]–[10]. A comprehensive comparison of codes with deterministic guarantees, which we collectively refer to as *deterministic codes*, can be found in [14].

Aside from deterministic codes are the coding schemes that provide probabilistic guarantees on the recovery threshold, i.e., the master can successfully decode from the results of *a randomly chosen* subset of workers of size no less than a certain threshold, *with high probability*. Examples of such codes include Sparse codes [13], Factored Luby-Transform (FLT) codes and Factored Raptor (FRT) codes [14], and Random Khatri-Rao Product (RKRP) codes [15]. All of these codes are highly numerically stable. Sparse codes and FLT/FRT codes achieve optimal recovery threshold asymptotically (with probability approaching 1) as the number of workers grows unbounded, whereas RKRP codes have optimal recovery threshold with probability 1. RKRP codes have a dense generator matrix, whereas Sparse codes and FLT/FRT codes have sparse generator matrices. As a result, the encoding/decoding complexity and the computation load of these codes can be substantially lower than those of RKRP codes, particularly when the input matrices are sparse [14]. The main difference between Sparse codes and FLT/FRT codes is that the communication cost of Sparse codes is substantially higher than that of FLT/FRT codes (or even RKRP codes), whereas the communication cost of FLT/FRT codes can be much lower than that of RKRP codes, particularly when the input matrices are sparse [14].

The authors are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (E-mail: {jiruowan, anoosheh, krn}@tamu.edu).

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. CIF-2008714.

In this work, we introduce a new coding scheme, referred to as *Sparse Random Khatri-Rao Product (SRKRP) codes*, which is a generalization of RKRP codes. An SRKRP code can have a very sparse generator matrix—similar to FLT/FRT codes. As a result, when the input matrices are sparse, the encoding complexity, the communication cost, and the computation cost of SRKRP codes can be much lower than those of the original RKRP codes. The decoding complexity of SRKRP codes is, however, comparable to that of the original RKRP codes, and higher than that of FLT/FRT codes. The numerical stability of SRKRP codes is also comparable to that of RKRP codes and FLT/FRT codes.

When compared to FLT/FRT codes, SRKRP codes—with generator matrices of the same size and the same degree of sparsity—have a substantially lower failure probability, even when the number of workers is in the order of tens or hundreds. While a theoretical analysis of the failure probability of SRKRP codes remains unknown in general, our simulations show that these codes can have a very low failure probability, even when the generator matrix of the code is much sparser than that of the original RKRP codes. In addition, our simulations show that a few extra computations (as little as one) performed locally at the master—in parallel to those computations performed by the workers—can substantially reduce the failure probability of SRKRP codes. To the best of our knowledge, this work is the first in the literature on coded distributed matrix multiplication to study the role of such extra computations.

## II. Problem Setup

We use bold-face capital (lowercase) letters for matrices (vectors). We denote the entry $(a,b)$ of matrix $\mathbf{M}$ by $(\mathbf{M})_{a,b}$. For any integers $1 < i < j$, we denote $\{i, i+1, \ldots, j\}$ by $[i:j]$, and for any integer $i \geq 1$, denote $\{1, \ldots, i\}$ by $[i]$.

Consider a distributed master-worker framework in which the master node has two input matrices $\mathbf{A} \in \mathbb{R}^{r\times s}$ and $\mathbf{B} \in \mathbb{R}^{r\times t}$, and wishes to compute the matrix $\mathbf{C} := \mathbf{A}^\mathsf{T}\mathbf{B}$ using the help of $N$ worker nodes. To do so, suppose that the master node splits the input matrix $\mathbf{A}$ column-wise into $m$ submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathbb{R}^{r\times \frac{s}{m}}$, and splits the input matrix $\mathbf{B}$ column-wise into $n$ submatrices $\mathbf{B}_1, \ldots, \mathbf{B}_n \in \mathbb{R}^{r\times \frac{t}{n}}$, where $m, n$ are two arbitrary integers such that $mn \leq N$. Note that the matrix $\mathbf{C} = \mathbf{A}^\mathsf{T}\mathbf{B} = [\mathbf{A}_i^\mathsf{T}\mathbf{B}_j]_{i\in[m],j\in[n]}$. Thus, in order to compute $\mathbf{C}$, the master node uses the help of the worker nodes to compute the $K := mn$ smaller matrix multiplications $\{\mathbf{A}_i^\mathsf{T}\mathbf{B}_j\}_{i\in[m],j\in[n]}$.

Suppose that the computations performed by a randomly chosen subset of $S$ worker nodes—whose identities are initially unknown at the master node—are subject to erasure. Such worker nodes are referred to as *stragglers* in the literature on distributed computing. Due to the existence of stragglers, the master node cannot simply request the worker nodes to compute the smaller matrix multiplications $\mathbf{A}_i^\mathsf{T}\mathbf{B}_j$. Instead, the master node first encodes the $m$ submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_m$ and the $n$ submatrices $\mathbf{B}_1, \ldots, \mathbf{B}_n$ into $N$ coded submatrices $\widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_N \in \mathbb{R}^{r\times \frac{s}{m}}$ and $N$ coded submatrices $\widetilde{\mathbf{B}}_1, \ldots, \widetilde{\mathbf{B}}_N \in \mathbb{R}^{r\times \frac{t}{n}}$, respectively. Then, for each $l \in [N]$, the master node sends $\widetilde{\mathbf{A}}_l$ and $\widetilde{\mathbf{B}}_l$ to the worker node $l$, and requests the worker node $l$ to compute $\widetilde{\mathbf{A}}_l^\mathsf{T}\widetilde{\mathbf{B}}_l$ and send the result back to the master node. For each $l \in [N]$, let

$$\widetilde{\mathbf{A}}_l := \sum_{i=1}^{m} p_{l,i}\mathbf{A}_i, \quad \text{and} \quad \widetilde{\mathbf{B}}_l := \sum_{j=1}^{n} q_{l,j}\mathbf{B}_j, \tag{1}$$

where $\mathbf{p}_l := [p_{l,1}, \ldots, p_{l,m}]$ and $\mathbf{q}_l := [q_{l,1}, \ldots, q_{l,n}]$ are two row-vectors with real entries representing the coding coefficients pertaining to $\widetilde{\mathbf{A}}_l$ and $\widetilde{\mathbf{B}}_l$, respectively.

In addition to the help from the worker nodes, in this work we assume that the master node can also perform some computations locally. To be more specific, we consider the case in which the master node can perform $R$ extra computations $\widetilde{\mathbf{A}}_{N+1}^\mathsf{T}\widetilde{\mathbf{B}}_{N+1}, \ldots, \widetilde{\mathbf{A}}_{N+R}^\mathsf{T}\widetilde{\mathbf{B}}_{N+R}$ in parallel. For each $l \in [N+1 : N+R]$, the coded submatrices $\widetilde{\mathbf{A}}_l$ and $\widetilde{\mathbf{B}}_l$ are constructed similarly as in (1), and the coding vectors $\mathbf{p}_l$ and $\mathbf{q}_l$ corresponding to $\widetilde{\mathbf{A}}_l$ and $\widetilde{\mathbf{B}}_l$ are defined as before. Note that the extra computations performed by the master node are not subject to erasures. That said, these computations are designed in advance—without the knowledge of the configuration of stragglers, and are performed in parallel to those computations performed by the worker nodes.

The goal is to design an encoding scheme, i.e., a (potentially randomized) algorithm for generating the coding vectors $\mathbf{p}_l$'s and $\mathbf{q}_l$'s such that the master node can successfully recover $\{\mathbf{A}_i^\mathsf{T}\mathbf{B}_j\}_{i\in[m],j\in[n]}$ by decoding the results of the $N-S$ computations performed by the non-straggling worker nodes and the results of the $R$ extra computations performed by the master node.

For each $l \in [N+R]$, let $\widetilde{\mathbf{C}}_l := \widetilde{\mathbf{A}}_l^\mathsf{T}\widetilde{\mathbf{B}}_l$. The results received by the master node and those computed locally at the master node can be written in matrix form as follows:

$$\begin{bmatrix} \widetilde{\mathbf{C}}_{l_1} \\ \vdots \\ \widetilde{\mathbf{C}}_{l_{N-S}} \\ \hdashline \widetilde{\mathbf{C}}_{N+1} \\ \vdots \\ \widetilde{\mathbf{C}}_{N+R} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{l_1} \otimes \mathbf{q}_{l_1} \\ \vdots \\ \mathbf{p}_{l_{N-S}} \otimes \mathbf{q}_{l_{N-S}} \\ \hdashline \mathbf{p}_{N+1} \otimes \mathbf{q}_{N+1} \\ \vdots \\ \mathbf{p}_{N+R} \otimes \mathbf{q}_{N+R} \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^\mathsf{T}\mathbf{B}_1 \\ \vdots \\ \mathbf{A}_1^\mathsf{T}\mathbf{B}_n \\ \vdots \\ \mathbf{A}_m^\mathsf{T}\mathbf{B}_1 \\ \vdots \\ \mathbf{A}_m^\mathsf{T}\mathbf{B}_n \end{bmatrix}, \tag{2}$$

where $l_1, l_2, \ldots, l_{N-S} \in [N]$ represent the indices of the $N-S$ non-straggling worker nodes, and $\mathbf{p} \otimes \mathbf{q}$ represents the Kronecker product of the row-vectors $\mathbf{p}$ and $\mathbf{q}$, i.e.,

$$\mathbf{p} \otimes \mathbf{q} = [p_1q_1, p_1q_2, \ldots, p_1q_n, \ldots, p_mq_1, p_mq_2, \ldots, p_mq_n],$$

where $\mathbf{p} = [p_1, \ldots, p_m]$ and $\mathbf{q} = [q_1, \ldots, q_n]$. One can easily observe that the decoding is successful if and only if the coefficient matrix in the system of linear equations (2) is full-rank. When the rank is full, the master node solves the system of linear equations in (2) and obtains an estimate of $\mathbf{C}_{i,j} := \mathbf{A}_i^\mathsf{T}\mathbf{B}_j$, denoted by $\widehat{\mathbf{C}}_{i,j}$, for each $i \in [m]$ and each $j \in [n]$. (Since the operations are performed over $\mathbb{R}$, the computations are prone to numerical errors, and hence, $\mathbf{C}_{i,j}$ and $\widehat{\mathbf{C}}_{i,j}$ may not necessarily be equal.) An estimate of $\mathbf{C} = \mathbf{A}^\mathsf{T}\mathbf{B}$ is then obtained by $\widehat{\mathbf{C}} := [\widehat{\mathbf{C}}_{i,j}]_{i\in[m],j\in[n]}$.

## III. Proposed Coding Scheme

We build upon RKRP codes of [15], and propose a generalization of these codes, referred to as *Sparse RKRP (SRKRP) codes*, which can have a sparse generator matrix.

### A. Encoding

Let $\mathrm{U}(x) := \sum_{k=1}^{m} \mathrm{U}_k x^k$ and $\mathrm{V}(x) := \sum_{k=1}^{n} \mathrm{V}_k x^k$ be the polynomial representation of two weight distributions, i.e., $0 \leq \mathrm{U}_k \leq 1$ for all $k \in [m]$, $0 \leq \mathrm{V}_k \leq 1$ for all $k \in [n]$, and $\sum_{k=1}^{m} \mathrm{U}_k = \sum_{k=1}^{n} \mathrm{V}_k = 1$. Similarly, let $\mathrm{U}^*(x) := \sum_{k=1}^{m} \mathrm{U}_k^* x^k$ and $\mathrm{V}^*(x) := \sum_{k=1}^{n} \mathrm{V}_k^* x^k$ be the polynomial representation of two weight distributions.

Let $X$ be an arbitrary random variable such that the CDF of $X$ is absolutely continuous with respect to the Lebesgue measure, e.g., the uniform random variable $X \sim \mathcal{U}(0,1)$.

In an SRKRP code, the coding vectors $\mathbf{p}_l = [p_{l,1}, \ldots, p_{l,m}]$ and $\mathbf{q}_l = [q_{l,1}, \ldots, q_{l,n}]$ for each $l \in [N]$ are constructed as follows:

1) Randomly choose a weight $u_l$ and a weight $v_l$ by sampling from the weight distribution $\mathrm{U}(x)$ and the weight distribution $\mathrm{V}(x)$, respectively, where the probability of $u_l = k$ is $\mathrm{U}_k$ for each $k \in [m]$, and the probability of $v_l = k$ is $\mathrm{V}_k$ for each $k \in [n]$.
2) Randomly choose a subset of $[m]$ of size $u_l$, say, $\mathcal{S}_l$, and randomly choose a subset of $[n]$ of size $v$, say, $\mathcal{T}_l$.
3) Let $\{p_{l,i} : i \in \mathcal{S}_l\}$ and $\{q_{l,j} : j \in \mathcal{T}_l\}$ be independently generated realizations of random variable $X$. Also, let $p_{l,i} = 0$ for all $i \notin \mathcal{S}_l$, and let $q_{l,j} = 0$ for all $j \notin \mathcal{T}_l$.

The coding vectors $\mathbf{p}_l$ and $\mathbf{q}_l$ for each $l \in [N+1 : N+R]$ are also constructed similarly as above except that in this case the weight distributions $\mathrm{U}(x)$ and $\mathrm{V}(x)$ are replaced by the weight distributions $\mathrm{U}^*(x)$ and $\mathrm{V}^*(x)$, respectively.

Let $l_1, \ldots, l_{N-S} \in [N]$ be the indices of the $N-S$ non-straggling worker nodes. Let $\mathbf{P}$ and $\mathbf{Q}$ be two matrices defined as $\mathbf{P} = [\mathbf{p}_{l_1}^\mathsf{T}, \ldots, \mathbf{p}_{l_{N-S}}^\mathsf{T}, \mathbf{p}_{N+1}^\mathsf{T}, \ldots, \mathbf{p}_{N+R}^\mathsf{T}]^\mathsf{T}$ and $\mathbf{Q} = [\mathbf{q}_{l_1}^\mathsf{T}, \ldots, \mathbf{q}_{l_{N-S}}^\mathsf{T}, \mathbf{q}_{N+1}^\mathsf{T}, \ldots, \mathbf{q}_{N+R}^\mathsf{T}]^\mathsf{T}$. Note that the size of $\mathbf{P}$ is $(N-S+R) \times m$, and the size of $\mathbf{Q}$ is $(N-S+R) \times n$. Each of the first $N-S$ rows of $\mathbf{P}$ contains $u_{\mathrm{avg}} = \sum_{k=1}^{m} \mathrm{U}_k k$ nonzero entries on average, and each of the last $R$ rows of $\mathbf{P}$ contains $u_{\mathrm{avg}}^* = \sum_{k=1}^{m} \mathrm{U}_k^* k$ nonzero entries on average. Similarly, each of the first $N-S$ rows of $\mathbf{Q}$ contains $v_{\mathrm{avg}} = \sum_{k=1}^{n} \mathrm{V}_k k$ nonzero entries on average, and each of the last $R$ rows of $\mathbf{Q}$ contains $v_{\mathrm{avg}}^* = \sum_{k=1}^{n} \mathrm{V}_k^* k$ nonzero entries on average. Let $\mathbf{G} := \mathbf{P} \odot \mathbf{Q}$ be the row-wise Khatri-Rao product of the matrices $\mathbf{P}$ and $\mathbf{Q}$, i.e.,

$$\mathbf{G} = \begin{bmatrix} \mathbf{p}_{l_1} \otimes \mathbf{q}_{l_1} \\ \vdots \\ \mathbf{p}_{l_{N-S}} \otimes \mathbf{q}_{l_{N-S}} \\ \mathbf{p}_{N+1} \otimes \mathbf{q}_{N+1} \\ \vdots \\ \mathbf{p}_{N+R} \otimes \mathbf{q}_{N+R} \end{bmatrix}. \tag{3}$$

It is easy to verify that each of the first $N-S$ rows of $\mathbf{G}$ contains $w_{\mathrm{avg}} = u_{\mathrm{avg}} v_{\mathrm{avg}}$ nonzero entries on average, and each of the last $R$ rows of $\mathbf{G}$ contains $w_{\mathrm{avg}}^* = u_{\mathrm{avg}}^* v_{\mathrm{avg}}^*$ nonzero entries on average. Recall that in the original RKRP codes [15], there are no weight distributions $\mathrm{U}^*(x)$ and $\mathrm{V}^*(x)$ since $R = 0$, and the weight distributions $\mathrm{U}(x) = x^m$ and $\mathrm{V}(x) = x^n$. Note that in this case, $u_{\mathrm{avg}} = m$ and $v_{\mathrm{avg}} = n$. This implies that the coding vectors in the original RKRP codes are dense. Taking $\mathrm{U}(x)$ and $\mathrm{V}(x)$ to be weight distributions with $u_{\mathrm{avg}} < m$ and $v_{\mathrm{avg}} < n$, SRKRP codes can take advantage of sparser coding vectors when compared to the original RKRP codes.

### B. Decoding

Note that the matrix $\mathbf{G}$ defined as in (3) is the coefficient matrix in the system of linear equations (2). Rewriting (2), for each $a \in [\frac{s}{m}]$ and each $b \in [\frac{t}{n}]$, we have

$$\underbrace{\begin{bmatrix} (\widetilde{\mathbf{C}}_{l_1})_{a,b} \\ \vdots \\ (\widetilde{\mathbf{C}}_{l_{N-S}})_{a,b} \\ (\widetilde{\mathbf{C}}_{N+1})_{a,b} \\ \vdots \\ (\widetilde{\mathbf{C}}_{N+R})_{a,b} \end{bmatrix}}_{\mathbf{y}^{(a,b)}} = \mathbf{G} \underbrace{\begin{bmatrix} (\mathbf{C}_{1,1})_{a,b} \\ \vdots \\ (\mathbf{C}_{1,n})_{a,b} \\ \vdots \\ (\mathbf{C}_{m,1})_{a,b} \\ \vdots \\ (\mathbf{C}_{m,n})_{a,b} \end{bmatrix}}_{\mathbf{z}^{(a,b)}}, \tag{4}$$

where $\widetilde{\mathbf{C}}_l$'s and $\mathbf{C}_{i,j}$'s are as defined in Section II. Note that $\mathbf{y}^{(a,b)}$ and $\mathbf{z}^{(a,b)}$ defined in (4) are two column-vectors with real entries, each of length $K$; and all $K$ coordinates of $\mathbf{y}^{(a,b)}$ are known by the master node, whereas the $K$ coordinates of $\mathbf{z}^{(a,b)}$ are unknown at the master node. Given that the matrix $\mathbf{G}$ is full-rank, the master node solves the system of linear equations (4), and obtains an estimate $\widehat{\mathbf{z}}^{(a,b)}$ of $\mathbf{z}^{(a,b)}$. (In the absence of numerical errors, $\widehat{\mathbf{z}}^{(a,b)} = \mathbf{z}^{(a,b)}$.) Upon computing $\widehat{\mathbf{z}}^{(a,b)}$ for all $a \in [\frac{s}{m}]$ and all $b \in [\frac{t}{n}]$, the master node obtains an estimate $\widehat{\mathbf{C}} = [\widehat{\mathbf{C}}_{i,j}]_{i \in [m], j \in [n]}$ of $\mathbf{C}$, where $(\widehat{\mathbf{C}}_{i,j})_{a,b}$ is the $((i-1)n+j)$th coordinate of $\widehat{\mathbf{z}}^{(a,b)}$.

## IV. Performance Analysis

To measure the performance of SRKRP codes, we consider the following metrics: (i) failure probability, (ii) computation load per worker, and (iii) communication cost per worker. Discussions about the encoding/decoding complexity and the numerical stability of SRKRP codes can be found in the long version of this work, [42].

*1) Failure probability:* Since the encoding scheme of SRKRP codes is randomized and the configuration of stragglers is assumed to be random, the decoding may or may not be successful for a given realization of the coding vectors and a given configuration of the stragglers. As a result, we consider the failure probability—defined as the probability that the decoding fails for a randomly generated code realization and a randomly chosen configuration of stragglers—as a metric to measure the performance.

Thinking of $\mathbf{P}$ and $\mathbf{Q}$ as two random matrices, it can be seen that the structure of the matrix $\mathbf{G} = \mathbf{P} \odot \mathbf{Q}$ is random, and hence, the rank of $\mathbf{G}$ is a random variable. Thus, the failure probability is equal to the probability that a randomly generated matrix $\mathbf{G}$ is not full-rank.

**No extra computation ($R = 0$):** As was shown in [15], for the case of $R = 0$, the matrix $\mathbf{G}$ is full-rank with probability 1 when $\mathrm{U}(x) = x^m$ and $\mathrm{V}(x) = x^n$. Also, when $\mathrm{U}(x) = x$ and $\mathrm{V}(x) = x$, it is easy to show that if $N - S = K$, the matrix $\mathbf{G}$ is full-rank with probability $K!/K^K$, which converges to 0 as $K$ grows unbounded. To the best of our knowledge, the full-rank probability of the matrix $\mathbf{G}$ is not known for any other $\mathrm{U}(x)$ and $\mathrm{V}(x)$, and hence, a theoretical analysis of the failure probability of SRKRP codes remains unknown in general. Notwithstanding, our simulation results in Section V reveal several interesting properties of these codes for the case of $R = 0$:

1) The failure probability depends mainly on the average weights $u_{\text{avg}}$ and $v_{\text{avg}}$, and does not change for different pairs of $\mathrm{U}(x)$ and $\mathrm{V}(x)$ which yield the same average weights $u_{\text{avg}}$ and $v_{\text{avg}}$, respectively.
2) For a fixed overall average weight $w_{\text{avg}}$, the closer are the average weights $u_{\text{avg}}$ and $v_{\text{avg}}$ to $\sqrt{w_{\text{avg}}}$, the smaller is the failure probability.
3) For $w_{\text{avg}} > \log K$, the failure probability is close to the probability that the matrix $\mathbf{G}$ has an all-zero column, and the latter probability is close to the probability that a random matrix $\mathbf{M}$ of the same size as the matrix $\mathbf{G}$ has an all-zero column, where the entries of the matrix $\mathbf{M}$ are realizations of i.i.d. Bernoulli random variables with success probability $w_{\text{avg}}/K$.

Observations (1) and (2) suggest that without loss of generality, we can consider the same weight distribution for both $\mathrm{U}(x)$ and $\mathrm{V}(x)$, i.e., $\mathrm{U}(x) = \mathrm{V}(x) = \Lambda(x)$ for some weight distribution $\Lambda(x)$, and we can restrict our attention to weight distributions $\Lambda(x)$ of simplest form that yield the overall average weight $w_{\text{avg}}$, i.e., $\Lambda(x) = x^{\sqrt{w_{\text{avg}}}}$ or $\Lambda(x) = \lambda x^{\lfloor\sqrt{w_{\text{avg}}}\rfloor} + (1-\lambda)x^{\lceil\sqrt{w_{\text{avg}}}\rceil}$ for $\lambda = (\lceil\sqrt{w_{\text{avg}}}\rceil - \sqrt{w_{\text{avg}}})/(\lceil\sqrt{w_{\text{avg}}}\rceil - \lfloor\sqrt{w_{\text{avg}}}\rfloor)$ when $w_{\text{avg}}$ is a perfect square or not a perfect square, respectively.

Observation (3) suggests that the failure probability of an SRKRP code with $R = 0$ can be closely approximated by $1 - (1 - (1 - w_{\text{avg}}/K)^{N-S})^K$, noting that a column of a random matrix of size $(N - S) \times K$—whose entries are realizations of i.i.d. Bernoulli random variables with success probability $w_{\text{avg}}/K$—is all-zero with probability $(1 - w_{\text{avg}}/K)^{N-S}$.

**Leveraging extra computations ($R \geq 1$):** The above observation implies that when $R = 0$ and $w_{\text{avg}} < K$, an SRKRP code fails with a nonzero probability. When $K$ grows unbounded and $N - S - K$ remains constant, if $w_{\text{avg}} > \log K$, the failure probability vanishes. However, when $K$ is finite and $N - S - K$ is small, the failure probability may not be as small as required, even for arbitrarily large $w_{\text{avg}} < K$. To alleviate this drawback, we propose to leverage $R$ extra computations performed by the master node.

Extending the result of [15] to the cases with $R \geq 1$, it is immediate that the matrix $\mathbf{G}$ is full-rank with probability 1 when $\mathrm{U}(x) = \mathrm{U}^*(x) = x^m$ and $\mathrm{V}(x) = \mathrm{V}^*(x) = x^n$. However, the probability of the matrix $\mathbf{G}$ being full-rank remains unknown for any $\mathrm{U}(x)$ and $\mathrm{V}(x)$ with $u_{\text{avg}} < m$ and $v_{\text{avg}} < n$, even when $\mathrm{U}^*(x) = x^m$ and $\mathrm{V}^*(x) = x^n$. Intuitively, for any $u_{\text{avg}}$ and $v_{\text{avg}}$ and any $R \geq 1$, we expect that the larger are the average weights $u^*_{\text{avg}}$ and $v^*_{\text{avg}}$, the larger is the probability that the matrix $\mathbf{G}$ is full-rank. Our simulation results in Section V are consistent with this intuition. Moreover, the results of our simulations show that a few extra computations (i.e., $R \in \{1, 2\}$) with sufficiently large $w^*_{\text{avg}}$ can significantly reduce the failure probability, even when $w_{\text{avg}}$ is as small as $\log K$.

*2) Computation load per worker:* Another performance metric that we consider is the average computational complexity of the matrix multiplication performed by a worker node. Let $nnz(\mathbf{A})$ and $nnz(\mathbf{B})$ denote the number of nonzero entries in $\mathbf{A}$ and $\mathbf{B}$, respectively. For the ease of exposition, assume that the positions of the nonzero entries of $\mathbf{A}$ and $\mathbf{B}$ are randomly chosen. Note that $nnz(\mathbf{A}_i)$ is $\mathcal{O}(\frac{nnz(\mathbf{A})}{m})$ for all $i \in [m]$, and $nnz(\mathbf{B}_j)$ is $\mathcal{O}(\frac{nnz(\mathbf{B})}{n})$ for all $j \in [n]$. For a given $l \in [N]$, let $u_l$ and $v_l$ be the number of nonzero coordinates in the coding vectors $\mathbf{p}_l$ and $\mathbf{q}_l$, respectively. Then, $nnz(\widetilde{\mathbf{A}}_l)$ and $nnz(\widetilde{\mathbf{B}}_l)$ are $\mathcal{O}(u_l\frac{nnz(\mathbf{A})}{m})$ and $\mathcal{O}(v_l\frac{nnz(\mathbf{B})}{n})$, respectively. This further implies that the complexity of computing $\widetilde{\mathbf{A}}_l^{\mathsf{T}}\widetilde{\mathbf{B}}_l$ is $\mathcal{O}(\min\{u_l t\frac{nnz(\mathbf{A})}{mn}, v_l s\frac{nnz(\mathbf{B})}{mn}\})$. Thus, the computation load per worker is given by

$$\mathcal{O}\left(\min\left\{u_{\text{avg}}t\frac{nnz(\mathbf{A})}{mn}, v_{\text{avg}}s\frac{nnz(\mathbf{B})}{mn}\right\}\right).$$

*3) Communication cost per worker:* The communication cost per worker is defined as the average amount of data that needs to be transferred from the master node to a worker node. For a given $l \in [N]$, the master node sends $\widetilde{\mathbf{A}}_l$ and $\widetilde{\mathbf{B}}_l$ to the worker node $l$. Thus, the communication cost per worker node $l$ is $nnz(\widetilde{\mathbf{A}}_l) + nnz(\widetilde{\mathbf{B}}_l)$. Since $nnz(\widetilde{\mathbf{A}}_l)$ and $nnz(\widetilde{\mathbf{B}}_l)$ are $\mathcal{O}(u_l\frac{nnz(\mathbf{A})}{m})$ and $\mathcal{O}(v_l\frac{nnz(\mathbf{B})}{n})$, respectively, the communication cost per worker node $l$ is $\mathcal{O}(u_l\frac{nnz(\mathbf{A})}{m} + v_l\frac{nnz(\mathbf{B})}{n})$. Thus, the communication cost per worker is given by

$$\mathcal{O}\left(u_{\text{avg}}\frac{nnz(\mathbf{A})}{m} + v_{\text{avg}}\frac{nnz(\mathbf{B})}{n}\right).$$

## V. Simulation Results

In this section, we present the results of our simulations for the failure probability of SRKRP codes. For each set of parameters being considered, we have performed Monte-Carlo simulations until 100 failures were observed (i.e., 100 rank-deficient matrices $\mathbf{G}$ were generated).

Fig. 1 presents the failure probability of SRKRP codes with parameters $m = n = 8$ ($K = 64$) and $N - S = K$, for different pairs of weight distributions $\mathrm{U}(x) = \alpha_1 x^2 + \beta_1 x^3 + (1 - \alpha_1 - \beta_1)x^4$ and $\mathrm{V}(x) = \alpha_2 x^2 + \beta_2 x^3 + (1 - \alpha_2 - \beta_2)x^4$, for all $\alpha_i, \beta_i \in \{0, 0.05, 0.1, \ldots, 1\}$ such that $w_{\text{avg}} = u_{\text{avg}}v_{\text{avg}} = (4 - 2\alpha_1 - \beta_1)(4 - 2\alpha_2 - \beta_2) = 9$. For fixed $(u_{\text{avg}}, v_{\text{avg}})$, each point corresponds to the failure probability for a different pair $(\mathrm{U}(x), \mathrm{V}(x))$ with the average weights $u_{\text{avg}}$ and $v_{\text{avg}}$, respectively. As can be seen, the failure probability is (almost) the same for different pairs $(\mathrm{U}(x), \mathrm{V}(x))$ with the same $(u_{\text{avg}}, v_{\text{avg}})$. In addition, it can be seen that the minimum failure probability corresponds to those distribution pairs with $u_{\text{avg}} = v_{\text{avg}} = 3$ $(= \sqrt{w_{\text{avg}}})$.

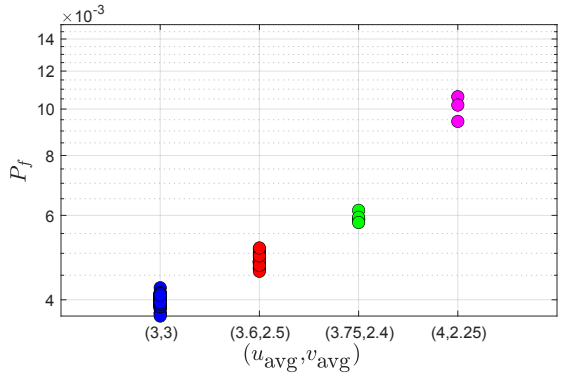


Fig. 1. Failure probability ($P_f$) for different weight distribution pairs $(\mathrm{U}(x),\mathrm{V}(x))$ yielding the same overall average weight $w_{\text{avg}} = u_{\text{avg}}v_{\text{avg}}$.

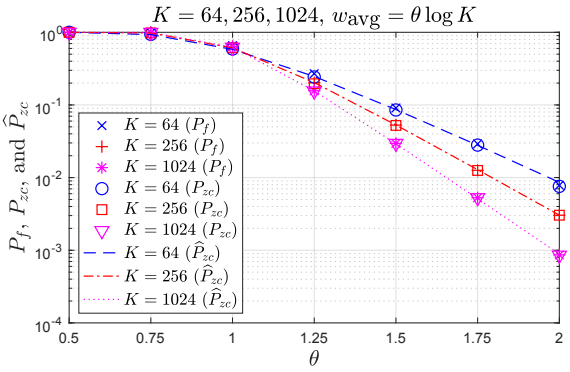


Fig. 2. Failure probability ($P_f$), probability of an all-zero column ($P_{zc}$) and its approximation ($\widehat{P}_{zc}$), for different values of $K$ and $w_{\text{avg}}$.

Fig. 2 depicts the failure probability of SRKRP codes, the probability of existence of an all-zero column in the generator matrix of such codes, and the probability of existence of an all-zero column in a random binary matrix of the same size and the same sparsity as the generator matrix, for parameters $m = n = 8, 16, 32$ ($K = 64, 256, 1024$) and $N - S = K$, and weight distributions $\mathrm{U}(x)$ and $\mathrm{V}(x)$ equal to $\Lambda(x)$ (as defined in Section IV) with $w_{\text{avg}} = \theta \log K$ for $\theta \in \{0.5, 0.75, 1, \ldots, 2\}$. As can be seen, for fixed $K$, as $\theta$ increases, the failure probability decreases, and for $\theta > 1$, the decay is (almost) exponential in $\theta$. In addition, for larger $K$, the failure probability decays faster as $\theta$ increases. It can also be seen that the failure probability is very close to (i) the probability of existence of an all-zero column in the generator matrix, and (ii) the probability of existence of an all-zero column in a random binary matrix with the same size and the same sparsity as the generator matrix.

Fig. 3 depicts the failure probability and its approximation for parameters $m = n = 8$ ($K = 64$) and $N - S \in \{K, K+1, \ldots, K+16\}$, and weight distribution $\Lambda(x)$ with $w_{\text{avg}} = \theta \log K$ for $\theta \in \{0.5, 1, 1.5, 2\}$. As can be seen, the failure probability and its approximation are close to each other, not only for $N - S = K$, but also for $N - S > K$. In addition, one can see that for fixed $K$, the larger is $w_{\text{avg}}$, the more accurate is the approximation.

Fig. 4 depicts the failure probability of SRKRP codes with $R = 0, 1, 2$ dense extra computations performed by the

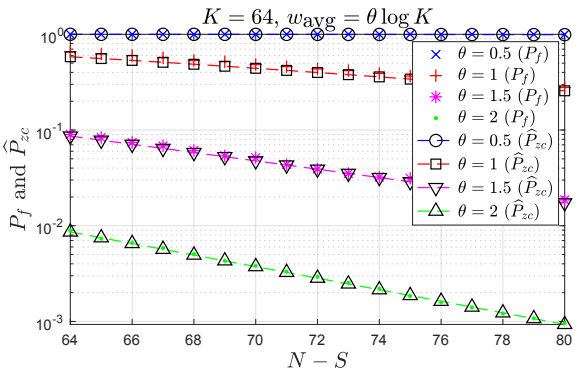


Fig. 3. Failure probability ($P_f$) and approximate probability of an all-zero column ($\widehat{P}_{zc}$) for different values of $N - S$ and $w_{\text{avg}}$.

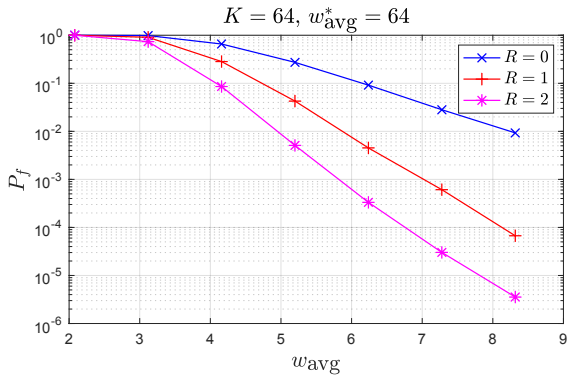


Fig. 4. Failure probability ($P_f$) for different values of $R$ and $w_{\text{avg}}$.

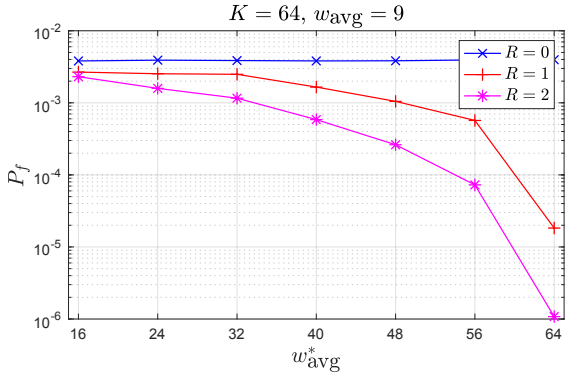


Fig. 5. Failure probability ($P_f$) for different values of $R$ and $w^*_{\text{avg}}$.

master node (i.e., $w^*_{\text{avg}} = mn$), for parameters $m = n = 8$ ($K = 64$) and $N - S = K$, and weight distribution $\Lambda(x)$ with $w_{\text{avg}} = \theta \log K$ for $\theta \in \{0.5, 0.75, 1, \ldots, 2\}$. As can be seen, the failure probability decays exponentially with $w_{\text{avg}}$, and the decay rate increases linearly with $R$.

For parameters $m = n = 8$ ($K = 64$) and $N - S = K$, and weight distribution $\Lambda(x)$ with $w_{\text{avg}} = 9$, Fig. 5 depicts the failure probability of SRKRP codes with $R = 0, 1, 2$ extra computations with distributions $\mathrm{U}^*(x) = \mathrm{V}^*(x)$ equal to $\Lambda^*(x)$ (defined similarly as $\Lambda(x)$) for different $w^*_{\text{avg}} \in \{16, 24, 32, \ldots, 64\}$. As can be seen, for fixed $R$, the failure probability decreases as $w^*_{\text{avg}}$ increases, and for larger $R$, the failure probability decreases faster as $w^*_{\text{avg}}$ increases.

## REFERENCES


[1] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[2] A. Ramamoorthy, A. B. Das, and L. Tang, "Straggler-Resistant Distributed Matrix Computation via Coding Theory: Removing a Bottleneck in Large-Scale Data Processing," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 136–145, 2020.

[3] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial Codes: An Optimal Design for High-Dimensional Coded Matrix Multiplication," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[4] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A Unified Coded Deep Neural Network Training Strategy Based on Generalized PolyDot Codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1585–1589.

[5] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the Optimal Recovery Threshold of Coded Matrix Multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[6] Q. Yu and A. S. Avestimehr, "Entangled Polynomial Codes for Secure, Private, and Batch Distributed Matrix Multiplication: Breaking the "Cubic" Barrier," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 245–250.

[7] M. Fahim and V. R. Cadambe, "Numerically Stable Polynomially Coded Computing," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2758–2785, 2021.

[8] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and Robust Distributed Matrix Computations via Convolutional Coding," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 6266–6282, 2021.

[9] A. Ramamoorthy and L. Tang, "Numerically Stable Coded Matrix Computations via Circulant and Rotation Matrix Embeddings," *IEEE Transactions on Information Theory*, vol. 68, no. 4, pp. 2684–2703, 2022.

[10] A. B. Das and A. Ramamoorthy, "Coded Sparse Matrix Computation Schemes That Leverage Partial Stragglers," *IEEE Transactions on Information Theory*, pp. 1–1, 2022.

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

[12] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-Proofing Massive-Scale Distributed Matrix Multiplication with D-Dimensional Product Codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1993–1997.

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

[14] A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, "Factored LT and Factored Raptor Codes for Large-Scale Distributed Matrix Multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 893–906, 2021.

[15] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao product codes for numerically-stable distributed matrix multiplication," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 253–259.

[16] S. Kianidehkordi, N. Ferdinand, and S. C. Draper, "Hierarchical Coded Matrix Multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 2, pp. 726–754, 2021.

[17] R. Ji, A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, "Squeezed Random Khatri-Rao Product Codes," in *2021 IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–6.

[18] S. Hong, H. Yang, and J. Lee, "Squeezed Polynomial Codes: Communication-Efficient Coded Computation in Straggler-Exploiting Distributed Matrix Multiplication," *IEEE Access*, vol. 8, pp. 190 516–190 528, 2020.

[19] A. B. Das, L. Tang, and A. Ramamoorthy, "C$^3$LES: Codes for Coded Computation That Leverage Stragglers," in *2018 IEEE Information Theory Workshop (ITW)*, 2018, pp. 1–5.

[20] W.-T. Chang and R. Tandon, "Random Sampling for Distributed Coded Matrix Multiplication," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8187–8191.

[21] V. Gupta, S. Wang, T. Courtade, and K. Ramchandran, "OverSketch: Approximate Matrix Multiplication for the Cloud," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 298–304.

[22] T. Jahani-Nezhad and M. A. Maddah-Ali, "CodedSketch: A Coding Scheme for Distributed Computation of Approximated Matrix Multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 4185–4196, 2021.

[23] T. Jahani-Nezhad and M. A. Maddah-ali, "Berrut Approximated Coded Computing: Straggler Resistance Beyond Polynomial Computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2022.

[24] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog Lagrange Coded Computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[25] H. Jeong, A. Devulapalli, V. R. Cadambe, and F. P. Calmon, "$\epsilon$-Approximate Coded Matrix Multiplication Is Nearly Twice as Efficient as Exact Multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 845–854, 2021.

[26] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of Stragglers in Coded Computation," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1988–1992.

[27] R. G. D'Oliveira, S. El Rouayheb, and D. Karpuk, "GASP Codes for Secure Distributed Matrix Multiplication," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1107–1111.

[28] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and Secure Distributed Matrix Multiplication With Flexible Communication Load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.

[29] W.-T. Chang and R. Tandon, "On the Upload versus Download Cost for Secure and Private Matrix Multiplication," in *2019 IEEE Information Theory Workshop (ITW)*, 2019, pp. 1–5.

[30] Z. Jia and S. A. Jafar, "On the Capacity of Secure Distributed Batch Matrix Multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7420–7437, 2021.

[31] R. G. L. D'Oliveira, S. El Rouayheb, D. Heinlein, and D. Karpuk, "Degree Tables for Secure Distributed Matrix Multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 907–918, 2021.

[32] B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate Polynomial Codes for Secure Distributed Matrix Multiplication," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 955–967, 2022.

[33] J. Kakar, A. Khristoforov, S. Ebadifar, and A. Sezgin, "Codes Trading Upload for Download Cost in Secure Distributed Matrix Multiplication," *IEEE Transactions on Communications*, vol. 69, no. 8, pp. 5409–5424, 2021.

[34] M. Kim, H. Yang, and J. Lee, "Private Coded Matrix Multiplication," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1434–1443, 2020.

[35] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "GCSA Codes With Noise Alignment for Secure Coded Multi-Party Batch Matrix Multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 306–316, 2021.

[36] H. Yang and J. Lee, "Secure Distributed Computing With Straggling Servers Using Polynomial Codes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2019.

[37] M. Kim and J. Lee, "Private Secure Coded Computation," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1918–1921, 2019.

[38] S. Dutta, V. Cadambe, and P. Grover, ""Short-Dot": Computing Large Linear Transforms Distributedly Using Coded Short Dot Products," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6171–6193, 2019.

[39] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1215–1225.

[40] Z. Jia and S. A. Jafar, "Cross Subspace Alignment Codes for Coded Distributed Batch Computation," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.

[41] ——, "Generalized Cross Subspace Alignment Codes for Coded Distributed Batch Matrix Multiplication," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[42] R. Ji, A. Heidarzadeh, and K. R. Narayanan, "Sparse Random Khatri-Rao Product Codes for Distributed Matrix Multiplication," May 2022. [Online]. Available: arXiv:2205.06162