Structured Polynomial Codes for Distributed Matrix Multiplication

Derya Malak

***Abstract*—We consider the practical source-worker-receiver framework for distributed matrix multiplication over finite fields. In a setting with two distributed sources with separate encoders, with the first source having access to $\mathbf{A}$ and the second to $\mathbf{B}$, with I distributed workers, where each has a bounded storage (memory) capability reflecting a bounded computational capability, and thus can multiply matrices of smaller size, and one receiver, which aims to reconstruct $\mathbf{A}^{\intercal}\mathbf{B}$ from the transmissions of workers. By combining a *structured linear encoding scheme* with the polynomial code framework, we introduce a novel class of *structured polynomial codes* (StPolyDot codes) for distributed matrix multiplication, leveraging the bilinear structure of this problem, and show that our codes can surpass the performance of the existing state of art in terms of both memory and communication costs, while also maintaining certain advantages in terms of security.**

***Index Terms*—Distributed computation, structured linear coding, polynomial codes, distributed matrix multiplication, storage-computation-communication costs.**

I. Introduction

Basic functions like matrix multiplication, currently constitute the bulk of computational load in scientific computing, as they are omnipresent in applications that include convolution [1], large linear transforms, Fourier transforms, quantum computing [2], as well as in applications of machine learning (ML) such as linear regression and least squares modeling [1], and for applications bottlenecked by the memory bandwidth such as Deep Neural Networks (DNNs) and Large Language Models (LLMs) [3] to mention just a few. The intensity of such computational loads often brings to the fore the necessity for distributed computing, and indeed, one of the most common use-cases is distributed matrix multiplication [4]. The advent of massive parallelization has led to distributed computing systems such as MapReduce to address these tasks.

Distributed computing requires an intense exchange of information among the participating nodes. In scenarios like matrix multiplication, it is evident that to successfully parallelize across multiple workers, we must maintain a reduced communication load, which is a main bottleneck of parallel processing. The need for minimizing this load is evident, which has motivated and enabled several parallel computing techniques, such as in [5]–[10] to mention a few.

The author is with the Commun. Systems Dept., EURECOM, Biot Sophia Antipolis, 06904 France (derya.malak@eurecom.fr).

This research was partially supported by European Research Council ERC-StG Project SENSIBILITÉ under Grant 101077361, by the Huawei France-Funded Chair Toward Future Wireless Networks, and by the program "PEPR Networks of the Future" of France 2030. Manuscript last revised: April 28, 2026.

In this work, we consider distributed computation for a prevalent class of non-linear functions, namely of product of matrices which are bilinear maps[1]. As suggested, this setting finds itself at the core of various technological fields in edge and cloud computing [12] and machine learning [13], and again, as suggested, this is a setting that entails considerable communication overheads as well as an intriguingly intertwined relationship between communication and parallelization. This bottleneck has been studied in seminal works such as [14]–[17], focusing mainly though on the linear function case. Our work focuses on the classical problem of distributed computing of matrix or dot products of two correlated sources, which are canonical examples of bilinear maps, while also capturing an important element of modern large data; the strong structural correlations of this data that serves as computing input. Thus, in this context, we consider, as in [14], [15], [18], two distributed sources and a receiver that wishes to compute such bilinear functions of these sources, and we aim to establish bounds on the optimal sum-rate (the minimum amount of information) that allows a receiver to compute these functions.

The technical contributions of this work are summarized in the following.

A. Main Contributions of this Work

- **New structured polynomial codes.** We consider the practical source-worker-receiver framework, where each distributed worker has a bounded memory capability. Prompted by the bounded computational capabilities of distributed workers, we devise *structured polynomial codes* (StPolyDot codes) for distributed matrix multiplication, addressing *symmetric matrix products* and *non-symmetric matrix products*. Unlike prior works (e.g., [5], [6], [19]–[34]) that restrict preprocessing to linear operations on $\mathbf{A}$ and $\mathbf{B}$, we consider carefully designed non-linear operations at source nodes. These yield a clear benefit because structured coding achieves significant savings in communication rate (compared to [5], [6]), without substantial additional computational load, which is indeed accounted for, where the non-linear component ideally has a small dimension relative to the linear one.
- **Distributed source implementations.** Our design here applies structured linear encoding to address a fundamental problem in the context of coded distributed

[1]A bilinear map is a function $B : U \times V \to W$ combining elements of two vector spaces to yield an element of a third vector space over the same field F, and is linear in each of its arguments [11].

matrix multiplication, a previously unexplored direction. StPolyDot codes naturally support distributed source implementations (e.g., [35]–[39]), enabling separate non-linear operations on $\mathbf{A}$ and $\mathbf{B}$ directly at two distributed source nodes, where each node has access to one matrix.

  - **Storage cost.** The StPolyDot coding framework can *reduce the required storage size per worker —averaged over transmission block length—* to half (Proposition 2) versus the state-of-the-art models (e.g., [5] and [6]) for distributed matrix multiplication.
  - **Computation cost.** We indicate that StPolyDot codes incur minimal computation overhead in the large memory parameter regime (Proposition 4). We similarly evaluate the computation complexity of StPolyDot codes for *non-symmetric matrix products*.
  - **Communication cost.** The StPolyDot scheme, through carefully designed non-linear operations, enables the assignment of lower-dimensional polynomials to each worker, compared to the existing purely linear polynomial codes (cf. [5] and [6]), meaning reduced communication cost from each source to the workers. In our comparisons, we also consider end-to-end communication costs, which involve the cost from the sources to the workers, as well as from the workers to the receiver. The gains become more pronounced when $\mathbf{A}$ and $\mathbf{B}$ are correlated, as demonstrated by numerical results (cf. Section IV). *The proposed approach can substantially lower end-to-end communication costs — up to a factor of* 2 *— as compared to the existing state of art* (Proposition 5).

    For StPolyDot codes, we analyze storage, communication, and computation tradeoffs, identifying regimes where structured codes outperform unstructured ones.

  The inclusion of distributed source compression fundamentally alters the computation–communication interplay: it reduces communication cost at the expense of additional computation at the sources. StPolyDot codes can be leveraged to enlarge the achievable communication–computation tradeoff region for distributed matrix multiplication, yielding a richer set of operating points and greater flexibility for system designers.
- Our schemes have some additional properties, like balancing the computation load across the nodes, as well as offering some security advantages. StPolyDot codes exhibit a degree of robustness to straggler effects, while imposing no structural constraints on the sources. The proposed StPolyDot coding scheme has positive security ramifications as it enables the desired computations without revealing the source information in its entirety via leveraging structured coding (see [14], [40], [41]), which stems directly from the structured binning mechanism inherent to our codes. We briefly discuss these in Section V.

B. Related Work and Connections of Our Work to the State of the Art

Below, we detail the main approaches to distributed coding for computation: unstructured coding, structured coding, coded computing, and matrix multiplication.

a) Unstructured Coding for Computing: Given two statistically dependent, finite alphabet source variables X_1 and X_2 separately observed by two transmitters, Slepian and Wolf have provided an unstructured coding technique for the asymptotic lossless compression of the distributed source sequences $X_1^n = \{X_{1i}\}_{i=1}^n$ and $X_2^n = \{X_{2i}\}_{i=1}^n$ that are i.i.d., achieving the necessary and sufficient rate for a receiver to jointly recover (X_1^n, X_2^n), as given by $R_{\text{SW}}^{\Sigma} = H_q(X_1, X_2)$ [18]. Yamamoto has derived the minimum rate at which a source has to compress X^n for distributed computing of $f(X^n, Y^n)$ with side information Y^n at the receiver, with vanishing error [42]. Exploiting Körner's characteristic graph G_X and its entropy [43], Orlitsky and Roche have devised an unstructured coding scheme to achieve this rate [44]. Their scheme is equivalent to performing Slepian-Wolf encoding on the colors of the sufficiently large n-th OR powers of G_X given Y^n [45]. Graph-theoretic techniques [46]–[49], and codes with function-dependent distances to correct computational errors [50] have been devised for various computing scenarios.

We next detail structured coding-based approaches that are more directly geared towards leveraging the structure of the computation task.

b) Structured Coding for Computing: Körner and Marton (cf. [14]) devised a *structured linear encoding* strategy for distributed computing the modulo-two sum of X_1 and X_2, i.e., $X_1 \oplus_2 X_2$, given i.i.d. source sequences (X_1^n, X_2^n), i.e., for all $i \in [n]$, (X_{1i}, X_{2i}) has the same joint distribution as (X_1, X_2). Their technique, based on the method of Elias [51] (see also [52, pp. 206-207]), constructs linear codes that achieve an asymptotically vanishing probability of error at the minimum sum rate of $2H(X_1 \oplus_2 X_2)$ [14], thus establishing the exact rate region for the modulo-two adder network. Subspace-based lossless linear computation schemes using nested codes — that are sum-rate optimal for a class of source distributions — have been devised in [17], generalizing [14]. The rate region has been extended to a class of non-symmetric source distributions [53], and to the reconstruction of the modulo-q sum $X_1 \oplus_q X_2$ [15, Lemma 5].

For compressing possibly non-additive functions, in [15], Han and Kobayashi have identified key function features that differentiate the Slepian-Wolf and Körner-Marton regions, providing conditions under which computing a general bivariate function $f(X_1^n, X_2^n) = \{f(X_{1i}, X_{2i})\}_{i=1}^n$ requires a lower rate than R_{SW}^{Σ}. Ahlswede and Csiszár have shown that for a decoder to determine most binary-valued, or componentwise functions of distributed sequences (X_1^n, X_2^n), separate encoders must transmit at rates as high as those needed to reconstruct (X_1^n, X_2^n) [54]. For functions like joint type, Hamming distance, or its parity, to determine $f(X_1^n, X_2^n)$ in the knowledge of X_2^n, the encoder of X_1 typically requires a rate comparable to reconstructing X_1^n itself. Additionally, given a distortion criterion, an exact characterization of the achievable rate region for $f(X_1^n, X_2^n)$ — excluding componentwise functions — may be as hard as that for reproducing X_1^n and X_2^n. In distributed computation of non-linear functions of (X_1, X_2), finding an injective mapping between the function and $X_1 \oplus_q X_2$ for a sufficiently large prime field $\mathbb{F}_q$ [55]–

[57], followed by *structured binning*, may yield rate savings over [18].

This paper builds on the foundational principles of structured codes and recent advances (cf. [41]) to develop distributed matrix multiplication techniques over finite fields. We demonstrate significant compression savings for matrix product computations compared to [18], accomplished by applying the structured encoding scheme from [14] to non-linear source mappings, utilizing a smaller field size than in [55], [56]. This use of the structured coding idea in the context of distributed matrix multiplication will prove pivotal in capturing source correlations and computation structures jointly, thus well capturing the distributed matrix multiplication problem. To address this well-known open problem, in a setting with two nodes and two correlated sources $\mathbf{A}$ *and* $\mathbf{B}$ *over finite fields, with the first node having access to* $\mathbf{A}$ *and the second to* $\mathbf{B}$*, we established bounds on the optimal sum rate that allows a receiver to compute the matrix product* $\mathbf{A}^\intercal\mathbf{B}$ *[41]. The objective of the current paper is to apply the ideas in [41] to the framework of coded distributed matrix multiplication within a source-worker-receiver framework.*

c) Coded Distributed Computing (CDC): For distributed computing to achieve the desired parallelization of computational loads across multiple workers, there is an undeniable need for extensive information exchange among the various network nodes. Reducing this communication load is essential for scalability [58], [59] in various topologies [60], [61]. At the heart of reducing communication costs are data placement strategies that account for function structure, thus lowering the sum-rate required for computation [62]–[65], and channel coding techniques, including distributed gradient coding [7], [8], and variants of CDC, such as coded MapReduce [66], and Lagrange coded computing [9], [10], that nicely yield gains in reliability, scalability, computation speed [67], and reduce the communication load [66], [68], as well as in the presence of stragglers, and under privacy-security constraints, via exploiting channel coding methods.

As our interest lies in coded distributed matrix multiplication, we proceed to provide some of the state of art related to this broad problem.

d) Distributed Matrix Multiplication: Numerous coding strategies have been developed to enhance distributed coded matrix multiplication to reduce download costs and mitigate stragglers, such as Short-Dot [69], Polynomial (outer product-based codes) [5], MatDot (inner product-based codes) [32], and PolyDot codes [6], [19], [20], all over finite fields. Exploiting the bounded memory capabilities of distributed workers, these approaches split source matrices into submatrices via linear transformations and transform matrix multiplication into inner or outer product computations. Distributing subtasks across worker nodes enables linearly separable processing of matrix rows and columns. For example, MatDot [32] and PolyDot codes [6] reduce communication costs and improve security, while Polynomial codes [5], as well as generalizations using algebraic function fields [23] are commonly used to mitigate stragglers. Recent research focuses on achieving even greater reductions in communication costs (e.g., [6], [28], [32], [70]). However, these approaches are not sum-rate optimal, even without stragglers. Furthermore, in the absence of stragglers, for Polynomial coding approaches, the user can directly recover the source matrices from subtasks, which holds when the workers are honest but curious [29], [34].

The capacity of secure distributed matrix multiplication, which is the maximum possible ratio of the desired information and the total communication received from a set of distributed workers, has been derived in [27]. Secure and private matrix multiplication has been investigated by linking privacy to the notion of private information retrieval (PIR) and perfect security to secret sharing schemes [28]. In the presence of honest but curious workers, in [29], [30], Polynomial codes based on arithmetic progressions have been studied for secure distributed matrix multiplication. To build resilience against stragglers and enhance privacy, other approaches have explored the tradeoff between computation time and the privacy constraint [31], as well as secure constructions for Polynomial and MatDot codes [32] and secure multi-party batch matrix multiplication [33].

To our knowledge, the existing techniques exploit channel coding methods to mitigate stragglers and enable security, but do not incorporate coded distributed compression aspects. Structured source coding techniques, such as [14], [55], [56], can be effectively incorporated into the existing CDC frameworks and coded matrix multiplication schemes to reduce communication costs further. In this paper, we propose a novel distributed matrix multiplication framework building on the structured linear coding scheme in [14] and our scheme from [40] that captures source correlations and computation structure, and leveraging channel coding schemes to ensure reliable computation at reduced communication cost over the existing approaches.

C. Organization

The rest of the paper is organized as follows. Section II describes the source-worker-receiver framework for distributed matrix multiplication, and details the construction of StPolyDot codes both for symmetric and square non-symmetric matrix products. Section III provides an analysis for the storage, end-to-end communication, and computational complexities and recovery thresholds of StPolyDot codes for distributed computation of *symmetric matrix products*, and expands the construction to *square non-symmetric matrix products*. Section IV contrasts StPolyDot codes with the state of the art, by numerically evaluating the tradeoff space between the communication and computational complexities and recovery thresholds. Finally, Section V summarizes the utility of our approach, providing perspectives and future directions.

Notation. We use regular type for random variables and boldface for vectors and matrices over the finite field $\mathbb{F}_q$. Logarithms base 2 and $q > 2$ are denoted by $\log$ and $\log_q$, respectively, and we write $\exp(\cdot) = q^{(\cdot)}$. We use $\oplus_q$ and $\ominus_q$ to denote modulo-q addition and subtraction (addition with the additive inverse in $\mathbb{F}_q$). For a random variable X with PMF P_X, its entropy in binary and q-ary units is $H(X)$ and $H_q(X) = H(X)/\log_2 q$, respectively. Likewise, for (X_1, X_2) with joint PMF P_{X_1,X_2}, the joint and conditional entropies

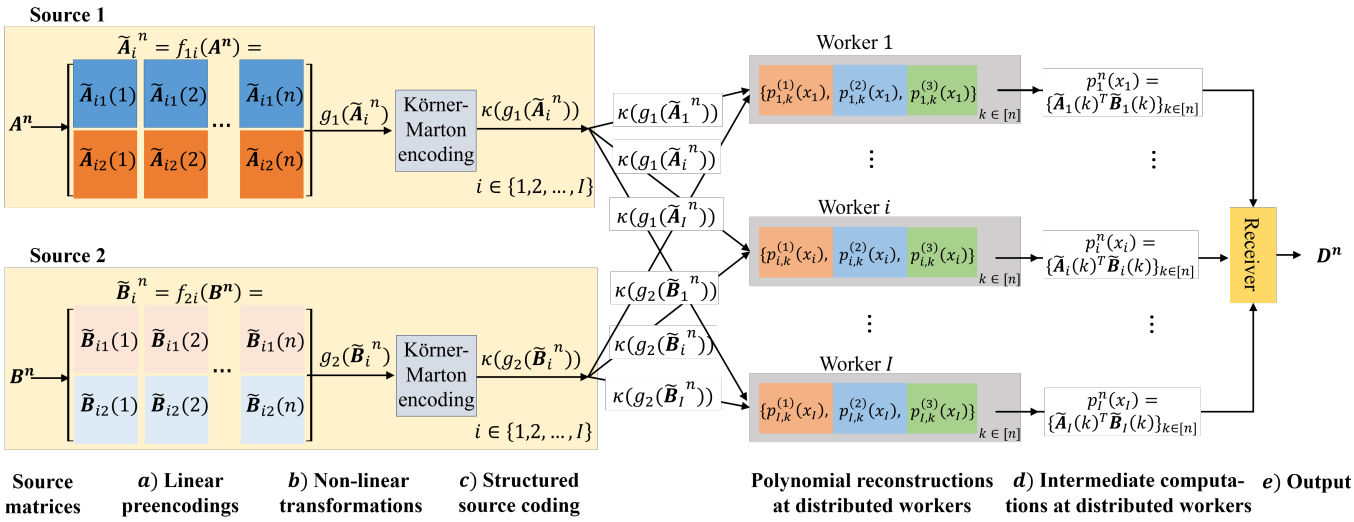

Fig. 1: The StPolyDot coding-based distributed matrix multiplication framework considered in the current work. The different stages, $a)-e)$, are specified on the bottom. The colorings for the input submatrices and the polynomials are chosen in accordance: different shades of orange denote $p_i^{(1)}(x_i) \triangleq \tilde{\mathbf{A}}_{i2} \oplus_q \tilde{\mathbf{B}}_{i1}$ and the submatrices $\tilde{\mathbf{A}}_{i2}$, $\tilde{\mathbf{B}}_{i1}$, shades of blue denote $p_i^{(2)}(x_i) \triangleq \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2}$ and $\tilde{\mathbf{A}}_{i1}$, $\tilde{\mathbf{B}}_{i2}$, and green denotes the non-linear parity polynomial $p_i^{(3)}(x_i) \triangleq \tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i1}^{\intercal}\tilde{\mathbf{B}}_{i2}$, which is linearly separable in terms of $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$.

are $H_q(X_1, X_2)$ and $H_q(X_1 \mid X_2)$, respectively. The acronym i.i.d. stands for independent and identically distributed, and $X_1 \perp\!\!\!\perp X_2$ is used to describe statistical independence between X_1 and X_2. $\mathbb{P}(A)$ is the probability of an event A. For a binomial variable $X \sim \mathrm{Bin}(l, p)$ with $l \in \mathbb{N}$ and $p \in [0, 1]$, the complementary cumulative distribution function is $\bar{F}(m; l, p) = \sum_{j=m}^{l} \binom{l}{j} p^j (1-p)^{l-j}$. When $l = 1$, then X is a Bernoulli variable, denoted $X \sim \mathrm{Bern}(p)$, and $h(p)$ denotes the binary entropy function.

We denote by $[l]$ the set $\{1, \ldots, l\}$, for $l \in \mathbb{Z}^+$, and by $[l_1,\ l_2]$ the set $\{l_1, \ldots, l_2\}$ for $l_1,\ l_2 \in \mathbb{Z}^+$ such that $l_1 \leq l_2$. Given a random matrix $\mathbf{X} = (x_{ij})_{i\in[\ell],\ j\in[m]} \in \mathbb{F}_q^{\ell\times m}$, its i-th row, j-th column, and transpose are given by $\mathbf{X}(i,:)$, $\mathbf{X}(:,j)$, $\mathbf{X}^{\intercal}$, respectively. Alternatively, $\mathbf{x} = (x_i)_{i\in[\ell]} \in \mathbb{F}_q^{\ell\times 1}$ (or $\mathbb{F}_q^{\ell}$) and $\mathbf{x} = \left((x_j)_{j\in[m]}\right)^{\intercal} \in \mathbb{F}_q^{1\times m}$ denote column and row vectors, respectively. For a given $\mathbf{x} \in \mathbb{F}_q^{1\times m}$, for $i,\ j \in \mathbb{Z}^+$, and $i \leq j$, then $\mathbf{x}(i:j) \triangleq [x_i,\ x_{i+1},\ \ldots,\ x_j]$, and similarly for a column vector. The vertical concatenation of $\mathbf{A} \in \mathbb{F}_q^{m_1\times m}$ and $\mathbf{B} \in \mathbb{F}_q^{m_2\times m}$ is denoted by $[\mathbf{A}; \mathbf{B}] \in \mathbb{F}_q^{(m_1+m_2)\times m}$. The notations $\mathbf{1}_{\ell\times m}$ and $\mathbf{0}_{\ell\times m}$ denote $\ell \times m$ matrices of all ones and all zeros, respectively. We write $X^n \triangleq \{X_i\}_{i=1}^n = (X_1, X_2, \ldots, X_n)$, and use both $[X_1; X_2; \ldots; X_n]$ and $(X_1, X_2, \ldots, X_n)$ to denote a sequence of i.i.d. realizations of X; the intended meaning will be clear from context. We extend this notation to matrices by defining $\mathbf{X}^n$, with $\mathbf{X}^n(i,j)$ representing the length-n sequence of realizations of the (i,j)-th component $\mathbf{X}(i,j) \in \mathbb{F}_q$.

II. StPolyDot Codes for Distributed Matrix Multiplication

In this section, we detail the source-worker-receiver-based framework for coded distributed matrix multiplication and the construction of novel structured polynomial codes (StPolyDot codes) by combining the scheme of [41] with the polynomial code framework. The proposed framework consists of two distributed sources with separate encoders, I distributed workers, each with limited storage and computational capabilities, and a receiver node.

The distributed sources separately observe realizations of statistically dependent (i.e., correlated) matrix variables $\mathbf{A} = (a_{ij})_{i\in[\ell],\ j\in[m]} \in \mathbb{F}_q^{\ell\times m}$ and $\mathbf{B} = (b_{ij})_{i\in[\ell],\ j\in[m]} \in \mathbb{F}_q^{\ell\times m}$, respectively[2], where $q \geq 2$. In other words, Source 1 observes $\mathbf{A}$ and Source 2 observes $\mathbf{B}$, respectively. The receiver aims to compute $\mathbf{D} = (d_{ij})_{i,\ j\in[m]} = \mathbf{A}^{\intercal}\mathbf{B} \in \mathbb{F}_q^{m\times m}$.

We take a non-real-time approach that relies on accumulating length-n sequences of potentially correlated source matrix realizations. Specifically, the distributed sources are block-encoded with block length n. We assume statistically dependent finite alphabet two source sequences $\mathbf{A}^n = (\mathbf{A}(1), \mathbf{A}(2), \ldots, \mathbf{A}(n))$ and $\mathbf{B}^n = (\mathbf{B}(1), \mathbf{B}(2), \ldots, \mathbf{B}(n))$ corresponding to length-n i.i.d. realizations of $\mathbf{A}$ and $\mathbf{B}$, respectively. We have the following additional assumptions. The observations are memoryless, namely each realization $\mathbf{A}(k)$ depends only on $\mathbf{B}(k)$, not on $\{\mathbf{A}(k'), \mathbf{B}(k')\}_{k'\neq k}$, which in turn means that $P_{\mathbf{A}^n,\mathbf{B}^n}(\mathbf{A}^n, \mathbf{B}^n) = \prod_{k=1}^n P_{\mathbf{A}(k),\mathbf{B}(k)}(\mathbf{A}(k), \mathbf{B}(k))$. The observations are i.i.d., namely each realization $\mathbf{A}(k)$ is jointly distributed with $\mathbf{B}(k)$, where $\mathbf{A}(k) \sim P_{\mathbf{A}}$ and $\mathbf{B}(k) \sim P_{\mathbf{B}}$ for all $k \in [n]$, and the pairs are i.i.d. across k under some distribution $P_{\mathbf{A},\mathbf{B}}$, which yields $P_{\mathbf{A}^n,\mathbf{B}^n}(\mathbf{A}^n, \mathbf{B}^n) = \prod_{k=1}^n P_{\mathbf{A},\mathbf{B}}(\mathbf{A}(k), \mathbf{B}(k))$.

The above assumptions can be applied to Fourier transforms or linear regression [1], even though these operations might involve deterministic operators and uncorrelated datasets, and there is typically no i.i.d. stream of operands. However, a line of related work considers distributed quantized matrix multiplication, which plays an important role in DNNs and LLMs,

[2]Our approach is not restricted to matrices $\mathbf{A}$ and $\mathbf{B}$ of equal dimensions, as detailed in Section II-.c).

and may indeed consider a stream of operands [3]. Having length-n i.i.d. realizations $\mathbf{A}^n$ and $\mathbf{B}^n$ enables establishing a fundamental lower bound on the communication cost (detailed in Section III-D). Schemes that do not account for streams of matrices may not achieve the same rate[3].

The current work here considers a practical source-worker-receiver-based framework with multiple workers, as shown in Figure 1. Two distributed sources separately observe length-n memoryless and i.i.d. realizations $\mathbf{A}^n$ and $\mathbf{B}^n$, respectively, and then encode their respective realizations independently. The system has $|\mathcal{I}| = I$ workers, where each worker has a bounded memory capability and thus cannot directly compute $\mathbf{D}^n$, which requires the distributed computation of $\mathbf{D}^n$ through multiple workers. To that end, the two sources devise linear preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ for worker i, respectively. They can perform general, and possibly non-linear, component-wise functions $g_1(\tilde{\mathbf{A}}_i^n) = \{g_1(\tilde{\mathbf{A}}_i(k))\}_{k=1}^n = \mathbf{X}_{i,1}^n$ and $g_2(\tilde{\mathbf{B}}_i^n) = \{g_2(\tilde{\mathbf{B}}_i(k))\}_{k=1}^n = \mathbf{X}_{i,2}^n$, where $\mathbf{X}_{i,1}(j) \in \mathbb{F}_q$ and $\mathbf{X}_{i,2}(j) \in \mathbb{F}_q$, respectively. As we will see later, in our setting, the encoders will apply the *structured linear coding* technique of Körner-Marton in [14] to the non-linear transformations $\mathbf{X}_{i,1}^n$ and $\mathbf{X}_{2,i}^n$, respectively. Prior to transmissions, $\mathbf{X}_{i,1}^n$ and $\mathbf{X}_{i,2}^n$ are separately encoded using a common random linear encoding matrix κ, as detailed below (cf. Proposition 1). The encoder outputs, denoted by $\kappa(\mathbf{X}_{i,1}^n)$ and $\kappa(\mathbf{X}_{i,2}^n)$, respectively, are then transmitted over *noiseless channels* to worker $i \in \mathcal{I}$. The goal of the worker is to determine

$$\mathbf{Z}_i^n = \mathbf{X}_{i,1}^n \oplus_q \mathbf{X}_{i,2}^n \ . \tag{1}$$

As we will also see, the worker will modulo-q add the received codewords to obtain $\kappa(\mathbf{X}_{i,1}^n \oplus_q \mathbf{X}_{i,2}^n)$ and then recover $\hat{\mathbf{Z}}^n = \psi(\kappa(\mathbf{X}_{i,1}^n \oplus_q \mathbf{X}_{i,2}^n))$ using a decoding function ψ, where[4] for any fixed $\epsilon \in (0,1)$ and for sufficiently large n, it holds that $\mathbb{P}(\hat{\mathbf{Z}}_i^n \neq \mathbf{Z}_i^n) < \epsilon$. Using the reconstructed sequence $\mathbf{Z}_i^n$, the worker then computes a sequence of matrix products $\{\tilde{\mathbf{A}}_i(k)^\intercal \tilde{\mathbf{B}}_i(k)\}_{k\in[n]}$, and subsequently transmits them to the receiver. Using $\{\tilde{\mathbf{A}}_i(k)^\intercal \tilde{\mathbf{B}}_i(k)\}_{k\in[n]}$ from a subset of $\{i \in \mathcal{I}\}$, the receiver will then proceed to recover the sequence $\mathbf{D}^n = (\mathbf{D}(1), \mathbf{D}(2), \ldots, \mathbf{D}(n))$ of desired matrix products, with $\mathbf{D}(k) = \mathbf{A}(k)^\intercal \mathbf{B}(k) \in \mathbb{F}_q^{m\times m}$, $k \in [n]$, with a small probability of error.

Figure 1 depicts our distributed matrix multiplication framework using StPolyDot codes, which consists of five consecutive stages, namely $a) - e)$, as indicated on the plot. Next, we detail these stages. While doing so, to simplify the notation, we drop the index k of realizations from matrices $\mathbf{A}(k)$ and $\mathbf{B}(k)$, their linear preencodings $\tilde{\mathbf{A}}_i(k)$ and $\tilde{\mathbf{B}}_i(k)$, non-linear transformations $g_1(\tilde{\mathbf{A}}_i(k))$ and $g_2(\tilde{\mathbf{B}}_i(k))$, and the polynomials $\mathbf{p}_{i,k}(x_i)$ devised using the non-linear transformations for each worker $i \in \mathcal{I}$, as detailed below, whenever it is clear from the context.

a) Linear Preencoding of the Source Matrices: To capture the finite storage capabilities of workers, we split $\mathbf{A} \in \mathbb{F}_q^{\ell\times m}$ and $\mathbf{B} \in \mathbb{F}_q^{\ell\times m}$ horizontally into s_r equal-width row blocks and vertically into s_c equal-length column blocks each:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} & \ldots & \mathbf{A}_{0,s_c-1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} & \ldots & \mathbf{A}_{1,s_c-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{s_r-1,0} & \mathbf{A}_{s_r-1,1} & \ldots & \mathbf{A}_{s_r-1,s_c-1} \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} & \ldots & \mathbf{B}_{0,s_c-1} \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} & \ldots & \mathbf{B}_{1,s_c-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{s_r-1,0} & \mathbf{B}_{s_r-1,1} & \ldots & \mathbf{B}_{s_r-1,s_c-1} \end{bmatrix}, \tag{2}$$

where sub-blocks have the same size, i.e., $\mathbf{A}_{j,k}$ and $\mathbf{B}_{j,k}$ for any $j \in [0, s_r-1]$, $k \in [0, s_c-1]$ is a $\frac{\ell}{s_r} \times \frac{m}{s_c}$ matrix, assuming that s_r divides ℓ, and s_c divides m, respectively.

We use (2) to rewrite the desired matrix product $\mathbf{D} \in \mathbb{F}_q^{m\times m}$ as

$$\mathbf{D} = \begin{bmatrix} \sum_{j=0}^{s_r-1} \mathbf{A}_{j,0}^\intercal \mathbf{B}_{j,0} & \sum_{j=0}^{s_r-1} \mathbf{A}_{j,0}^\intercal \mathbf{B}_{j,1} & \ldots & \sum_{j=0}^{s_r-1} \mathbf{A}_{j,0}^\intercal \mathbf{B}_{j,s_c-1} \\ \sum_{j=0}^{s_r-1} \mathbf{A}_{j,1}^\intercal \mathbf{B}_{j,0} & \sum_{j=0}^{s_r-1} \mathbf{A}_{j,1}^\intercal \mathbf{B}_{j,1} & \ldots & \sum_{j=0}^{s_r-1} \mathbf{A}_{j,1}^\intercal \mathbf{B}_{j,s_c-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=0}^{s_r-1} \mathbf{A}_{j,s_c-1}^\intercal \mathbf{B}_{j,0} & \sum_{j=0}^{s_r-1} \mathbf{A}_{j,s_c-1}^\intercal \mathbf{B}_{j,1} & \ldots & \sum_{j=0}^{s_r-1} \mathbf{A}_{j,s_c-1}^\intercal \mathbf{B}_{j,s_c-1} \end{bmatrix}, \tag{3}$$

noting that each sub-block satisfies $\sum_{j=0}^{s_r-1} \mathbf{A}_{j,k}^\intercal \mathbf{B}_{j,k'} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ for $k, k' \in [0, s_c-1]$.

Next, we apply StPolyDot codes to the practical source-worker-receiver framework, where each worker $i \in \mathcal{I}$ has bounded storage. To compute using multiple workers, the source nodes assign distinct coefficients x_i from $\mathbb{F}_q$ to different workers. The source nodes separately perform a linear preencoding[5] step on $\mathbf{A} \in \mathbb{F}_q^{\ell\times m}$ and $\mathbf{B} \in \mathbb{F}_q^{\ell\times m}$, for $q \geq 2$, respectively, to form the two polynomials $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ that incorporate the indeterminate x_i, for each worker $i \in \mathcal{I}$, reflecting the finite storage capabilities of workers:

$$\tilde{\mathbf{A}}_i = f_{1i}(\mathbf{A}) \triangleq \sum_{j=0}^{s_r-1} \sum_{k=0}^{s_c-1} \mathbf{A}_{j,k} x_i^k x_i^{s_c j} \in \mathbb{F}_q^{\frac{\ell}{s_r}\times\frac{m}{s_c}} \ , \quad i \in \mathcal{I} \ , \tag{4a}$$

$$\tilde{\mathbf{B}}_i = f_{2i}(\mathbf{B}) \triangleq \sum_{j=0}^{s_r-1} \sum_{k=0}^{s_c-1} \mathbf{B}_{j,k} x_i^{s_c(s_r-1-j)} x_i^{s_c(2s_r-1)k} \in \mathbb{F}_q^{\frac{\ell}{s_r}\times\frac{m}{s_c}} \ , \quad i \in \mathcal{I} \ , \tag{4b}$$

which will be used subsequently to construct the polynomials assigned to worker $i \in \mathcal{I}$ (cf. (9) in Section II-.c)), where $f_{1i}, \ f_{2i} \ : \ \mathbb{F}_q^{\ell\times m} \to \mathbb{F}_q^{\frac{\ell}{s_r}\times\frac{m}{s_c}}$ are preprocessing functions[6], and $s_r|\ell$ and $s_c|m$, where s_r and s_c describe the split of input

[3]The communication cost at finite block lengths can be captured using Kolmogorov's complexity [71, Chapter 14], and the communication complexity per source will increase approximately by the logarithm of the total length of the transmitted strings.

[4]The tuple (κ, κ) is called an (n, ϵ)-coding scheme if there exists a function $\psi : \mathcal{R}_\kappa \times \mathcal{R}_\kappa \to \boldsymbol{\mathcal{Z}}_i^n$ such that where $\mathcal{R}_\kappa$ and $\boldsymbol{\mathcal{Z}}^n$ denote the domain and the codomain, writing $\hat{\mathbf{Z}}_i^n \triangleq \psi(\kappa(\mathbf{X}_{i,1}^n \oplus_q \mathbf{X}_{i,2}^n))$, we have $\mathbb{P}(\hat{\mathbf{Z}}_i^n \neq \mathbf{Z}_i^n) < \epsilon$ [14].

[5]The memoryless and i.i.d. observations assumptions on $\mathbf{A}^n$ and $\mathbf{B}^n$, along with that the source nodes choose arbitrarily distinct elements $x_i \in \mathbb{F}_q$ across each worker $i \in \mathcal{I}$, ensure that for a given realization of $\mathbf{A}(k)$ and $\mathbf{B}(k)$, $k \in [n]$, the preencodings $\tilde{\mathbf{A}}_i(k)$ and $\tilde{\mathbf{B}}_i(k)$ given in (4) for each $i \in \mathcal{I}$ are i.i.d. across the set of workers $\mathcal{I}$.

[6]We refer the reader to [6, Remark V.1] for alternative ways of choosing the exponents of x_i's.

matrices in the number of rows and columns, respectively. We set $s = s_r s_c$, which is specified by a storage size constraint per worker, denoted by $\mu_{\text{StPolyDot}}$, and the relation between s and $\mu_{\text{StPolyDot}}$ will be clarified later on.

The product of preencodings in (4a) and (4b) yields

$$\tilde{\mathbf{A}}_i^{\intercal}\tilde{\mathbf{B}}_i = \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1}\sum_{j'=0}^{s_r-1}\sum_{k'=0}^{s_c-1} \mathbf{A}_{j,k}^{\intercal}\mathbf{B}_{j',k'} \times x_i^{k+s_c(s_r-1+j-j')+s_c(2s_r-1)k'} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}. \quad (5)$$

To recover $\mathbf{D} \in \mathbb{F}_q^{m\times m}$, the receiver must obtain the coefficient of the product polynomial $\sum_{j=0}^{s_r-1}\mathbf{A}_{j,k}^{\intercal}\mathbf{B}_{j,k'} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ for $k, k' \in [0, s_c-1]$, which we will detail below.

b) Non-Linear Transformations of the Preencodings: We next exploit the row-block representations of $\tilde{\mathbf{A}}_i \in \mathbb{F}_q^{\frac{\ell}{s_r}\times\frac{m}{s_c}}$ and $\tilde{\mathbf{B}}_i \in \mathbb{F}_q^{\frac{\ell}{s_r}\times\frac{m}{s_c}}$, allowing us to rewrite the preencodings as

$$\tilde{\mathbf{A}}_i = \left[\tilde{\mathbf{A}}_{i1}; \tilde{\mathbf{A}}_{i2}\right], \quad \tilde{\mathbf{B}}_i = \left[\tilde{\mathbf{B}}_{i1}; \tilde{\mathbf{B}}_{i2}\right], \quad i \in \mathcal{I}, \quad (6)$$

where the submatrices satisfy $\tilde{\mathbf{A}}_{i1}, \tilde{\mathbf{A}}_{i2} \in \mathbb{F}_q^{\frac{\ell}{2s_r}\times\frac{m}{s_c}}$ and $\tilde{\mathbf{B}}_{i1}, \tilde{\mathbf{B}}_{i2} \in \mathbb{F}_q^{\frac{\ell}{2s_r}\times\frac{m}{s_c}}$. Sources separately preprocess $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ for each $i \in \mathcal{I}$ to obtain the non-linear mappings $g_1(\tilde{\mathbf{A}}_i) \in \mathbb{F}_q^{\mu_{\text{StPolyDot}}}$ and $g_2(\tilde{\mathbf{B}}_i) \in \mathbb{F}_q^{\mu_{\text{StPolyDot}}}$, respectively. We next describe these transformations for both symmetric and square non-symmetric cases and specify the corresponding values of $\mu_{\text{StPolyDot}}$.

c) Structured Source Coding on the Non-Linear Transformations of the Preencodings: Given length-n i.i.d. realizations $\mathbf{A}^n$ and $\mathbf{B}^n$, the source nodes employ the structured linear source coding scheme of Körner-Marton [14] to the non-linear transformations $\{g_1(\tilde{\mathbf{A}}_i(k))\}_{k=1}^n$ and $\{g_2(\tilde{\mathbf{B}}_i(k))\}_{k=1}^n$ to further compress them for each $i \in \mathcal{I}$. We next provide a structured coding scheme employing the achievability construction from [14] and Proposition 1 from [72].

Proposition 1. *Given $\mathbf{A}^n$ and $\mathbf{B}^n$, the distributed sources with separate encoders employ a common random linear encoding given by $\kappa \in \mathbb{F}_q^{n'\mu_{\textit{StPolyDot}}\times n\mu_{\textit{StPolyDot}}}$, where $\kappa : \mathbb{F}_q^{n\mu_{\textit{StPolyDot}}} \to \mathbb{F}_q^{n'\mu_{\textit{StPolyDot}}}$. The source nodes employ Körner-Marton encoding [14] on $g_1(\tilde{\mathbf{A}}_i^n)$ and $g_2(\tilde{\mathbf{B}}_i^n)$. Specifically, sources apply the common encoding matrix to the non-linear transformations given by $\{g_1(\tilde{\mathbf{A}}_i(k))\}_{k=1}^n$, and $\{g_2(\tilde{\mathbf{B}}_i(k))\}_{k=1}^n$, and devise the structured encodings*

$$\kappa(\{g_1(\tilde{\mathbf{A}}_i(k))\}_{k=1}^n) \in \mathbb{F}_q^{n'\mu_{\textit{StPolyDot}}}, \quad \kappa(\{g_2(\tilde{\mathbf{B}}_i(k))\}_{k=1}^n) \in \mathbb{F}_q^{n'\mu_{\textit{StPolyDot}}}, \quad (7)$$

respectively, to be then transmitted to worker $i \in \mathcal{I}$ over noiseless and interference-free links, as indicated in Figure 1. Each source chooses

$$n'\mu_{\textit{StPolyDot}} \approx nH_q(\mathbf{p}_i(x_i)) \quad (8)$$

to ensure that worker $i \in \mathcal{I}$ can successfully recover the following length-n realizations of the polynomial evaluations at point x_i:

$$\mathbf{p}_i^n(x_i) = g_1(\tilde{\mathbf{A}}_i^n) \oplus_q g_2(\tilde{\mathbf{B}}_i^n) = \{\mathbf{p}_{i,1}(x_i), \ldots, \mathbf{p}_{i,k}(x_i), \ldots, \mathbf{p}_{i,n}(x_i)\} \quad (9)$$

where each component is given by

$$\mathbf{p}_{i,k}(x_i) = g_1(\tilde{\mathbf{A}}_i(k)) \oplus_q g_2(\tilde{\mathbf{B}}_i(k)), \quad k \in [n]. \quad (10)$$

The structured encodings (7) will ensure that worker $i \in \mathcal{I}$ successfully recovers the sequence $\mathbf{p}_i^n(x_i)$ in (9) in an asymptotically lossless manner, as demonstrated below (cf. Section II-.d)).

Proof. The proof builds upon the structured source coding scheme in [14] to develop a novel polynomial coding framework. (8) follows from a generalization of the achievability scheme in [14], devised for structural compression of binary sequences X_1^n and X_2^n. Specifically, based on the method of Elias [51] (see also [52, pp. 206-207]), [14] constructs linear codes that achieve an asymptotically vanishing probability of error at the minimum sum rate of $2H(X_1 \oplus_2 X_2)$ [14].

The proof of [14, Theorem 1] is a direct application of Elias's lemma, and is given next.

Lemma 1 (Elias [52])**.** *Let $\{Z_i\}_{i=1}^\infty$ be an i.i.d. binary sequence. For any fixed $\epsilon > 0$ and sufficiently large n, there exists a matrix $\kappa \in \mathbb{F}_2^{n'\times n}$ and a function $\psi : \mathbb{F}_2^{n'} \to \mathbb{F}_2^n$ such that*

1) *$n' < n(H(Z) + \epsilon)$,*
2) *letting $\kappa(z^n)$ denote the modulo-two product of the matrix κ with $z^n \in \mathbb{F}_q^n$, we have*

$$\mathbb{P}(\kappa(Z^n) \neq Z^n) < \epsilon. \quad (11)$$

The rest of the proof of [14, Theorem 1] is as follows. Lemma 1 is applied to the sequence $\{Z_i\}$ of the modulo-two adder source network with symmetric distribution and the function $\kappa(\cdot)$ is chosen to be the code of both X_1^n and X_2^n, namely the encoder outputs are $\kappa(X_1^n)$ and $\kappa(X_2^n)$, respectively. Further, we define the function $\phi : \{0,1\}^{n'} \times \{0,1\}^{n'} \to \{0,1\}^n$ by

$$\phi(\mathbf{a}^{n'}, \mathbf{b}^{n'}) \triangleq \psi(\mathbf{a}^{n'} \oplus_2 \mathbf{b}^{n'})$$

where $\mathbf{a}^{n'}$ and $\mathbf{b}^{n'}$ are binary sequences of length n' and $\mathbf{a}^{n'} \oplus_2 \mathbf{b}^{n'}$ denotes their modulo-two sum. Noting that the mapping κ is linear, 2) implies

$$\mathbb{P}(\phi(\kappa(X_1^n), \kappa(X_2^n)) \neq \mathbf{Z}^n) = \mathbb{P}(\psi(\kappa(X_1^n \oplus_2 X_2^n)) \neq Z^n) = \mathbb{P}(\psi(\kappa(Z^n)) \neq Z^n) < \epsilon. \quad (12)$$

Thus (κ, κ) is an (n, ϵ)-coding scheme. Further, since $\mathcal{R}_\kappa = \{0,1\}^{n'}$, 1) implies

$$n^{-1}\log\|\mathcal{R}_\kappa\| < H(Z) + \epsilon. \quad (13)$$

Thus, the achievability of the tuple $(H(Z), H(Z))$ is proved.

Exploiting [72, Proposition 1] provides a generalization of [14, Theorem 1] to the case of q-ary vector sequences, and its proof results from the application of the Han and Kobayashi scheme [15], and is detailed in [72]. Hence, application of [72, Proposition 1] implies the achievability of the tuple $(H_q(\mathbf{p}_i(x_i)), H_q(\mathbf{p}_i(x_i)))$, which yields the result in (8). □

Definition	Symbol
Block length	n
Source matrices	$\mathbf{A} \in \mathbb{F}_q^{\ell \times m}$ and $\mathbf{B} \in \mathbb{F}_q^{\ell \times m}$
Number of column-blocks; number of row-blocks in $\mathbf{A}$ and $\mathbf{B}$	s_c; s_r, where $s = s_r s_c$
Block-encoded source matrices with block length n	$\mathbf{A}^n$ and $\mathbf{B}^n$
Number of distributed workers; set of distributed workers	I; $\mathcal{I} = [I]$
Linear preencodings devised for worker $i \in \mathcal{I}$	$\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$
Non-linear transformations devised for worker $i \in \mathcal{I}$	$g_1(\tilde{\mathbf{A}}_i)$ and $g_2(\tilde{\mathbf{B}}_i)$
Körner-Marton encodings to be transmitted to worker $i \in \mathcal{I}$	$\kappa(g_1(\tilde{\mathbf{A}}_i^n))$ and $\kappa(g_2(\tilde{\mathbf{B}}_i^n))$
Polynomials reconstructed at worker $i \in \mathcal{I}$	$\mathbf{p}_i(x_i) = g_1(\tilde{\mathbf{A}}_i) \oplus_q g_2(\tilde{\mathbf{B}}_i)$
Intermediate computations performed by worker $i \in \mathcal{I}$	$p_i(x_i)$
The storage size constraint per worker for a given coding scheme	μ_{scheme}
Recovery threshold for a given coding scheme	$I_{r_{\text{scheme}}}$
Total computation cost of the source nodes	Γ_{source}
The computation cost per worker	Γ_{worker}
The ratio of the worker storage required by PolyDot to that of StPolyDot	η_S
The ratio of the total computation cost of StPolyDot to that of PolyDot	χ_{Comp}
The ratio of the total communication cost of PolyDot to that of StPolyDot	η_{Comm}

TABLE I: Notation.

Our structured codes admit a simplified form when the target matrix $\mathbf{D} = \mathbf{A}^\intercal\mathbf{B}$ is symmetric, i.e., $\mathbf{D} = \mathbf{D}^\intercal$ while remaining applicable without requiring symmetry, as will be discussed subsequently (cf. (19)). When $\mathbf{D}$ is symmetric, the sources then use the preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ in (4) with the representations in (6) and the non-linear transformations given as

$$g_1(\tilde{\mathbf{A}}_i) = \left[\tilde{\mathbf{A}}_{i2}; \tilde{\mathbf{A}}_{i1}; \tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{A}}_{i1}\right] \in \mathbb{F}_q^{\mu^{\text{sym.}}_{\text{StPolyDot}}} , \tag{14a}$$

$$g_2(\tilde{\mathbf{B}}_i) = \left[\tilde{\mathbf{B}}_{i1}; \tilde{\mathbf{B}}_{i2}; \tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{B}}_{i2}\right] \in \mathbb{F}_q^{\mu^{\text{sym.}}_{\text{StPolyDot}}} \tag{14b}$$

where $\mu^{\text{sym.}}_{\text{StPolyDot}} = \left(\frac{\ell}{s_r} + \frac{m}{s_c}\right) \times \frac{m}{s_c}$. The distributed sources employ a common random linear encoding $\kappa : \mathbb{F}_q^{n\mu_{\text{StPolyDot}}} \to \mathbb{F}_q^{n'\mu_{\text{StPolyDot}}}$ to $g_1(\tilde{\mathbf{A}}_i^n)$ and $g_2(\tilde{\mathbf{B}}_i^n)$, yielding:

$$\kappa(g_1(\tilde{\mathbf{A}}_i^n)) , \quad \kappa(g_2(\tilde{\mathbf{B}}_i^n)) , \tag{15}$$

respectively, to be then transmitted to worker $i \in \mathcal{I}$, where $n' \approx nH_q(\mathbf{p}_i(x_i))$ from (8) — with the corresponding polynomials $\mathbf{p}_i(x_i)$ detailed next —, which ensures that worker $i \in \mathcal{I}$ successfully recovers the following three polynomial evaluations at point x_i (cf. (10)):

$$\begin{aligned}\mathbf{p}_{i,k}(x_i) &= \left[p_{i,k}^{(j)}(x_i)\right]_{j=1}^{3} \\ &= \left[p_{i,k}^{(1)}(x_i) \; ; \; p_{i,k}^{(2)}(x_i) \; ; \; p_{i,k}^{(3)}(x_i)\right] , \quad i \in \mathcal{I} \; , \; k \in [n]\end{aligned} \tag{16}$$

in an asymptotically lossless manner, as functions of both $\mathbf{A}$ and $\mathbf{B}$, where the *linear polynomials* $p_i^{(1)}(x_i)$ and $p_i^{(2)}(x_i)$ result from linear processing of the submatrices of $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ in (6), while the *non-linear parity polynomial* $p_i^{(3)}(x_i)$ is due to non-linear processing of submatrices in (6):

$$p_i^{(1)}(x_i) \triangleq \tilde{\mathbf{A}}_{i2} \oplus_q \tilde{\mathbf{B}}_{i1} \tag{17a}$$

$$\begin{aligned}&= \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{A}_{j2,k} x_i^k x_i^{s_c j} \\ &\oplus_q \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{B}_{j1,k} x_i^{s_c(s_r-1-j)} x_i^{s_c(2s_r-1)k} \in \mathbb{F}_q^{\frac{\ell}{2s_r}\times\frac{m}{s_c}} ,\end{aligned}$$

$$p_i^{(2)}(x_i) \triangleq \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2} \tag{17b}$$

$$\begin{aligned}&= \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{A}_{j1,k} x_i^k x_i^{s_c j} \\ &\oplus_q \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{B}_{j2,k} x_i^{s_c(s_r-1-j)} x_i^{s_c(2s_r-1)k} \in \mathbb{F}_q^{\frac{\ell}{2s_r}\times\frac{m}{s_c}} ,\end{aligned}$$

$$p_i^{(3)}(x_i) \triangleq \tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{B}}_{i2} \tag{17c}$$

$$\begin{aligned}&= \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1}\sum_{j'=0}^{s_r-1}\sum_{k'=0}^{s_c-1} \mathbf{A}_{j2,k}^\intercal \mathbf{A}_{j'1,k'} x_i^{k+k'} x_i^{s_c(j+j')} \\ &\oplus_q \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1}\sum_{j'=0}^{s_r-1}\sum_{k'=0}^{s_c-1} \mathbf{B}_{j1,k}^\intercal \mathbf{B}_{j'2,k'} \\ &\quad \times x_i^{s_c(2s_r-2-j-j')} x_i^{s_c(2s_r-1)(k+k')} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}\end{aligned}$$

where, to simplify the notation, we dropped the index k of realizations from the polynomials $\mathbf{p}_{i,k}(x_i)$ for each $i \in \mathcal{I}$, for each realization of $\mathbf{A}(k)$ and $\mathbf{B}(k)$, whenever it is clear from the context. We note that although the parity polynomial $p_i^{(3)}(x_i)$ is nonlinear due to products such as $\tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{A}}_{i1} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ and $\tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{B}}_{i2} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$, it is still linearly separable in the preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$, enabling the convenient use of the Körner-Marton encoding scheme [14].

As we will demonstrate later, each distributed worker $i \in \mathcal{I}$ exploits (17) to derive $p_i(x_i)$ (cf. (23) in Section II-.d)), and a collection of $\{p_i(x_i)\}_{i\in\mathcal{I}}$, each with a specified degree

as determined by the memory parameters (cf. (27)), will be subsequently collected by the receiver to determine the desired matrix product $\mathbf{D} = \mathbf{A}^{\intercal}\mathbf{B}$ (cf. Section II-.e)).

Our model can also capture $\mathbf{A} \in \mathbb{F}_q^{\ell \times m}$ and $\mathbf{B} \in \mathbb{F}_q^{\ell \times m'}$ where $m' \neq m$. We next exploit the row-block representations of $\tilde{\mathbf{A}}_i \in \mathbb{F}_q^{\frac{\ell}{s_r} \times \frac{m}{s_c}}$ and $\tilde{\mathbf{B}}_i \in \mathbb{F}_q^{\frac{\ell}{s_r} \times \frac{m'}{s_c}}$, allowing us to rewrite preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ as functions of submatrices according to (6), where $\tilde{\mathbf{A}}_{i1}, \tilde{\mathbf{A}}_{i2} \in \mathbb{F}_q^{\frac{\ell}{2s_r} \times \frac{m}{s_c}}$ and $\tilde{\mathbf{B}}_{i1}, \tilde{\mathbf{B}}_{i2} \in \mathbb{F}_q^{\frac{\ell}{2s_r} \times \frac{m'}{s_c}}$. The sources then obtain the non-linear transformations, as determined by (14), where $g_1(\tilde{\mathbf{A}}_i) = \left[\tilde{\mathbf{A}}_{i2}; \tilde{\mathbf{A}}_{i1}; \tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{A}}_{i1}\right] \in \mathbb{F}_q^{\mu}$ and $g_2(\tilde{\mathbf{B}}_i) = \left[\tilde{\mathbf{B}}_{i1}; \tilde{\mathbf{B}}_{i2}; \tilde{\mathbf{B}}_{i1}^{\intercal}\tilde{\mathbf{B}}_{i2}\right] \in \mathbb{F}_q^{\mu'}$, with $\mu = \left(\frac{\ell}{s_r} + \frac{m}{s_c}\right) \times \frac{m}{s_c}$ and $\mu' = \left(\frac{\ell}{s_r} + \frac{m'}{s_c}\right) \times \frac{m'}{s_c}$. Assume $m' > m$. We then zero-pad $g_1(\tilde{\mathbf{A}}_i)$ to obtain

$$g_1^{\text{ZP}}(\tilde{\mathbf{A}}_i) = \begin{bmatrix} g_1(\tilde{\mathbf{A}}_i) & \mathbf{0}_{\left(\frac{\ell}{s_r} + \frac{m}{s_c}\right) \times \frac{m'-m}{s_c}} \\ \mathbf{0}_{\frac{m'-m}{s_c} \times \frac{m}{s_c}} & \mathbf{0}_{\frac{m'-m}{s_c} \times \frac{m'-m}{s_c}} \end{bmatrix} \in \mathbb{F}_q^{\mu'} \tag{18}$$

which ensures that the lengths of $g_1^{\text{ZP}}(\tilde{\mathbf{A}}_i)$ and $g_2(\tilde{\mathbf{B}}_i)$ are matched. Otherwise, i.e., if $m < m'$, we zero-pad $g_2(\tilde{\mathbf{B}}_i)$ to ensure the lengths are compatible. Next, given the matched dimensions, the distributed sources employ a common random linear encoding given by $\kappa : \mathbb{F}_q^{n\mu'} \to \mathbb{F}_q^{n'\mu'}$, to $g_1^{\text{ZP}}(\tilde{\mathbf{A}}_i^n)$, and $g_2(\tilde{\mathbf{B}}_i^n)$, and devise the structured encodings $\kappa(g_1^{\text{ZP}}(\tilde{\mathbf{A}}_i^n))$ and $\kappa(g_2(\tilde{\mathbf{B}}_i^n))$, respectively, to be then transmitted to worker $i \in \mathcal{I}$, where $n' \approx nH_q(\mathbf{p}_{\text{ZP},i}(x_i))$ from (8). Worker i subsequently recovers $\mathbf{p}_{\text{ZP},i}^n(x_i) = g_1^{\text{ZP}}(\tilde{\mathbf{A}}_i^n) \oplus_q g_2(\tilde{\mathbf{B}}_i^n)$ in an asymptotically lossless manner.

When $\mathbf{D} = \mathbf{A}^{\intercal}\mathbf{B}$ is non-symmetric, i.e., $\mathbf{D} \neq \mathbf{D}^{\intercal}$, the sources employ the preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ in (4) with the representations in (6) and the non-linear transformations given by

$$g_1(\tilde{\mathbf{A}}_i) = \left[\tilde{\mathbf{A}}_{i1}; \tilde{\mathbf{A}}_{i2}; \mathbf{0}_{\frac{\ell}{2s_r} \times \frac{m}{s_c}}; \tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{A}}_{i1}\right] \in \mathbb{F}_q^{\mu_{\text{StPolyDot}}^{\text{non-sym.}}}, \tag{19a}$$

$$g_2(\tilde{\mathbf{B}}_i) = \left[\tilde{\mathbf{B}}_{i2}; \mathbf{0}_{\frac{\ell}{2s_r} \times \frac{m}{s_c}}; \tilde{\mathbf{B}}_{i1}; \tilde{\mathbf{B}}_{i1}^{\intercal}\tilde{\mathbf{B}}_{i2}\right] \in \mathbb{F}_q^{\mu_{\text{StPolyDot}}^{\text{non-sym.}}} \tag{19b}$$

where $\mu_{\text{StPolyDot}}^{\text{non-sym.}} = \left(\frac{3\ell}{2s_r} + \frac{m}{s_c}\right) \times \frac{m}{s_c}$. The sources then devise structured encodings using (15), where $n' \approx nH_q(\mathbf{p}_i(x_i))$ from (8), to be then transmitted to worker $i \in \mathcal{I}$, which ensures that worker $i \in \mathcal{I}$ successfully recovers the following four polynomial evaluations at point x_i (cf. (10)):

$$\begin{aligned} \mathbf{p}_{i,k}(x_i) &= \left[p_{i,k}^{(j)}(x_i)\right]_{j=1}^{4} \\ &= \left[p_{i,k}^{(1)}(x_i)\ ;\ p_{i,k}^{(2)}(x_i)\ ;\ p_{i,k}^{(3)}(x_i)\ ;\ p_{i,k}^{(4)}(x_i)\right], \\ &\quad i \in \mathcal{I}\ ,\ k \in [n] \end{aligned} \tag{20}$$

in an asymptotically lossless manner, as functions of both $\mathbf{A}$ and $\mathbf{B}$:

$$\begin{aligned} p_i^{(1)}(x_i) &\triangleq \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2} \\ &= \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{A}_{j1,k} x_i^k x_i^{s_c j} \\ &\quad \oplus_q \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{B}_{j2,k} x_i^{s_c(s_r-1-j)} x_i^{s_c(2s_r-1)k} \in \mathbb{F}_q^{\frac{\ell}{2s_r} \times \frac{m}{s_c}} \end{aligned} \tag{21a}$$

$$p_i^{(2)}(x_i) \triangleq \tilde{\mathbf{A}}_{i2} = \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{A}_{j2,k} x_i^k x_i^{s_c j} \in \mathbb{F}_q^{\frac{\ell}{2s_r} \times \frac{m}{s_c}}\ , \tag{21b}$$

$$\begin{aligned} p_i^{(3)}(x_i) &\triangleq \tilde{\mathbf{B}}_{i1} = \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1} \mathbf{B}_{j1,k} x_i^{s_c(s_r-1-j)} x_i^{s_c(2s_r-1)k} \\ &\quad \in \mathbb{F}_q^{\frac{\ell}{2s_r} \times \frac{m}{s_c}}\ , \end{aligned} \tag{21c}$$

$$\begin{aligned} p_i^{(4)}(x_i) &\triangleq \tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2}^{\intercal}\tilde{\mathbf{B}}_{i1} \\ &= \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1}\sum_{j'=0}^{s_r-1}\sum_{k'=0}^{s_c-1} \mathbf{A}_{j2,k}^{\intercal}\mathbf{A}_{j'1,k'} x_i^{k+k'} x_i^{s_c(j+j')} \\ &\quad \oplus_q \sum_{j=0}^{s_r-1}\sum_{k=0}^{s_c-1}\sum_{j'=0}^{s_r-1}\sum_{k'=0}^{s_c-1} \mathbf{B}_{j2,k}^{\intercal}\mathbf{B}_{j'1,k'} \\ &\quad \times x_i^{s_c(2s_r-2-j-j')} x_i^{s_c(2s_r-1)(k+k')} \in \mathbb{F}_q^{\frac{m}{s_c} \times \frac{m}{s_c}}\ . \end{aligned} \tag{21d}$$

Establishing the steps $a) - c)$ to encoding, we next describe the processing by the workers.

d) Intermediate Computations at Distributed Workers: Each distributed worker $i \in \mathcal{I}$ reconstructs the sequence of polynomial evaluations $\mathbf{p}_i^n(x_i)$ given in (9), with $\mathbf{p}_{i,k}(x_i) = g_1(\tilde{\mathbf{A}}_i(k)) \oplus_q g_2(\tilde{\mathbf{B}}_i(k))$ given in (16) for the symmetric case, and in (20) for the non-symmetric case, in an asymptotically lossless manner from the structured source encodings $\kappa(\{g_1(\tilde{\mathbf{A}}_i(k))\}_{k=1}^n)$ and $\kappa(\{g_2(\tilde{\mathbf{B}}_i(k))\}_{k=1}^n)$ in (7), given by Proposition 1. Similarly to [14, Theorem 1] and from [72, Proposition 1], the decoding argument draws upon Elias's lemma [52] (cf. Lemma 1), which states that for an i.i.d. binary sequence $\{Z_i\}_{i=1}^{\infty}$, and for any fixed $\epsilon > 0$ and sufficiently large n, there exist a binary matrix[7] $\kappa \in \mathbb{F}_2^{n' \times n}$ and a decoding function $\psi : \mathbb{F}_2^{n'} \to \mathbb{F}_2^n$ such that (8) holds with an average decoding error probability bounded by ϵ (cf. Proposition 1).

Upon reconstructing $\mathbf{p}_i^n(x_i)$, worker $i \in \mathcal{I}$ then performs post-processing on these polynomials to generate a sequence of computational outputs to be sent to the receiver, which we detail below.

Consider the following special case, where the following symmetry condition holds:

$$\begin{aligned} &\mathbf{B}_{j'1,k'}^{\intercal}\mathbf{A}_{j1,k} = \mathbf{A}_{j1,k}^{\intercal}\mathbf{B}_{j'1,k'}, \\ &\qquad \forall\, j, j' \in [0, s_r - 1],\ k, k' \in [0, s_c - 1]. \end{aligned} \tag{22}$$

Then, using the polynomials in (17), worker i computes

$$\begin{aligned} p_i(x_i) &= \left(p_i^{(1)}(x_i)\right)^{\intercal} \cdot p_i^{(2)}(x_i) \ominus_q p_i^{(3)}(x_i) \\ &= (\tilde{\mathbf{A}}_{i2} \oplus_q \tilde{\mathbf{B}}_{i1})^{\intercal} \cdot (\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2}) \\ &\quad \ominus_q (\tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i1}^{\intercal}\tilde{\mathbf{B}}_{i2}) \\ &= \tilde{\mathbf{B}}_{i1}^{\intercal}\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{B}}_{i2} \overset{(a)}{=} \tilde{\mathbf{A}}_{i1}^{\intercal}\tilde{\mathbf{B}}_{i1} \oplus_q \tilde{\mathbf{A}}_{i2}^{\intercal}\tilde{\mathbf{B}}_{i2} = \tilde{\mathbf{A}}_i^{\intercal}\tilde{\mathbf{B}}_i \end{aligned} \tag{23}$$

where (a) holds true as (22) guarantees the same symmetry condition for $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ due to the linearity of the preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ in (4). The worker then transmits $p_i(x_i)$ to the receiver.

[7] From [73, Appendix IV, Proof of Theorem 10, p. 411], a random linear mapping $\kappa \in \mathbb{F}_2^{n' \times n}$, whose components are all chosen i.i.d. and uniformly from $\mathbb{F}_2$, yields an (n, ϵ)-coding scheme (cf. (13)). The case $q > 2$ is considered in [15, Lemma 4].

If the above-mentioned symmetry in (22) does not hold, but the relaxed condition $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i = \tilde{\mathbf{B}}_i^\intercal \tilde{\mathbf{A}}_i$ for all $i \in \mathcal{I}$ is satisfied, worker i computes

$$\begin{aligned} p_i(x_i) &= \big(p_i^{(1)}(x_i)\big)^\intercal \cdot p_i^{(2)}(x_i) \ominus_q p_i^{(3)}(x_i) \\ &= \tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{B}}_{i2} \end{aligned} \tag{24}$$

and transmits $p_i(x_i)$ to the receiver. The receiver can then easily derive

$$\begin{aligned} \tilde{p}_i(x_i) &= \frac{1}{2}(p_i(x_i) \oplus_q \big(p_i(x_i)\big)^\intercal) \\ &\overset{(a)}{=} \frac{1}{2}(\tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{B}}_{i2}) \oplus_q \frac{1}{2}(\tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{B}}_{i2})^\intercal \\ &= \frac{1}{2}(\tilde{\mathbf{B}}_{i1}^\intercal \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{B}}_{i2} \oplus_q \tilde{\mathbf{A}}_{i1}^\intercal \tilde{\mathbf{B}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2}^\intercal \tilde{\mathbf{A}}_{i2}) \\ &= \frac{1}{2}(\tilde{\mathbf{B}}_i^\intercal \tilde{\mathbf{A}}_i \oplus_q \tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i) \overset{(b)}{=} \tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i \end{aligned} \tag{25}$$

where (a) follows from using (24) and employing the expression for $\big(p_i(x_i)\big)^\intercal$, and (b) from employing the definitions of $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ in (4a) and (4b), and using $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i = \tilde{\mathbf{B}}_i^\intercal \tilde{\mathbf{A}}_i$.

For the case of non-symmetric matrices, employing (21), worker $i \in \mathcal{I}$ computes

$$\begin{aligned} p_i(x_i) &= p_i^{(2)}(x_i)^\intercal \cdot p_i^{(1)}(x_i) \oplus_q p_i^{(1)}(x_i)^\intercal \cdot p_i^{(3)}(x_i) \ominus_q p_i^{(4)}(x_i) \\ &= \tilde{\mathbf{A}}_{i2}^\intercal (\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2}) \oplus_q (\tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2})^\intercal \tilde{\mathbf{B}}_{i1} \\ &\ominus_q (\tilde{\mathbf{A}}_{i2}^\intercal \tilde{\mathbf{A}}_{i1} \oplus_q \tilde{\mathbf{B}}_{i2}^\intercal \tilde{\mathbf{B}}_{i1}) = \tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i \end{aligned} \tag{26}$$

and then transmits the intermediate computation $p_i(x_i)$ to the receiver.

In all these cases, if the worker is successful (no delay or error), it sends $p_i(x_i)$ to the receiver. However, the workers may straggle, and produce delayed or erroneous outputs.

e) Computation of the Desired Matrix Product by the Receiver: The receiver seeks to determine the desired matrix product $\mathbf{D} = \mathbf{A}^\intercal \mathbf{B}$ by collecting and post-processing the computational outputs of the workers, i.e., from a subset of $\{\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i\}_{i \in \mathcal{I}}$. The minimum number of workers needed for the receiver to successfully evaluate $\mathbf{D}$ is known as the recovery threshold, and we denote the recovery threshold for StPolyDot codes by $I_{r_{\text{StPolyDot}}}$ where $I_{r_{\text{StPolyDot}}} \leq I$. Here, we rely on a worst-case scenario such that the outputs of any $I_{r_{\text{StPolyDot}}}$ workers are sufficient to perform the desired task. Otherwise, the computation task cannot be realized.

Exploiting the product $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$ in (5), the degree of $p_i(x_i) = \tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$ in (26) is expressed as

$$\begin{aligned} \deg(p_i) &= (s_c - 1) + s_c(s_r - 1 + s_r - 1) \\ &\quad + s_c(2s_r - 1)(s_c - 1) = s_c^2(2s_r - 1) - 1 \ . \end{aligned} \tag{27}$$

Therefore, the recovery threshold of the StPolyDot scheme is

$$I_{r_{\text{StPolyDot}}} = s_c^2(2s_r - 1) \ . \tag{28}$$

The scheme extends naturally to arbitrary bilinear functions. Let $\mathbf{F} = B(\mathbf{e}_i, \mathbf{e}_j) \in \mathbb{F}_q^{\ell \times \ell}$ be the matrix representation of the bilinear form. For $\mathbf{a},\ \mathbf{b} \in \mathbb{F}^{\ell \times 1}$, $B(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\intercal \mathbf{F} \mathbf{b} = \sum_{i,j \in [\ell]} a_i f_{ij} b_j$. Thus, defining $\tilde{\mathbf{b}} = \mathbf{F}\mathbf{b}$ reduces computing $B(\mathbf{a}, \mathbf{b})$ to the distributed inner product of $\mathbf{a}$ and $\tilde{\mathbf{b}}$.

We next detail the cost analysis for StPolyDot codes.

III. Storage, Computation, and Communication Cost of StPolyDot Codes

In this section, under the source-worker-receiver framework, we contrast the StPolyDot coding framework with state-of-the-art coded computing techniques. After reviewing PolyDot codes in Section III-A, we characterize the storage, computation, and communication costs of StPolyDot in Sections III-B-III-D, and compare them with those of PolyDot in Propositions 2, 4, and 5.

A. A Primer on PolyDot Codes

Polynomial codes in general assume that a centralized node has full access to $\mathbf{A},\ \mathbf{B} \in \mathbb{F}_q^{\ell \times m}$, and encoding is separated in $\mathbf{A}$ and $\mathbf{B}$. Although joint encoding is possible in centralized settings, it has not, to our knowledge, been leveraged in prior work. Consider the row and column-block representation in (2). For PolyDot codes, as introduced in [6], we define the linear preencodings

$$\tilde{\mathbf{A}}_i = \sum_{j=0}^{s_r-1} \sum_{k=0}^{s_c-1} \mathbf{A}_{j,k} x_i^k x_i^{s_c j} \in \mathbb{F}_q^{\frac{\ell}{s_r} \times \frac{m}{s_c}} \ , \tag{29a}$$

$$\tilde{\mathbf{B}}_i = \sum_{j=0}^{s_r-1} \sum_{k=0}^{s_c-1} \mathbf{B}_{j,k} x_i^{s_c(s_r-1-j)} x_i^{s_c(2s_r-1)k} \in \mathbb{F}_q^{\frac{\ell}{s_r} \times \frac{m}{s_c}} \ . \tag{29b}$$

For a fixed value of the memory parameter $s = s_r s_c$, setting $s_c = 1$ yields MatDot codes [32]. On the other hand, setting $s_r = 1$ yields Polynomial codes [5].

The centralized node sends to each worker $i \in \mathcal{I}$ (cf. 29) two polynomials, $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ determined by linear transformations of $\mathbf{A}$ and $\mathbf{B}$, respectively. The product of $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ is

$$\begin{aligned} \tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i = &\sum_{j=0}^{s_r-1} \sum_{k=0}^{s_c-1} \sum_{j'=0}^{s_r-1} \sum_{k'=0}^{s_c-1} \mathbf{A}_{j,k}^\intercal \mathbf{B}_{j',k'} \\ &\times x_i^{k+s_c(s_r-1+j-j')+s_c(2s_r-1)k'} \in \mathbb{F}_q^{\frac{m}{s_c} \times \frac{m}{s_c}}. \end{aligned} \tag{30}$$

Thus, from (5), $\mathbf{A}^\intercal \mathbf{B}$ can be recovered by interpolating the polynomial $p_i(x_i) = \tilde{\mathbf{A}}_i^\top \tilde{\mathbf{B}}_i$ (cf. (23)) from sufficiently many evaluations over $i \in \mathcal{I}_{r_{\text{PolyDot}}}$.

For PolyDot codes, the memory required per worker in the number of bits for each $k \in [n]$ is

$$\mu_{\text{PolyDot}} = \frac{2\ell m}{s_r s_c} \ . \tag{31}$$

For PolyDot codes, from (29), computation of $\tilde{\mathbf{A}}_i$ (and of $\tilde{\mathbf{B}}_i$) requires the addition of $s_c \cdot s_r$ matrices each of size $\frac{\ell}{s_r} \times \frac{m}{s_c}$. Thus, the total computation complexity of the sources is

$$\Gamma_{\text{source}}^{\text{PolyDot}} = I \cdot 2\ell m \ . \tag{32}$$

Using the assigned polynomials in (29), worker $i \in \mathcal{I}$ evaluates $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$. The computational complexity is measured in terms of the number of multiplications and the number of additions required. Each element of $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i \in \mathbb{F}_q^{\frac{m}{s_c} \times \frac{m}{s_c}}$ is given by an inner product $\tilde{\mathbf{A}}_{i,k}^\intercal \tilde{\mathbf{B}}_{i,k'}$ that involves $\frac{\ell}{s_r}$ multiplications

and $\frac{\ell}{s_r}-1$ additions. Therefore, the cost of computing $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$ by worker i is

$$\Gamma_{\text{worker}}^{\text{PolyDot}} = \Big(\frac{2\ell}{s_r}-1\Big)\frac{m^2}{s_c^2} \ . \tag{33}$$

Because the source communicates to a worker i the polynomials $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$, exploiting (29), the total cost of communication from the source to all workers satisfies

$$H_q(\{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i\}_{i\in\mathcal{I}}) \leq I \cdot \mu_{\text{PolyDot}} \ . \tag{34}$$

The cost of communication from each worker to the receiver is $H_q(p_i(x_i)) = \frac{m^2}{s_c^2}$. Given the recovery threshold $I_{r_{\text{PolyDot}}} = s_c^2(2s_r-1)$, meaning that only $I_{r_{\text{PolyDot}}}$ workers out of I succeed in transmitting, the total cost of communication from the workers to the receiver is

$$H_q(\{p_i(x_i)\}_{i\in\mathcal{I}_{r_{\text{PolyDot}}}}) \leq I_{r_{\text{PolyDot}}}\frac{m^2}{s_c^2} = (2s_r-1)m^2 \tag{35}$$

where equality is achieved when the elements of $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$ are all i.i.d. and uniform over $\mathbb{F}_q$.

We next continue with the cost analysis of StPolyDot codes.

B. Storage Cost of StPolyDot Codes

Modern computing systems are often limited by memory bandwidth rather than computation. In workloads such as GPUs, LLM inference, and distributed ML, arithmetic operations are relatively inexpensive, whereas memory access dominates runtime. This effect becomes particularly pronounced for large matrices $\mathbf{A},\ \mathbf{B} \in \mathbb{F}_q^{\ell\times m}$, where data movement outweighs the cost of arithmetic operations. For example, computing $\mathbf{A}^\intercal\mathbf{B}$ requires $2m^2\ell$ FLOPs but involves loading and storing $m\ell + m\ell + mm$ entries from memory. A common approach to mitigate this bottleneck is lossy matrix compression, which reduces data transfers at the expense of some precision [3].

While read operations are typically faster than writes, for sufficiently large block lengths, startup overheads and read–write asymmetries become negligible. In such regimes, system performance is often characterized primarily by communication and storage rates.

Unlike prior one-shot models, we consider block processing with joint decoding of the output sequence $\mathbf{D}^n$, which requires buffering blocks of length n (Section II-.c)). To capture the resulting $O(n)$ factor, performance can be analyzed at finite block lengths, where the average error probability satisfies $P_e \leq \exp(-nE_r(R))$, with $E_r(R)$ denoting the random-coding exponent for rate $R = \frac{n'}{n}$ (cf. [52, Theorem 6.2.1]). A detailed finite-length analysis is left to future work.

In the proposed StPolyDot scheme, each worker stores $\mathbf{p}_i^n(x_i)$ in (9) over a block of length n. In the symmetric case, from (17), $p_i^{(1)}(x_i) \in \mathbb{F}_q^{\frac{\ell}{2s_r}\times\frac{m}{s_c}}$, $p_i^{(2)}(x_i) \in \mathbb{F}_q^{\frac{\ell}{2s_r}\times\frac{m}{s_c}}$, and $p_i^{(3)}(x_i) \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$. Thus, the memory required per worker measured in bits at each time step $k\in[n]$ is given by

$$\mu_{\text{StPolyDot}}^{\text{sym.}} = \frac{m}{s_c}\left(\frac{\ell}{s_r}+\frac{m}{s_c}\right) \ . \tag{36}$$

This expression represents the average memory requirement per worker, computed as the total volume of data processed by that worker, normalized by the storage duration (i.e., over a block length n). It therefore characterizes the storage rate at each worker.

For the non-symmetric case, using (21), we similarly evaluate the memory requirement:

$$\mu_{\text{StPolyDot}}^{\text{non-sym.}} = \frac{m}{s_c}\left(\frac{3\ell}{2s_r}+\frac{m}{s_c}\right) \ . \tag{37}$$

Contrasting (36) and (37) with (31), it is easy to observe that $\mu_{\text{StPolyDot}}^{\text{sym.}} \leq \mu_{\text{PolyDot}}$ when $\frac{m}{s_c} \leq \frac{\ell}{s_r}$ and $\mu_{\text{StPolyDot}}^{\text{non-sym.}} \leq \mu_{\text{PolyDot}}$ when $\frac{m}{s_c} \leq \frac{\ell}{2s_r}$, respectively.

Proposition 2. (Achievable gain in the storage size of the workers.) *Given $\mathbf{A} \in \mathbb{F}_q^{\ell\times m}$ and $\mathbf{B} \in \mathbb{F}_q^{\ell\times m}$ with the row and column-block representations given in (2) according to memory parameters s_r and s_c, the ratios of the worker storage requirement for PolyDot codes to that of StPolyDot codes in the symmetric and non-symmetric cases are*

$$\eta_S^{\textit{sym.}} = \frac{2\ell}{\ell+\frac{ms_r}{s_c}} \ , \quad \eta_S^{\textit{non-sym.}} = \frac{2\ell}{\frac{3\ell}{2}+\frac{ms_r}{s_c}} \tag{38}$$

where the ratios $\eta_S^{\textit{sym.}}$ approaches 2 and $\eta_S^{\textit{non-sym.}}$ approaches $\frac{4}{3}$ when $ms_r \ll \ell s_c$.

C. Computation Cost of StPolyDot Codes

In this part, we detail the cost of computation at the sources and the worker nodes, and the decoding cost at the receiver node. The computational complexity is measured in terms of the number of multiplications and the number of additions required.

a) Computation Cost of the Sources: The sources first compute the linear preencodings $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ for each $i\in\mathcal{I}$, where each has a cost of $s_c\cdot s_r\cdot\frac{\ell}{s_r}\cdot\frac{m}{s_c} = \ell m$. Each source next determines for each $i\in\mathcal{I}$ non-linear transformations $g_1(\tilde{\mathbf{A}}_i)$ and $g_2(\tilde{\mathbf{B}}_i)$ from the linear preencodings, respectively. Each nonlinear mapping $\tilde{\mathbf{A}}_{i2}^\intercal\tilde{\mathbf{A}}_{i1}$ and $\tilde{\mathbf{B}}_{i1}^\intercal\tilde{\mathbf{B}}_{i2}$ (cf. (14) and (19)), arising from the block representations in (6) requires $\frac{m}{s_c}\cdot\frac{\ell}{2s_r}\cdot\frac{m}{s_c} = \frac{\ell m^2}{2s_r s_c^2}$ multiplications and $(\frac{\ell}{2s_r}-1)\frac{m^2}{s_c^2}$ additions. Using the linear structured encoding scheme of Körner-Marton would require incorporating the additional costs of separate encoding at the distributed sources [14]. Note from (7) that for each $i\in\mathcal{I}$, each source multiplies vectors $\{g_1(\tilde{\mathbf{A}}_i(k))\}_{k=1}^n \in \mathbb{F}_q^{n\mu_{\text{StPolyDot}}}$ or $\{g_2(\tilde{\mathbf{B}}_i(k))\}_{k=1}^n \in \mathbb{F}_q^{n\mu_{\text{StPolyDot}}}$ with $\kappa \in \mathbb{F}_q^{n'\mu_{\text{StPolyDot}}\times n\mu_{\text{StPolyDot}}}$ (cf. (8)), which costs $O(n'n)$ for each $i\in\mathcal{I}$ over a block length n. If the same encoding matrix κ is reused across multiple length-n blocks, the arithmetic cost remains $O(n'n)$ per block, while the cost of loading κ can be amortized. Furthermore, if $\mathbf{p}_i^n(x_i)$ is sparse, i.e., it has only $H_q(\mathbf{p}_i^n(x_i))$ nonzero entries with $H_q(\mathbf{p}_i^n(x_i)) \ll n$, then the complexity becomes $O(n'H_q(\mathbf{p}_i^n(x_i)))$ per block of length n. In this work, we focus on the regime in which the cost of computing the encoder outputs is negligible. Hence, for I workers, summing the above costs yields the total computation

cost at the sources

$$\Gamma_{\text{source}}^{\text{StPolyDot}} = 2I\Big(\ell m + \big(\frac{\ell}{s_r} - 1\big)\frac{m^2}{s_c^2}\Big) \ . \tag{39}$$

b) Computation Cost of a Worker: Each worker $i \in \mathcal{I}$ sums the received structured encodings $\kappa(g_1(\tilde{\mathbf{A}}_i^n)) \in \mathbb{F}_q^{n'\mu_{\text{StPolyDot}}}$ and $\kappa(g_2(\tilde{\mathbf{B}}_i^n)) \in \mathbb{F}_q^{n'\mu_{\text{StPolyDot}}}$ in (7) to reconstruct $\mathbf{p}_i^n(x_i)$ (cf. (9)), incurring a cost of $n'\mu_{\text{StPolyDot}}$ over a block of length n. Using the coding scheme in [14] would additionally require joint typicality decoding at the workers, with naive complexity $O(2^{nH_q(\mathbf{p}_i(x_i))})$, i.e., exponential in n. This decoding overhead is inherent to such structured coding schemes and is therefore omitted from our analysis. Finally, each worker post-processes $\mathbf{p}_i(x_i)$ to compute $p_i(x_i)$ and transmits it to the receiver.

In the symmetric case, exploiting (23), the cost of evaluating $p_i(x_i)$ incorporates the cost of multiplying $\big(p_i^{(1)}(x_i)\big)^\intercal$ and $p_i^{(2)}(x_i)$ that is $\big(\frac{\ell}{s_r} - 1\big)\frac{m^2}{s_c^2}$, which is then followed by the addition of $\big(p_i^{(1)}(x_i)\big)^\intercal p_i^{(2)}(x_i) \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ and $p_i^{(3)}(x_i) \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$, where the complexity is $\frac{m^2}{s_c^2}$. Hence, the computation cost of worker $i \in \mathcal{I}$, including the cost of computing $\kappa(g_1(\tilde{\mathbf{A}}_i^n)) \oplus_q \kappa(g_2(\tilde{\mathbf{B}}_i^n))$, is

$$\begin{aligned}\Gamma_{\text{worker}}^{\text{StPolyDot,sym.}} &= \frac{n'\mu_{\text{StPolyDot}}^{\text{sym.}}}{n} + \big(\frac{\ell}{s_r} - 1\big)\frac{m^2}{s_c^2} + \frac{m^2}{s_c^2} \\ &\approx \frac{m}{s_c}\left(\frac{\ell}{s_r} + \frac{m}{s_c}\right) + \frac{\ell m^2}{s_r s_c^2} \ . \end{aligned} \tag{40}$$

In the non-symmetric case, exploiting (26), evaluating $p_i(x_i)$ involves multiplying the terms $\big(p_i^{(2)}(x_i)\big)^\intercal$ and $p_i^{(1)}(x_i)$ and the terms $\big(p_i^{(1)}(x_i)\big)^\intercal$ and $p_i^{(3)}(x_i)$ that cost $\big(\frac{\ell}{s_r} - 1\big)\frac{m^2}{s_c^2}$ each, which is then followed by the addition of $\big(p_i^{(2)}(x_i)\big)^\intercal p_i^{(1)}(x_i) \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$, $\big(p_i^{(1)}(x_i)\big)^\intercal p_i^{(3)}(x_i) \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$, and $p_i^{(4)}(x_i) \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$, where the complexity of these two additions is $\frac{2m^2}{s_c^2}$. Hence, the computation cost of worker $i \in \mathcal{I}$, including the cost of computing $\kappa(g_1(\tilde{\mathbf{A}}_i^n)) \oplus_q \kappa(g_2(\tilde{\mathbf{B}}_i^n))$, is

$$\begin{aligned}\Gamma_{\text{worker}}^{\text{StPolyDot,non-sym.}} &= \frac{n'\mu_{\text{StPolyDot}}^{\text{non-sym.}}}{n} + 2\big(\frac{\ell}{s_r} - 1\big)\frac{m^2}{s_c^2} + \frac{2m^2}{s_c^2} \\ &\approx \frac{m}{s_c}\left(\frac{3\ell}{2s_r} + \frac{m}{s_c}\right) + \frac{2\ell m^2}{s_r s_c^2} \ . \end{aligned} \tag{41}$$

c) Decoding Cost of the Receiver: This part details the complexity analysis for decoding StPolyDot codes at the receiver. The receiver aims to evaluate the computational output $\tilde{p}_i(x_i) = \frac{1}{2}(p_i(x_i) + \big(p_i(x_i)\big)^\intercal) = \tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ given in (25), provided that $\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i = \tilde{\mathbf{B}}_i^\intercal\tilde{\mathbf{A}}_i$ for all $i \in \mathcal{I}$. Decoding of $\mathbf{A}^\intercal\mathbf{B} \in \mathbb{F}_q^{m\times m}$ requires interpolating a subset of products of preencodings (cf. (5)) in the polynomial $\tilde{p}_i(x_i)$ with $\deg(\tilde{p}_i) = s_c^2(2s_r - 1) - 1$ in (27) for all $i \in \mathcal{I}$. To that end, we let

$$\tilde{p}_i(x_i) = \mathbf{D}_0 + \mathbf{D}_1 x_i + \cdots + \mathbf{D}_{\deg(\tilde{p}_i)} x_i^{\deg(\tilde{p}_i)} \tag{42}$$

where $\mathbf{D}_j \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$, $j \in [0, \deg(\tilde{p}_i)]$ are the matrix coefficients given by the linear combinations of $\mathbf{A}_{j,k}^\intercal\mathbf{B}_{j',k'}$, for $j, j' \in [0, s_r - 1]$, and $k, k' \in [0, s_c - 1]$ (cf. (5)). Decoding of $\{\mathbf{D}_j\}_{j\in[0,\deg(\tilde{p}_i)]}$ requires $\deg(\tilde{p}_i)+1$ unique evaluation points of $\tilde{p}_i(x_i)$, thus $I_{r_{\text{StPolyDot}}} = s_c^2(2s_r - 1)$ workers.

We next characterize the decoding cost at the receiver to recover $\mathbf{A}^\intercal\mathbf{B}$ by interpolating $\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ evaluated by workers $i \in \mathcal{I}_{r_{\text{StPolyDot}}}$ (cf. (5)).

Proposition 3. (Decoding complexity of the StPolyDot codes at the receiver.) *The total decoding complexity of the StPolyDot codes at the receiver is given as*

$$\Big(\frac{m}{s_c}\Big)^2 \cdot (\deg(\tilde{p}_i) + 1) + (\deg(\tilde{p}_i) + 1)^3 \ , \tag{43}$$

where $\deg(\tilde{p}_i)+1 = s_c^2(2s_r - 1)$*, and the first term dominates when* $\frac{m}{s_c} \gg s_c^2(2s_r - 1)$.

Proof. The proof follows from using the steps in [6, Section III-B]. See Appendix A. □

In Proposition 3, we neglected the additional cost of computing $\tilde{p}_i(x_i)$ from $p_i(x_i)$ due to (25). The receiver can easily handle this operation via a simple linear operation.

Classical coded computing models focus on offloading computation to workers, treating the master asymmetrically. In contrast, our formulation accounts for non-negligible source-side computation due to distributed compression, making worker-only complexity insufficient. To this end, next, we bound the end-to-end computation costs of StPolyDot codes in the symmetric case, excluding the costs of joint typicality decoding at the workers and interpolation at the receiver.

Proposition 4. (Guarantees in the total computation cost.) *The ratio of the total cost of computation of StPolyDot to the total cost of computation of PolyDot is upper bounded as*

$$\chi_{\text{Comp}} = \frac{\Gamma_{\textit{source}}^{\textit{StPolyDot}} + I\cdot\Gamma_{\textit{worker}}^{\textit{StPolyDot,sym.}}}{\Gamma_{\textit{source}}^{\textit{PolyDot}} + I\cdot\Gamma_{\textit{worker}}^{\textit{PolyDot}}} \le 1 + \frac{1}{s} + \frac{\ell}{2s^2} \ . \tag{44}$$

In the non-symmetric case, χ_{Comp} can be evaluated at $\Gamma_{\text{worker}}^{\text{StPolyDot,non-sym.}}$ (cf. (41)).

Proof. The total computation cost ratio of StPolyDot codes (cf. (39) and (40) in the symmetric case) to PolyDot codes (cf. (32) and (33)) is expressed as

$$\begin{aligned}\chi_{\text{Comp}} &= \frac{2I\Big(\ell m + \big(\frac{\ell}{s_r} - 1\big)\frac{m^2}{s_c^2}\Big) + I\Big(\frac{\ell m}{s_r s_c} + \frac{\ell m^2}{s_r s_c^2} + \frac{m^2}{s_c^2}\Big)}{I\big(2\ell m + \big(\frac{2\ell}{s_r} - 1\big)\frac{m^2}{s_c^2}\big)} \\ &\overset{(a)}{=} 1 + \frac{\frac{\ell}{s} + \frac{\ell m}{s s_c}}{2\ell + \frac{2\ell m}{s s_c} - \frac{m}{s_c^2}} \overset{(b)}{\le} 1 + \frac{1}{s} + \frac{\ell}{2s^2} \end{aligned} \tag{45}$$

where (a) follows from using $s = s_r s_c$, and (b) from upper bounding the numerator in (a) using $\frac{m}{\ell} \le \frac{1}{s}$, lower bounding the denominator in (a) by 2ℓ since the term $\frac{2\ell m}{s s_c} - \frac{m}{s_c^2}$ is nonnegative because $s_r \le \ell$, and using $s_r, s_c \ge 1$. Hence, the upper bound in (44) is obtained. □

D. Communication Cost of StPolyDot Codes

In the proposed framework, the distributed sources do not reveal $\{\tilde{\mathbf{A}}_i\}_{i\in\mathcal{I}}$ and $\{\tilde{\mathbf{B}}_i\}_{i\in\mathcal{I}}$ in their entirety; instead, they

enable compression gains by structured coding of functions of the form $g_1(\tilde{\mathbf{A}}_i^n) \oplus_q g_2(\tilde{\mathbf{B}}_i^n)$ (Section II-.c)) using a common encoding matrix [14] before communicating with the workers. Although worker $i \in \mathcal{I}$ does not observe $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$, it can still recover $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$ (Section II-.d)). This requires less communication than existing polynomial coding schemes, yielding savings for computing $\mathbf{A}^\intercal\mathbf{B}$ at the receiver. Thus, in StPolyDot coding, distributed sources are essential not only for communication but also for encoding. In this part, we detail the complexity of communication from the source nodes to the worker nodes as well as from the workers to the receiver for the proposed StPolyDot codes.

a) Communication cost of the source nodes: We start by analyzing the communication cost from two distributed source nodes. Source one does not transmit $\mathbf{A}$ (or $\mathbf{B}$ for source two); instead, it only sends encodings for the recovery of the polynomials defined in (17).

Using the Körner-Marton scheme for the reconstruction of $\mathbf{p}_i^n(x_i)$ by worker $i \in \mathcal{I}$ where $\mathbf{p}_i(x_i)$ given in (17), the block length needed for each source is given by (8), each source achieves a rate $\frac{n'}{n} \approx H_q(\mathbf{p}_i(x_i))$. Hence, the communication cost for transmitting $\mathbf{p}_i(x_i)$ to worker $i \in \mathcal{I}$ can be made as small as $2H_q(\mathbf{p}_i(x_i))$, which is twice the optimal transmission cost for $\mathbf{p}_i(x_i)$ [14]. Specifically, exploiting the achievability construction in (8), the communication cost from each source node to the set of all workers $\mathcal{I}$ is given by

$$H_q(\{\mathbf{p}_i(x_i)\}_{i\in\mathcal{I}}) \leq I \cdot \mu_{\text{StPolyDot}} \tag{46}$$

where the quantities $\mu_{\text{StPolyDot}}$ and $H_q(\mathbf{p}_i(x_i))$ are determined differently for the symmetric and non-symmetric schemes, and exploiting the dimensions of $p_i^{(1)}(x_i)$, $p_i^{(2)}(x_i)$, and $p_i^{(3)}(x_i)$ given in (17) for the symmetric case, and the polynomials in (21) for the non-symmetric case. The inequality in (46) arises because $\mathbf{p}_i(x_i)$ remain dependent, even if $\mathbf{A}$ and $\mathbf{B}$ are independent.

b) Communication cost of a worker: Upon successfully recovering the polynomials $\mathbf{p}_i(x_i)$ and evaluating $p_i(x_i)$ in (23) when the symmetry condition in (22) holds (or evaluating $\tilde{p}_i(x_i)$ in (25) when the symmetry condition $\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i = \tilde{\mathbf{B}}_i^\intercal\tilde{\mathbf{A}}_i$ holds), worker $i \in \mathcal{I}$ transmits $p_i(x_i) = \tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ to the receiver. Thus, the communication cost of worker $i \in \mathcal{I}$ is

$$H_q(p_i(x_i)) = H_q(\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i) \leq \frac{m^2}{s_c^2} \ , \tag{47}$$

where equality is achieved when the elements of $\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i$ are all i.i.d. and uniform over $\mathbb{F}_q$. Given $I_{r_{\text{StPolyDot}}} = s_c^2(2s_r-1)$, the total communication cost from workers to the receiver is

$$H_q(\{p_i(x_i)\}_{i\in\mathcal{I}_{r_{\text{StPolyDot}}}}) \leq I_{r_{\text{StPolyDot}}}\frac{m^2}{s_c^2} = (2s_r-1)m^2 \ . \tag{48}$$

The communication costs from the source to all workers (cf. (34) for PolyDot and (46) for StPolyDot codes) and from a worker to the receiver (cf. (35) for PolyDot and (47) for StPolyDot codes) capture $P_{\mathbf{A}(k),\mathbf{B}(k)}(\mathbf{A}(k),\mathbf{B}(k))$ at any time $k \in [n]$. The scheme does not require a specific source-correlation model and accommodates general, possibly non-i.i.d., correlations.

Next, we study the gain of StPolyDot (symmetric case) over PolyDot codes in total communication cost, assuming for each $i \in \mathcal{I}$, the entries of $\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i$ are i.i.d. uniform over $\mathbb{F}_q$.

Proposition 5. (Achievable gain in the total communication cost.) *Given a fixed value of $s = s_c$, i.e., $s_r = 1$ and $I_{r_{StPolyDot}} = s_c^2$, and for $\frac{I\ell}{sm} \gg 1$, the total communication cost of StPolyDot approaches half of the total communication cost of PolyDot, i.e.,*

$$\eta_{\text{Comm}} = \frac{H_q(\{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i\}_{i\in\mathcal{I}}) + H_q(\{p_i(x_i)\}_{i\in\mathcal{I}_{r_{PolyDot}}})}{H_q(\{\mathbf{p}_i(x_i)\}_{i\in\mathcal{I}}) + H_q(\{p_i(x_i)\}_{i\in\mathcal{I}_{r_{StPolyDot}}})} \approx 2 \ . \tag{49}$$

Proof. The total communication cost ratio of PolyDot codes to StPolyDot codes is given as

$$\begin{aligned}\eta_{\text{Comm}} &= \frac{I\left(\frac{2\ell m}{s_r s_c}\right) + (2s_r-1)m^2}{I\frac{m}{s_c}\left(\frac{\ell}{s_r}+\frac{m}{s_c}\right) + (2s_r-1)m^2} \\ &\geq \frac{\frac{2I\ell}{sm}+1}{\frac{I\ell}{sm}+\frac{I\ell}{s^3m}+1} \approx 2 \ ,\end{aligned} \tag{50}$$

where the inequality follows from fixing $s = s_r s_c$ and letting $s_r = 1$ (both for PolyDot and StPolyDot codes), and using $m \leq \frac{\ell}{s}$, which implies that $\frac{Im}{s_c^2} = \frac{Im}{s^2} \leq \frac{I\ell}{s^3}$. In (50), the approximation holds when $\frac{I\ell}{s^3m} \ll \frac{I\ell}{sm}$ and $\frac{I\ell}{sm} \gg 1$. □

For the non-symmetric case, the communication cost at each source is upper bounded in a manner analogous to (46), while the communication cost at each worker is obtained as in (47).

In Table II, we contrast the performance of StPolyDot codes (symmetric case) with the prior art, namely MatDot codes (with $s_c = 1$) [32], Polynomial codes (with $s_r = 1$) [5], and PolyDot codes [6]. From (40), the computation cost per worker is less for StPolyDot codes compared to PolyDot codes with a computational complexity $\frac{\ell m^2}{s_r s_c^2}$, provided that $1 < \frac{\ell}{2s_r}$. For this regime, the ratio of the computation cost of StPolyDot codes to that of PolyDot codes is tightly upper bounded for large s (cf. Proposition 4). The total communication costs for PolyDot and StPolyDot can be made equal when $\frac{\ell}{s_r} = \frac{m}{s_c}$. Similarly, the computation cost per worker is less for StMatDot codes compared to MatDot codes with a computational complexity per worker given by $\frac{\ell m^2}{s}$, provided $1 < \frac{\ell}{2s}$. The same cost is less for StPoly codes compared to Polynomial codes with a computational complexity per worker given by $\frac{\ell m^2}{s^2}$, provided $1 < \frac{\ell}{2}$.

In PolyDot coding, the polynomials $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ assigned to each worker $i \in \mathcal{I}$ (cf. (29)) need not be generated at a centralized source. However, using separate encoders for source matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, and transmitting $\{\tilde{\mathbf{A}}_i\}_{i\in\mathcal{I}}$ and $\{\tilde{\mathbf{B}}_i\}_{i\in\mathcal{I}}$, the end-to-end communication cost of the existing polynomial codes may exceed that of StPolyDot codes. To reduce communication costs, existing techniques (e.g., [5] and [6]) can be adapted to the distributed source setting via employing structured coding [14]. This suggests incorporating the stages of preencoding (Section II-.a)) and non-linear transformations (Section II-.b)), followed by the

Cost description	Polynomial codes [5]	MatDot codes [6]	PolyDot codes [6]	StPolyDot codes
Storage parameters	$s_r = 1,\ s = s_c$	$s_c = 1,\ s = s_r$	$s = s_r s_c$	$s = s_r s_c$
Source(s)-Storage	$2\ell m$	$2\ell m$	$2\ell m$	$2\ell m$
Workers-Storage	$2I\frac{\ell m}{s}$	$2I\frac{\ell m}{s}$	$2I\frac{\ell m}{s_r s_c}$	$I\left(\frac{\ell m}{s_r s_c} + \frac{m^2}{s_c^2}\right)$
Source(s)-Comp. (no. of mults. and adds.)	$2I\ell m$	$2I\ell m$	$2I\ell m$	$2I\left(\ell m + \frac{\ell m^2}{s_r s_c^2} - \frac{m^2}{s_c^2}\right)$
Source(s)-Commun. (bits)	$2I\frac{\ell m}{s}$	$2I\frac{\ell m}{s}$	$2I\frac{\ell m}{s_r s_c}$	$I\left(\frac{\ell m}{s_r s_c} + \frac{m^2}{s_c^2}\right)$
Workers-Comp. (no. of mults. and adds.)	$I\frac{m^2}{s^2}(2\ell - 1)$	$Im^2\left(\frac{2\ell}{s} - 1\right)$	$I\left(\frac{m}{s_c}\right)^2\left(\frac{2\ell}{s_r} - 1\right)$	$I\left(\frac{m}{s_c}\left(\frac{\ell}{s_r} + \frac{m}{s_c}\right) + \frac{\ell m^2}{s_r s_c^2}\right)$
Workers-Commun. (bits)	m^2	$(2s-1)m^2$	$(2s_r - 1)m^2$	$(2s_r - 1)m^2$
Receiver-Comp. (no. of mults. and adds.)	$m^2 + s^6$	$(2s-1)m^2$ $+(2s-1)^3$	$(2s_r - 1)m^2$ $+(s_c^2(2s_r - 1))^3$	$(2s_r - 1)m^2$ $+(s_c^2(2s_r - 1))^3$
Recovery threshold (#)	s^2	$2s - 1$	$s_c^2(2s_r - 1)$	$s_c^2(2s_r - 1)$

TABLE II: Cost comparison of StPolyDot with that of [5] and [6] in the symmetric case. The communication costs are upper bounded using the independence bound and assuming all matrix elements are i.i.d. and uniform over $\mathbb{F}_q$.

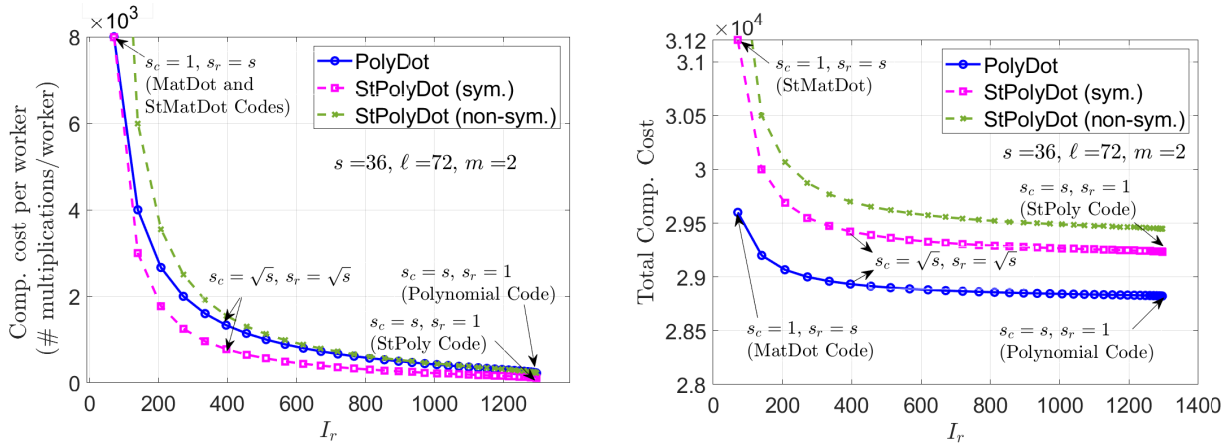

Fig. 2: (Left) Computation cost per worker for computing $\mathbf{A}^\intercal\mathbf{B}$ ($\#$ operations$/I$). (Right) Total computation cost.

structured source coding (Section II-.c)) to reduce the communication cost in the same manner via our approach. However, this would incur computational overhead due to separate encoding at the distributed sources (cf. Section III-C.a)) and joint typicality decoding at the receiver (cf. Section III-C.c)).

IV. Numerical Evaluations

Next, for the proposed source-worker-receiver framework, we provide numerical evaluations to contrast StPolyDot codes with the prior work, namely [5] and [6]. More specifically, we investigate the computation and communication costs per worker, denoted by $\Gamma_{\text{worker}}^{\text{StPolyDot}}$ and $H_q(p_i(x_i))$, respectively, and the total costs of computation and communication, denoted by $\Gamma_{\text{source}}^{\text{StPolyDot}} + I \cdot \Gamma_{\text{worker}}^{\text{StPolyDot}}$ and $H_q(\{\mathbf{p}_i(x_i)\}_{i\in\mathcal{I}}) + H_q(\{p_i(x_i)\}_{i\in\mathcal{I}_{r_{\text{StPolyDot}}}})$, respectively, and how they behave as functions of $I_{r_{\text{StPolyDot}}}$ for different memory parameters s. We further contrast these costs obtained for StPolyDot codes with those of PolyDot codes. The numerical results do not consider the overheads due to Körner-Marton decoding [14].

We first consider $\mathbf{A}$ and $\mathbf{B}$, whose components are all chosen i.i.d. and uniformly from $\mathbb{F}_2$.

a) Computation costs: In Figure 2 (Left), exploiting the computation cost of worker for StPolyDot given in (40) determined as a function of $\{\ell,\ m,\ s_r,\ s_c\}$, we illustrate the computation cost per worker for computing $\mathbf{A}^\intercal\mathbf{B}$, i.e., the total number of operations is normalized by the number of workers I, and contrast with that of PolyDot codes in Table II. With s and ℓ fixed, the costs for both models scale quadratically with m, and their costs relative to I_r will appear identical. In Figure 2 (Right), we demonstrate the total end-to-end computation costs, for PolyDot and StPolyDot codes, excluding the decoding costs, as outlined in (cf. (43)), required to recover $\mathbf{A}^\intercal\mathbf{B}$ by interpolating products of preencodings, namely $\{\tilde{\mathbf{A}}_i^\intercal\tilde{\mathbf{B}}_i\}_{i\in\mathcal{I}}$ (cf. (5)).

b) Communication costs: In Figure 3 (Left), utilizing (47), we numerically evaluate the total communication cost of workers for computing an element of $\mathbf{A}^\intercal\mathbf{B} \in \mathbb{F}_q^{m\times m}$, i.e., the total number of transmitted symbols normalized by m^2. The curves for PolyDot and StPolyDot codes overlap because

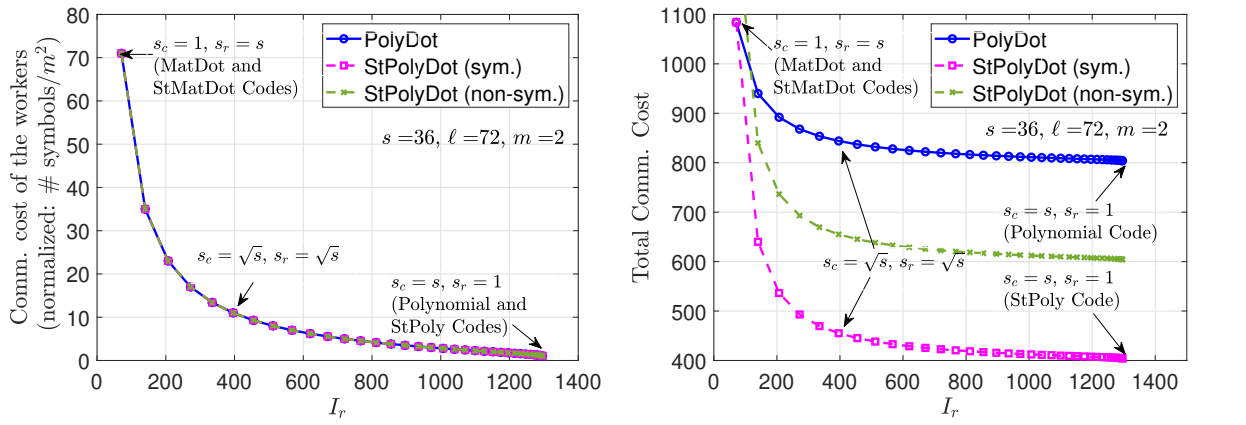

Fig. 3: (Left) Communication cost of workers for computing an element of $\mathbf{A}^\intercal\mathbf{B}$ (norm.: #symbols/m^2). (Right) Total communication cost.

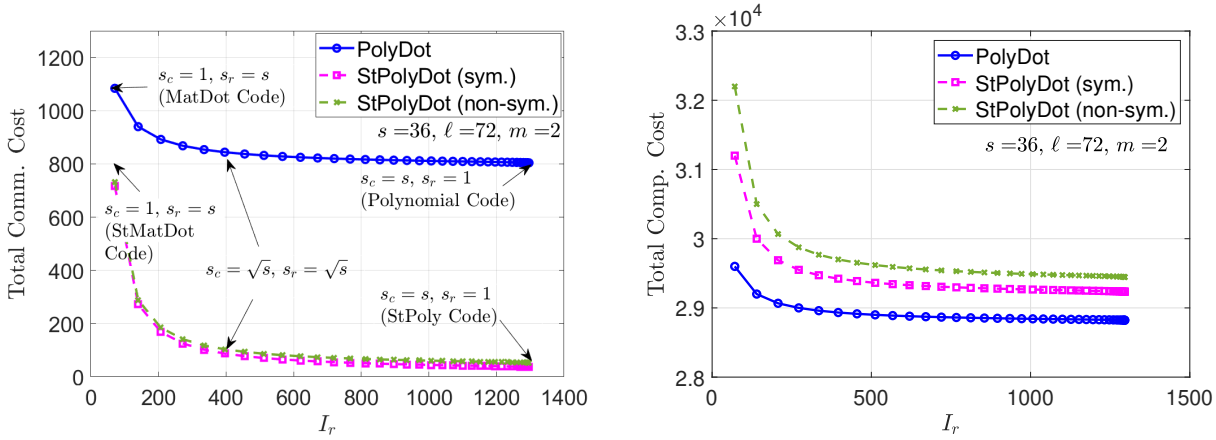

Fig. 4: A high correlation scenario: (Left) Total communication cost. (Right) Total computation cost.

in both approaches worker $i \in \mathcal{I}$ computes and then transmits $\tilde{\mathbf{A}}_i^\intercal \tilde{\mathbf{B}}_i$. On the other hand, the total communication cost of the sources and the workers, given in Figure 3 (Right), showcases that for the same set of parameters, our technique, by leveraging the benefits of *structured source coding*, can significantly reduce the amount of communication from the source nodes to the workers. As s_r approaches 1, as Proposition 5 suggests $\eta_{\text{Comm}} = 2$. When I_r is small, the receiver can recover $\mathbf{A}^\intercal\mathbf{B}$ at a higher communication cost from the sources. In the non-symmetric scenario, the computation costs are modified using (39), (41), and the communication costs (46) and (47) are computed using $\mathbf{p}_i(x_i)$ in (21) and $\mu_{\text{StPolyDot}}^{\text{non-sym.}}$ in (37).

We next demonstrate how communication savings depend on correlation.

c) High/low correlation scenarios: These scenarios are enabled by incorporating additive noise with low/high variance, where each element of $\mathbf{A} \oplus_2 \mathbf{B}$ is considered to be the noise and is distributed according to $\text{Bern}(p)$ [14]. Figures 4 (Left) and (Right) denote the behavior of the total communication and computation costs for $p = 0.01$ (high correlation), respectively. Figures 5 (Left) and (Right) demonstrate how this cost varies with m for $p = 0.01$ and $p = 0.5$ (low correlation scenario with the total communication cost plotted in Figure 3), respectively. Clearly, the savings in the end-to-end communication cost increase with correlation.

d) Shared low-rank components scenarios: Let $\tilde{\mathbf{A}}_i$, $\tilde{\mathbf{B}}_i \in \mathbb{F}_2^{\frac{\ell}{s_r} \times \frac{m}{s_c}}$ with $\ell = 72 s_r$ and $m \in \{2s_c, 30s_c\}$ where $s = 36$. We construct $\tilde{\mathbf{A}}_i = \mathbf{u}_s\mathbf{v}_s \oplus \mathbf{u}_A\mathbf{v}_A$, and $\tilde{\mathbf{B}}_i = \mathbf{u}_s\mathbf{v}_s \oplus \mathbf{u}_B\mathbf{v}_B$ with a shared rank-1 component, where $\mathbf{u}_s, \mathbf{u}_A, \mathbf{u}_B \in \mathbb{F}_2^{\ell \times 1}$ and $\mathbf{v}_s, \mathbf{v}_A, \mathbf{v}_B \in \mathbb{F}_2^{1 \times m}$. The term $\mathbf{u}_s\mathbf{v}_s$ represents the shared rank-1 component, while $\mathbf{u}_A\mathbf{v}_A$ and $\mathbf{u}_B\mathbf{v}_B$ correspond to private rank-1 components of $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$, respectively. In Figures 6 (Left) and (Right), we plot the total communication cost of this model as a function of I_r for $m = 2$ and $m = 30$, respectively. Savings in the end-to-end cost reduce as the size of the parity polynomial $p_i^{(3)}(x_i)$ increases.

From the comparisons, it can be seen that the StPolyDot coding framework can reduce the computation cost per worker, but its total computation cost is higher versus the PolyDot coding framework, including at the sources, caused by the additional overhead at the source nodes caused by the linear preencodings and the structured codings on the non-linear transformations of the preencodings for the receiver to reconstruct $\mathbf{p}_i(x_i)$ (cf. (16) or (20)).

Our numerical results demonstrate that PolyDot codes are more powerful from a *computational complexity perspective* and StPolyDot codes are more efficient from the *standpoint*

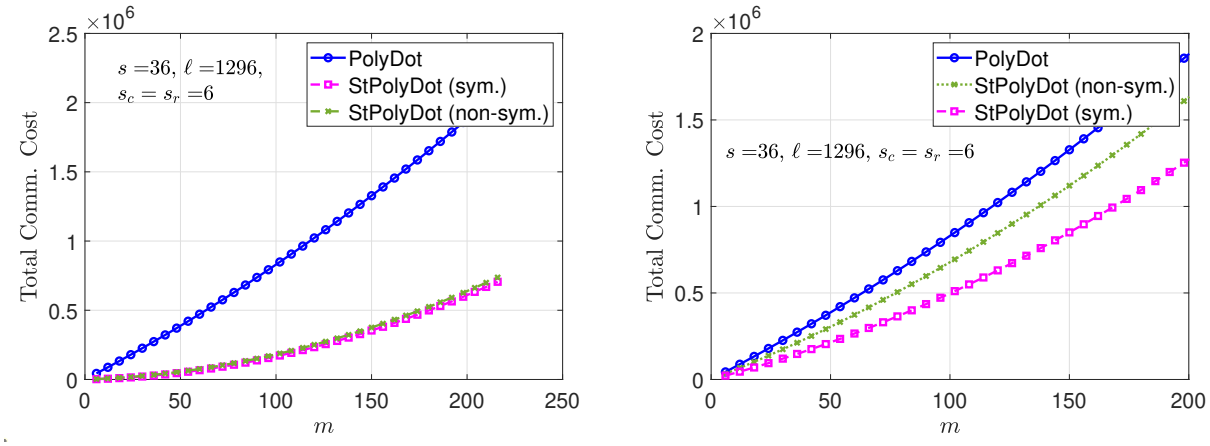

Fig. 5: Total communication cost versus m: (Left) A high correlation scenario. (Right) A low correlation scenario.

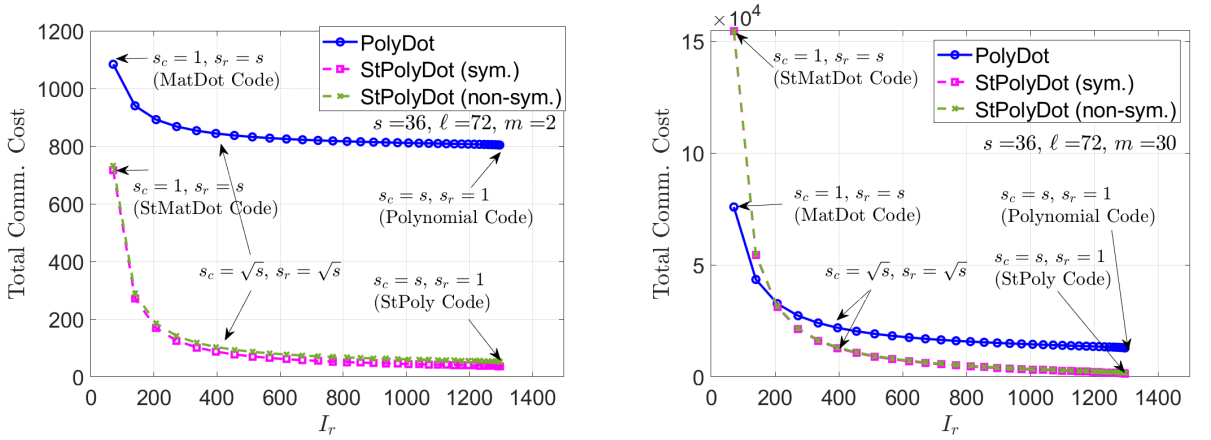

Fig. 6: A high correlation scenario: (Left) Total communication cost versus I_r under shared rank components ($m = 2$). (Right) Total communication cost versus I_r under shared rank components ($m = 30$).

of communication cost because they can capture the benefits of structured source coding. Furthermore, at large s, the computational complexity of StPolyDot codes approaches that of PolyDot codes. The complementary nature of these two schemes allows us to provide a more general design tradeoff space between the two designs (depending on the set of parameters s, ℓ, m).

V. Conclusions

We address the well-known open problem of distributed computation of matrix products. Our key contribution is the application of the structured linear coding scheme of Körner-Marton to a practical source-worker-receiver framework through the design of *new structured polynomial codes, termed StPolyDot codes,* for distributed matrix multiplication. Our design incorporates a small amount of non-linearity, through a non-linear parity polynomial of source matrices, to capture the problem structure. The proposed structured codes reduce communication cost compared with state-of-the-art schemes, even in distributed-source settings. By refining the centralized communication bounds in [5], [6], our approach achieves up to a twofold reduction in large-memory regimes with minimal additional computation overhead. More broadly, StPolyDot codes improve the tradeoff between communication and computation costs and identify operating points at which structured coding schemes outperform unstructured ones. Thus, StPolyDot codes can be combined with existing schemes to expand the rate–memory tradeoff region in polynomial coding, with structured coding reducing rate and unstructured techniques reducing memory usage.

Further research is required to tighten our achievability schemes and extend our coding constructions to general source classes for scalable matrix multiplications. While our results are asymptotically accurate, for exact recovery, zero-error adaptations, as in [74], and for approximate or finite length representations, one-shot models, e.g., [75], [76], could be employed. The proposed scheme is not without limitations. Future work may focus on refining the encoding and decoding procedures to reduce computational cost. Computational costs can be reduced by leveraging the Strassen algorithm [77]. Our future directions include expanding the proposed design to a wider range of non-linear functions, such as general bilinear maps, including tensor products, and chain matrix products. The recovery threshold can be reduced using entangled Polynomial codes [78]. Further research is required to study privacy and security-related aspects. Our design and

analysis tools are also relevant to CDC over real numbers [79] and to DNN and LLM literature, where large-scale matrix multiplication is a key component, and hardware memory is the main bottleneck [3]. Thus, approximate matrix multiplication via quantization becomes the natural solution, and can be achieved using lossy generalizations of structured codes, to allow a receiver to estimate the product, and analyze the approximation error as a function of the quantization rate.

APPENDIX

A. Proof of Proposition 3

To characterize the decoding cost of the StPolyDot codes at the receiver, we assume without loss of generality that the $\deg(\tilde{p}_i)+1$ fastest workers that evaluate the polynomial $\tilde{p}_i(x_i)$ form the set of indices $[1, \deg(\tilde{p}_i)+1]$, where we denote by $x_1, x_2, \ldots, x_{\deg(\tilde{p}_i)+1}$ the $\deg(\tilde{p}_i)+1$ unique values of the evaluation of $\tilde{p}_i(x_i)$ at $i \in [1, \deg(\tilde{p}_i)+1]$, respectively.

Using the Vandermonde matrix given by

$$\mathbf{V} = \begin{bmatrix} 1 & x_1 & \ldots & x_1^{\deg(\tilde{p}_i)} \\ 1 & x_2 & \ldots & x_2^{\deg(\tilde{p}_i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{\deg(\tilde{p}_i)+1} & \ldots & x_{\deg(\tilde{p}_i)+1}^{\deg(\tilde{p}_i)} \end{bmatrix} \in \mathbb{F}_q^{(\deg(\tilde{p}_i)+1)\times(\deg(\tilde{p}_i)+1)} , \tag{51}$$

and we infer that

$$(\mathbf{V} \otimes \mathbf{I}_{\frac{m}{s_c}}) \cdot \mathbf{D}_{\text{vec}} = \tilde{p}_{\text{vec}} \in \mathbb{F}_q^{\frac{m\cdot(\deg(\tilde{p}_i)+1)}{s_c} \times \frac{m}{s_c}} , \tag{52}$$

by letting $\tilde{p}_{\text{vec}} = \left[\tilde{p}_1(x_1); \tilde{p}_2(x_2); \ldots; \tilde{p}_{\deg(\tilde{p}_i)+1}(x_{\deg(\tilde{p}_i)+1})\right]$, and $\mathbf{D}_{\text{vec}} = \left[\mathbf{D}_0; \mathbf{D}_1; \ldots; \mathbf{D}_{\deg(\tilde{p}_i)}\right]$, where $\otimes$ denotes the Kronecker product. The receiver first inverts $\mathbf{V}$ in (52), which has a complexity at most $(s_c^2(2s_r-1))^3$ using a naïve inversion algorithm:

$$\mathbf{D}_{\text{vec}} = (\mathbf{V}^{-1} \otimes \mathbf{I}_{\frac{m}{s_c}}) \cdot \tilde{p}_{\text{vec}} \in \mathbb{F}_q^{\frac{m\cdot(\deg(\tilde{p}_i)+1)}{s_c} \times \frac{m}{s_c}} . \tag{53}$$

The receiver builds $\mathbf{A}^\intercal\mathbf{B}$ exploiting (3). The computation $\sum_{j=0}^{s_r-1} \mathbf{A}_{j,k}^\intercal \mathbf{B}_{j,k'} \in \mathbb{F}_q^{\frac{m}{s_c}\times\frac{m}{s_c}}$ for a given pair $(k,\ k') \in [0, s_c-1]$ can be obtained as a linear combination of the $\deg(\tilde{p}_i)+1$ constitutent matrix coefficients $\mathbf{D}_j$, $j \in [0, \deg(\tilde{p}_i)]$ of $\tilde{p}_i(x_i)$. Note that in (25), $\sum_{j=0}^{s_r-1} \mathbf{A}_{j,k}^\intercal \mathbf{B}_{j,k'}$ is the coefficient of $x_i^{k+s_c(s_r-1)+s_c(2s_r-1)k'}$. Here, for distinct pairs $(k,\ k')$, the exponent of x_i ranges from $s_c(s_r-1)$ to $s_c(s_r+(2s_r-1)(s_c-1))-1$. One can readily see that for every distinct pair (k, k'), the coefficient $\sum_{j=0}^{s_r-1} \mathbf{A}_{j,k}^\intercal \mathbf{B}_{j,k'}$ has a distinct exponent. To that end, the receiver performs the below linear computation to recover the relevant s_c^2 matrices from (53), denoted by $\mathbf{D}_{\text{vec},[\mathrm{s}_c^2]} = \left[\mathbf{D}_0; \mathbf{D}_1; \ldots; \mathbf{D}_{s_c^2-1}\right]$, using the corresponding s_c^2 rows of $\mathbf{V}^{-1}$, denoted as $\mathbf{V}^{-1}_{[s_c^2],:}$

$$\mathbf{D}_{\text{vec},[\mathrm{s}_c^2]} = (\mathbf{V}^{-1}_{[s_c^2],:} \otimes \mathbf{I}_{\frac{m}{s_c}}) \cdot \tilde{p}_{\text{vec}} \in \mathbb{F}_q^{m\cdot s_c \times \frac{m}{s_c}} , \tag{54}$$

to decode the desired product $\mathbf{A}^\intercal\mathbf{B}$. The computational complexity of computing (54) is $\left(\frac{m}{s_c}\right)^2 \cdot (\deg(\tilde{p}_i)+1)$. Thus, the total decoding complexity, including the inversion of $\mathbf{V}$ in (52), is

$$\left(\frac{m}{s_c}\right)^2 \cdot (\deg(\tilde{p}_i)+1) + (\deg(\tilde{p}_i)+1)^3 , \tag{55}$$

where the first term dominates when $\frac{m}{s_c} \gg s_c^2(2s_r-1)$.

REFERENCES

[1] G. Strang, *Introduction to Linear Algebra*. Cambridge Univ. Press, 2023.

[2] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, "Quantum fingerprinting," *Phys. Rev. Lett.*, vol. 87, no. 16, p. 167902, Sep. 2001.

[3] O. Ordentlich and Y. Polyanskiy, "Optimal quantization for matrix multiplication," *IEEE Trans. Inf. Theory*, Dec. 2025.

[4] D. Anastasia and Y. Andreopoulos, "Throughput-distortion computation of generic matrix multiplication: Toward a computation channel for digital signal processing systems," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 2024–37, Nov. 2011.

[5] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc., NeurIPS*, vol. 30, Long Beach, CA, Dec. 2017, pp. 4403–4413.

[6] 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, Jul. 2019.

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc., Int. Conf. on Machine Learning*, Sydney, Australia, Aug. 2017, pp. 3368–3376.

[8] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic MDS codes and expander graphs," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7475–7489, Dec. 2020.

[9] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog Lagrange coded computing," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 283–295, Feb. 2021.

[10] 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 *Proc., AISTATS*, Naha, Okinawa, Japan, Apr. 2019, pp. 1215–1225.

[11] B. Cooperstein, *Advanced linear algebra*. CRC Press, 2010, vol. 2.

[12] J. Shamsi, M. A. Khojaye, and M. A. Qasmi, "Data-intensive cloud computing: Requirements, expectations, challenges, and solutions," *J. Grid Comput.*, vol. 11, no. 2, pp. 281–310, Jun. 2013.

[13] H. Yang, T. Ding, and X. Yuan, "Federated learning with lossy distributed source coding: Analysis and optimization," *IEEE Trans. Commun.*, vol. 71, no. 8, pp. 4561–4576, May 2023.

[14] J. Körner and K. Marton, "How to encode the modulo-two sum of binary sources (corresp.)," *IEEE Trans. Inf. Theory*, vol. 25, no. 2, pp. 219–221, Mar. 1979.

[15] T. S. Han and K. Kobayashi, "A dichotomy of functions F(X, Y) of correlated sources (X, Y)," *IEEE Trans. Inf. Theory*, vol. 33, no. 1, pp. 69–76, Jan. 1987.

[16] B. Nazer and M. Gastpar, "Computation over multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3498–3516, Sep. 2007.

[17] V. Lalitha, N. Prakash, K. Vinodh, P. V. Kumar, and S. S. Pradhan, "Linear coding schemes for the distributed computation of subspaces," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 678–690, Mar. 2013.

[18] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.

[19] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Jan. 2020.

[20] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Trans. Inf. Foren. and Secur.*, vol. 14, no. 1, pp. 141–150, Jun. 2018.

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

[22] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc., IEEE ISIT*, Paris, France, Jul. 2019, pp. 3022–3026.

[23] A. Fidalgo-Díaz and U. Martínez-Peñas, "Distributed matrix multiplication with straggler tolerance using algebraic function fields," *arXiv preprint arXiv:2401.13573*, Jan. 2024.

[24] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2758–2785, Jan. 2021.

[25] A. B. Das, A. Ramamoorthy, D. J. Love, and C. G. Brinton, "Distributed matrix computations with low-weight encodings," *IEEE J. Sel. Areas Inf. Theory*, Aug. 2023.

[26] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-product codes for numerically-stable distributed matrix multiplication," in *Proc., Allerton*, Monticello, IL, Sep. 2019, pp. 253–259.

[27] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *Proc., IEEE Globecom*, Abu Dhabi, UAE, Dec. 2018, pp. 1–6.

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

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

[30] R. G. L. D'Oliveira, S. El Rouayheb, D. Heinlein, and D. Karpuk, "Degree tables for secure distributed matrix multiplication," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 3, pp. 907–918, Aug. 2021.

[31] A. B. Das, A. Ramamoorthy, D. J. Love, and C. G. Brinton, "Preserving sparsity and privacy in straggler-resilient distributed matrix computations," in *Proc., IEEE Allerton*, Monticello, IL, Sep. 2023, pp. 1–8.

[32] H. H. López, G. L. Matthews, and D. Valvo, "Secure MatDot codes: A secure, distributed matrix multiplication scheme," in *Proc., IEEE ITW*, Mumbai, India, Nov. 2022, pp. 149–154.

[33] J. Zhu, Q. Yan, and X. Tang, "Improved constructions for secure multiparty batch matrix multiplication," *IEEE Trans. Commun.*, vol. 69, no. 11, pp. 7673–7690, Aug. 2021.

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

[35] Y. Yang, P. Grover, and S. Kar, "Computing linear transformations with unreliable components," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3729–3756, Apr. 2017.

[36] H. Jeong, T. M. Low, and P. Grover, "Masterless coded computing: A fully-distributed coded FFT algorithm," in *Proc., Allerton*, Monticello, IL, Oct. 2018, pp. 887–894.

[37] 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 *Proc., IEEE ISIT*, Vail, CO, Jun. 2018, pp. 1585–1589.

[38] Y. Yang, P. Grover, and S. Kar, "Can a noisy encoder be used to communicate reliably?" in *Proc., Allerton*, Monticello, IL, Sep. 2014, pp. 659–666.

[39] M. G. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell System Technical Journal*, vol. 47, no. 10, pp. 2299–2337, Dec. 1968.

[40] D. Malak, "Distributed structured matrix multiplication," in *Proc., IEEE ISIT*, Athens, Greece, Jul. 2024.

[41] ——, "Structured codes for distributed matrix multiplication," *to appear, IEEE Trans. Inf. Theory*, Apr. 2026.

[42] H. Yamamoto, "Wyner-Ziv theory for a general function of the correlated sources," *IEEE Trans. Inf. Theory*, vol. 28, no. 5, pp. 803–7, Sep. 1982.

[43] J. Körner, "Coding of an information source having ambiguous alphabet and the entropy of graphs," in *Proc., 6th Prague Conf. Inf. Theory*, Prague, Czech Republic, Sep. 1973, pp. 411–425.

[44] A. Orlitsky and J. R. Roche, "Coding for computing," *IEEE Trans. Inf. Theory*, vol. 47, no. 3, p. 903–917, Mar. 2001.

[45] S. Feizi and M. Médard, "On network functional compression," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5387–5401, Sep. 2014.

[46] M. Sefidgaran and A. Tchamkerten, "On computing a function of correlated sources," *arXiv preprint arXiv:1107.5806*, Jul. 2011.

[47] D. Malak, "Fractional graph coloring for functional compression with side information," in *Proc., IEEE ITW*, Mumbai, India, Nov. 2022.

[48] M. R. D. Salehi and D. Malak, "An achievable low complexity encoding scheme for coloring cyclic graphs," in *Proc., Allerton*, Monticello, IL, Sep. 2023, pp. 1–8.

[49] D. Malak, M. R. Deylam Salehi, B. Serbetci, and P. Elia, "Multi-functional distributed computing," in *Proc., IEEE Allerton*, Urbana-Champaign, IL, 2024, pp. 1–8.

[50] A. Lenz, R. Bitar, A. Wachter-Zeh, and E. Yaakobi, "Function-correcting codes," *IEEE Trans. Inf. Theory*, May 2023.

[51] P. Elias, "Coding for noisy channels," in *IRE WESCON Convention Record*, vol. 2, 1955, pp. 94–104.

[52] R. G. Gallager, *Information Theory and Reliable Communication*. Wiley: New York, Jan. 1968, vol. 588.

[53] C. Nair and Y. N. Wang, "On optimal weighted-sum rates for the modulo sum problem," in *Proc., IEEE ISIT*, Jun. 2020, pp. 2416–2420.

[54] R. Ahlswede and I. Csiszár, "To get a bit of information may be as hard as to get full information," *IEEE Trans. Inf. Theory*, vol. 27, no. 4, pp. 398–408, Jul. 1981.

[55] D. Krithivasan and S. S. Pradhan, "Distributed source coding using abelian group codes: A new achievable rate-distortion region," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1495–1519, Feb. 2011.

[56] S. S. Pradhan, A. Padakandla, and F. Shirani, "An algebraic and probabilistic framework for network information theory," *Found. Trends Commun. Inf. Theory*, vol. 18, no. 2, pp. 173–379, Dec. 2020.

[57] M. A. Sohail, T. A. Atif, and S. S. Pradhan, "Unified approach for computing sum of sources over CQ-MAC," in *Proc., IEEE ISIT*, Espoo, Finland, 2022, pp. 1868–1873.

[58] W. Li, Z. Chen, Z. Wang, S. A. Jafar, and H. Jafarkhani, "Flexible constructions for distributed matrix multiplication," in *Proc., IEEE ISIT*, Virtual Conference, Jul. 2021, pp. 1576–1581.

[59] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Tech.*, vol. 67, no. 12, pp. 12 137–51, Sep. 2018.

[60] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 2, pp. 1492–1525, Dec. 2017.

[61] N. Shivaratri, P. Krueger, and M. Singhal, "Load distributing for locally distributed systems," *Computer*, vol. 25, no. 12, pp. 33–44, Dec. 1992.

[62] K. Wan, H. Sun, M. Ji, and G. Caire, "Distributed linearly separable computation," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 1259–1278, Nov. 2021.

[63] A. Khalesi and P. Elia, "Multi-user linearly-separable distributed computing," *IEEE Trans. Inf. Theory*, vol. 69, no. 10, pp. 6314–39, Jun. 2023.

[64] A. Tanha and D. Malak, "The influence of placement on transmission in distributed computing of boolean functions," in *Proc., IEEE Int. Wksh. Signal Proces. Advances in Wireless Commun.*, Lucca, Italy, Sep. 2024.

[65] M. R. D. Salehi, V. K. K. Purakkal, and D. Malak, "Non-linear function computation broadcast," in *Proc., IEEE ISIT*, Ann Arbor, MI, Jun. 2025.

[66] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded MapReduce," in *Proc., Allerton*, Monticello, IL, Sep. 2015, pp. 964–971.

[67] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," in *Proc., IEEE ISIT*, Istanbul, Türkiye, July 2013, pp. 1077–1081.

[68] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Sep. 2017.

[69] S. Dutta, V. Cadambe, and P. Grover, ""Short-Dot": Computing large linear transforms distributedly using coded short dot products," in *Proc., Adv. Neural Inf. Process. Syst.*, vol. 29, Barcelona, Spain, Dec. 2016.

[70] J. Zhu, S. Li, and J. Li, "Information-theoretically private matrix multiplication from MDS-coded storage," *IEEE Trans. Inf. Forensics and Secur.*, vol. 18, pp. 1680–1695, Feb. 2023.

[71] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[72] D. Malak, "Distributed structured matrix multiplication," *arXiv preprint arXiv:2405.02904*, May 2024.

[73] R. Ahlswede and T. Han, "On source coding with side information via a multiple-access channel and related problems in multi-user information theory," *IEEE Trans. Inf. Theory*, vol. 29, no. 3, pp. 396–412, May 1983.

[74] N. Charpenay, M. Le Treust, and A. Roumy, "Optimal zero-error coding for computing under pairwise shared side information," in *Proc., IEEE ITW*, Saint-Malo, France, Apr. 2023, pp. 97–101.

[75] S. Torabi and J. M. Walsh, "Lossy interactive sum modulo two computation of binary sources," 2016. [Online]. Available: https://faculty.coe.drexel.edu/jwalsh/Torabi_ICASSP16.pdf

[76] ——, "Distributed lossy interactive function computation," in *Proc., Allerton*, Monticello, IL, Sep. 2016, pp. 393–400.

[77] V. Strassen *et al.*, "Gaussian elimination is not optimal," *Numerische mathematik*, vol. 13, no. 4, pp. 354–356, Aug. 1969.

[78] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the" cubic" barrier," in *Proc., IEEE ISIT*, Jun. 2020, pp. 245–250.

[79] P. Moradi, H. Akbarinodehi, and M. A. Maddah-Ali, "General coded computing: Adversarial settings," *arXiv preprint arXiv:2502.08058*, Feb. 2025.