GCSA Codes with Noise Alignment for Secure Coded Multi-Party Batch Matrix Multiplication

Zhen Chen, Zhuqing Jia, Zhiying Wang and Syed A. Jafar
Center for Pervasive Communications and Computing (CPCC)
University of California Irvine
Email: {zhenc4, zhuqingj, zhiying, syed}@uci.edu

***Abstract*—A secure multi-party batch matrix multiplication problem (SMBMM) is considered, where the goal is to allow a master to efficiently compute the pairwise products of two batches of massive matrices, by distributing the computation across S servers. Any X colluding servers gain no information about the input, and the master gains no additional information about the input beyond the product. A solution called Generalized Cross Subspace Alignment codes with Noise Alignment (GCSA-NA) is proposed in this work, based on cross-subspace alignment codes. The state of art solution to SMBMM is a coding scheme called polynomial sharing (PS) that was proposed by Nodehi and Maddah-Ali. GCSA-NA outperforms PS codes in several key aspects — more efficient and secure inter-server communication, lower latency, flexible inter-server network topology, efficient batch processing, and tolerance to stragglers.**

I. Introduction

Recent interest in coding for secure, private, and distributed computing combines a variety of elements such as coded distributed massive matrix multiplication, straggler tolerance, batch computing and private information retrieval [1]–[39]. These related ideas converged recently in Generalized Cross Subspace Alignment (GCSA) codes presented in [39]. GCSA codes originated in the setting of secure private information retrieval [36] and have recently been developed further in [39] for applications to coded distributed batch computation problems where they generalize and improve upon the state of art schemes such as Polynomials codes [2], MatDot codes and PolyDot codes [3], Generalized PolyDot codes [4] and Entangled Polynomial Codes [5] (all based on matrix partitioning) and Lagrange Coded Computing [6] (based on batch processing).

As the next step in the expanding scope of coding for distributed computing, recently in [40] Nodehi and Maddah-Ali explored its application to secure multiparty computation [41]. Specifically, Nodehi et al. consider a system including N sources, S servers and one master. Each source sends a coded function of its data (called a share) to each server. The servers process their inputs and while doing so, may communicate with each other. After that each server sends a message to the master, such that the master can recover the required function of the source inputs. The input data must be kept perfectly secure from the servers even if up to X of the servers collude among themselves. The master must not gain any information about the input data beyond the result. Nodehi et al. propose a scheme called polynomial sharing (PS), which admits basic matrix operations such as addition and multiplication. By concatenating basic operations, arbitrary polynomial function can be calculated. The PS scheme has a few key limitations. It needs multiple rounds of communication among servers where every server needs to send messages to every other server. This carries a high communication cost and requires the network topology among servers to be a complete graph (otherwise data security may be compromised), does not tolerate stragglers, and does not lend itself to batch processing. These aspects (batch processing, improved inter-server communication efficiency, various network topologies) are highlighted as open problems by Nodehi et al. in [40].

Since GCSA codes are particularly efficient at batch processing and already encompass prior approaches to coded distributed computing, in this work we explore whether GCSA codes can also be applied to the problem identified by Nodehi et al. In particular, we focus on the problem of multiplication of two matrices. As it turns out, in this context the answer is in the affirmative. Securing the data against any X colluding servers is already possible with GCSA codes as shown in [39]. The only remaining challenge is how to prevent the master from learning anything about the inputs besides the result of the computation. Let us refer to the additional terms that are contained in the answers sent by the servers to the master, which may collectively reveal information about the inputs beyond the result of the computation, as *interference* terms. To secure these interference terms, we use the idea of Noise Alignment (NA) – the workers communicate among themselves to share noise terms (unknown to the master) that are structured in the same manner as the interfering terms. Because of their matching structures, when added to the answer, the noise terms align perfectly with the interference terms and as a result no information is leaked to the master about the input data besides the result of the computation. Notably, the idea of noise alignment is not novel. While there are superficial distinctions, noise alignment is used essentially in the same manner in [42].

The combination of GCSA codes with noise alignment, GCSA-NA in short, leads to significant advantages over PS schemes. Foremost, because it uses GCSA codes, it allows the benefits of batch processing as well as straggler robustness, neither of which are available in the PS scheme of [40]. The only reason any inter-server communication is needed in a GCSA-NA scheme is to share the aligned noise terms

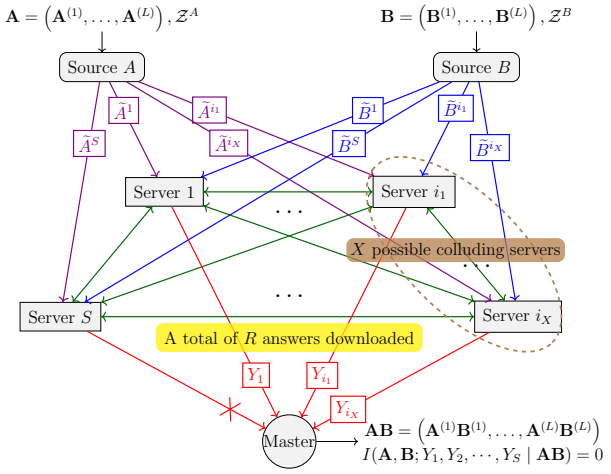


Fig. 1: *The SMBMM problem.*

among the servers. Since these terms do not depend on the data inputs, the inter-server communication in a GCSA-NA scheme is secure in a stronger sense than possible with PS, i.e., even if all inter-server communication is leaked, it can reveal nothing about the data inputs. In fact, the inter-server communication can take place *before* the input data is determined, say during off-peak hours. This directly leads to another advantage. The GCSA-NA scheme allows the inter-server communication network graph to be any connected graph unlike PS schemes which require a complete graph.

Notation: For positive integers M, N, $[N]$ and $[M:N]$ stand for the set $\{1,2,\ldots,N\}$ and $\{M,M+1,\ldots,N\}$, respectively. For $\mathcal{I}=\{i_1,i_2,\ldots,i_N\}$, $X_{\mathcal{I}}$ denotes the set $\{X_{i_1},X_{i_2},\ldots,X_{i_N}\}$. For a matrix $\mathbf{M}$, $|\mathbf{M}|$ denotes the number of elements. For a polynomial P, $deg_\alpha(P)$ denotes its degree with respect to a variable α. The notation $\widetilde{\mathcal{O}}(a\log^2 b)$ suppresses polylog terms, which means that such a term may be replaced with $\mathcal{O}(a\log^2 b)$ if the field $\mathbb{F}$ supports the Fast Fourier Transform, and with $\mathcal{O}(a\log^2 b\log\log(b))$ otherwise.

II. Problem Statement

Consider a system including 2 sources, S servers (workers) and one master, as illustrated in Fig. 1. Each source is connected to every single server. Servers are connected to each other, and all of the servers are connected to the master. All of these links are secure and error free.

Each source generates a sequence of L matrices, denoted as $\mathbf{A}=\left(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(L)}\right)$, $\mathbf{B}=\left(\mathbf{B}^{(1)},\ldots,\mathbf{B}^{(L)}\right)$, such that for all $l\in[L]$, $\mathbf{A}^{(l)}\in\mathbb{F}^{\lambda\times\kappa},\mathbf{B}^{(l)}\in\mathbb{F}^{\kappa\times\mu}$. The master is interested in the sequence of product matrices, $\mathbf{AB}=\left(\mathbf{A}^{(1)}\mathbf{B}^{(1)},\ldots,\mathbf{A}^{(L)}\mathbf{B}^{(L)}\right)$. The system operates in three phases: sharing, computation and communication, and reconstruction.

1) Sharing: Each source encodes (encrypts) its matrices for the s^{th} server as $\widetilde{A}^s$ and $\widetilde{B}^s$, so $\widetilde{A}^s=f_s(\mathbf{A},\mathcal{Z}^A),\widetilde{B}^s=g_s(\mathbf{B},\mathcal{Z}^B)$, where $\mathcal{Z}^A$ and $\mathcal{Z}^B$ represent private randomness (noise) generated by the source. The encoded matrices, $\widetilde{A}^s,\widetilde{B}^s$, are sent to the s^{th} server.

2) Computation and Communication: Denote the communication from Server s to Server s' as $M_{s\to s'}$. Define $\mathcal{M}_s=\{M_{s'\to s},s'\in[S]\setminus\{s\}\}$ and $\mathcal{M}=\{\mathcal{M}_s,s\in[S]\}$. After the communication among servers, each server s computes a response Y_s and sends it to the master. Y_s is a function of $\widetilde{A}^s$, $\widetilde{B}^s$ and $\mathcal{M}_s$, i.e., $Y_s=h_s(\widetilde{A}^s,\widetilde{B}^s,\mathcal{M}_s)$, where $h_s,s\in[S]$ are the functions used to produce the answer, and we denote them collectively as $\boldsymbol{h}=(h_1,h_2,\ldots,h_S)$.

3) Reconstruction: The master downloads information from servers. Some servers may fail to respond (or respond after the master executes the reconstruction), such servers are called stragglers. The master decodes the sequence of product matrices $\mathbf{AB}$ based on the information from the responsive servers, using a class of decoding functions (denoted $\boldsymbol{d}$). Define $\boldsymbol{d}=\{d_{\mathcal{R}}:\mathcal{R}\subset[S]\}$ where $d_{\mathcal{R}}$ is the decoding function used when the set of responsive servers is $\mathcal{R}$.

This scheme must satisfy three constraints.

Correctness: The master must be able to recover the desired products $\mathbf{AB}$, i.e., $H(\mathbf{AB}\mid Y_{\mathcal{R}})=0$, or equivalently $\mathbf{AB}=d_{\mathcal{R}}(Y_{\mathcal{R}})$, for some $\mathcal{R}$.

Security & Strong Security: The servers must remain oblivious to $\mathbf{A},\mathbf{B}$, even if X of them collude. Formally, $\forall\mathcal{X}\subset[S],|\mathcal{X}|\leq X$, $I(\mathbf{A},\mathbf{B};\widetilde{A}^{\mathcal{X}},\widetilde{B}^{\mathcal{X}},\mathcal{M}_{\mathcal{X}})=0$.

In this paper, *strong security* is also considered. It requires that the information transmitted among servers is independent of $\mathbf{A},\mathbf{B}$ and all of the shares $\widetilde{A}^{[S]},\widetilde{B}^{[S]}$, i.e., $I(\mathbf{A},\mathbf{B},\widetilde{A}^{[S]},\widetilde{B}^{[S]};\mathcal{M})=0$.

Privacy: The master must not gain any additional information about $\mathbf{A},\mathbf{B}$, beyond the required product. Precisely, $I(\mathbf{A},\mathbf{B};Y_1,Y_2,\cdots,Y_S\mid\mathbf{AB})=0$.

We say that $(\boldsymbol{f},\boldsymbol{g},\boldsymbol{h},\boldsymbol{d})$ form an SMBMM code if it satisfies these three constraints. An SMBMM code is said to be r-recoverable if the master is able to recover the desired products from the answers obtained from any r servers. In particular, an SMBMM code $(\boldsymbol{f},\boldsymbol{g},\boldsymbol{h},\boldsymbol{d})$ is r-recoverable if for any $\mathcal{R}\subset[S]$, $|\mathcal{R}|=r$, and for any realization of $\mathbf{A}$, $\mathbf{B}$, we have $\mathbf{AB}=d_{\mathcal{R}}(Y_{\mathcal{R}})$. Define the recovery threshold R of an SMBMM code $(\boldsymbol{f},\boldsymbol{g},\boldsymbol{h},\boldsymbol{d})$ to be the minimum integer r such that the SMBMM code is r-recoverable.

The communication cost is comprised of 3 parts: source upload cost, server communication cost, and master download cost. The (normalized) upload costs $U_A=\frac{\sum_{s\in[S]}|\widetilde{A}^s|}{L\lambda\kappa}$ and $U_B=\frac{\sum_{s\in[S]}|\widetilde{B}^s|}{L\kappa\mu}$. Similarly, the (normalized) server communication cost $CC=\frac{|\mathcal{M}|}{L\lambda\mu}$ and download cost $D=\max_{\mathcal{R},\mathcal{R}\subset[S],|\mathcal{R}|=R}\frac{\sum_{s\in\mathcal{R}}|Y_s|}{L\lambda\mu}$.

Next let us consider the complexity of encoding, decoding and server computation. Define the (normalized) computational complexity at each server, $\mathcal{C}_s$, to be the order of the number of arithmetic operations required to compute the function h_s at each server, normalized by L. Similarly, define the (normalized) encoding computational complexity $\mathcal{C}_{eA}$ for $\widetilde{A}^{[S]}$ and $\mathcal{C}_{eB}$ for $\widetilde{B}^{[S]}$ as the order of the number of arithmetic operations required to compute the functions $\boldsymbol{f}$ and $\boldsymbol{g}$, respectively, each normalized by L. Finally, define the (normalized) decoding computational complexity $\mathcal{C}_d$ to be

	Polynomial Sharing (PS [40])	GCSA-NA
Strong Security	No	Yes
Recovery Threshold (R)	$2pmn + 2X - 1$	$pmn(\ell+1)K_c + 2X - 1$
Straggler Tolerance	No $(S = R)$	Yes. Tolerates $S - R$ stragglers
Server Network Topology	Complete Graph	Any Connected Graph
Source Encoding Complexity $(\mathcal{C}_{eA}, \mathcal{C}_{eB})$	$\left(\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{pm}\right), \widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{pn}\right)\right)$	$\left(\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_c pm}\right), \widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_c pn}\right)\right)$
Source Upload Cost (U_A, U_B)	$\left(\frac{S}{pm}, \frac{S}{pn}\right)$	$\left(\frac{S}{K_c pm}, \frac{S}{K_c pn}\right)$
Server Communication Cost (CC)	$\frac{S(S-1)}{mn}$	$\frac{S-1}{\ell K_c mn}$
Server Computation Complexity $(\mathcal{C}_s)$	$\mathcal{O}\left(\frac{\lambda\kappa\mu}{pmn}\right) + \mathcal{O}(\lambda\mu) + \widetilde{\mathcal{O}}\left(\frac{S\log^2 S\lambda\mu}{mn}\right)$ $+\mathcal{O}\left(\frac{(S-1)\lambda\mu}{mn}\right) \approx \mathcal{O}\left(\frac{\lambda\kappa\mu}{pmn}\right)$ if $\frac{\kappa}{p} \gg S$	$\mathcal{O}\left(\frac{\lambda\kappa\mu}{K_c pmn}\right) + \mathcal{O}\left(\frac{\lambda\mu}{K_c mn}\right) + \widetilde{\mathcal{O}}\left(\frac{\lambda\mu\log^2 S}{\ell K_c mn}\right)$ $\approx \mathcal{O}\left(\frac{\lambda\kappa\mu}{K_c pmn}\right)$ if $\frac{\kappa}{p} \gg S$
Master Download Cost (D)	$\frac{mn+X}{mn}$	$\frac{R}{\ell K_c mn}$
Master Decoding Complexity $(\mathcal{C}_d)$	$\widetilde{\mathcal{O}}\left(\lambda\mu\log^2(mn+X)\right)$	$\widetilde{\mathcal{O}}\left(\lambda\mu p\log^2(R)\right)$

TABLE I: *Performance comparison of Polynomial Sharing (PS) and* GCSA with Noise Alignment *(GCSA-NA).*

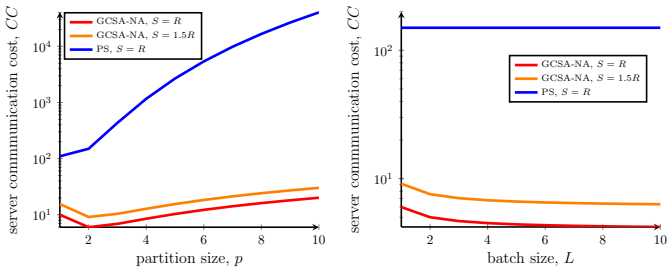
the order of the number of arithmetic operations required to compute $d_{\mathcal{R}}(Y_{\mathcal{R}})$, maximized over $\mathcal{R}, \mathcal{R} \subset [S], |\mathcal{R}| = R$, and normalized by L.

III. Main Result

Our main result appears in the following theorem.

Theorem 1. *For SMBMM over a field $\mathbb{F}$ with S servers, X-security, and positive integers (ℓ, K_c, p, m, n) such that $m \mid \lambda$, $p \mid \kappa$, $n \mid \mu$ and $L = \ell K_c \leq |\mathbb{F}| - S$, the GCSA-NA scheme presented in Section IV is a solution, and its recovery threshold, cost, and complexity are listed in Table I.*

A side-by-side comparison of the GCSA-NA solution with polynomial sharing (PS) appears in Table I. GCSA-NA schemes are strongly secure, i.e., even if all inter-server communication is leaked it does not compromise the security of input data. In GCSA-NA the inter-server network graph can be any connected graph. This is not possible with PS. For example, if the inter-server network graph is a star graph, then the hub server can decode $\mathbf{AB}$ by monitoring all the inter-server communication in a PS scheme, violating the security constraint. Unlike the PS scheme, in GCSA-NA, all inter-server communication can take place during off-peak hours, even before the input data is generated, giving GCSA-NA a significant latency advantage. Unlike PS where every server must communicate with every server, i.e., $S(S-1)$ such inter-server communications must take place, GCSA-NA only requires $S-1$ inter-server communications to propagate structured noise terms across all servers. This improvement is shown numerically in Fig. 2a. The server computation complexity is also lower for the GCSA-NA scheme than the PS scheme. This is because in PS, each server needs to multiply the two shares received from the sources, calculate the shares for *every other server* and sum up all the shares from *every other server*. However, in GCSA-NA, each server only needs to multiply the two shares received from the sources and add noise (which can be precomputed during off-peak hours). Note that when restricted to batch size 1, i.e., with $\ell = K_c = 1$, GCSA-NA has the same recovery threshold as PS. The GCSA-NA scheme naturally allows robustness to stragglers, which is particularly important for massive matrix multiplications. Now consider batch processing, i.e., batch size $L > 1$, e.g., with $L = K_c, \ell = 1$. PS can be applied to batch processing by repeating the scheme L times. Fig. 2b shows that the normalized server communication cost of GCSA-NA decreases as L increases and is significantly less than that in PS. For the same number of servers S, the upload cost of GCSA-NA is smaller by a factor of $1/K_c$ compared to PS. GCSA-NA does have higher download cost and decoding complexity than PS by approximately a factor of p, which depends on how the matrices are partitioned. If p is a small value, e.g., $p = 1$, then the costs are quite similar. The improvement in download cost and decoding complexity of PS by a factor of $1/p$ comes at the penalty of increased inter-server communication cost by a factor of S. But since $S \geq R \geq 2pmn + 2X - 1 \geq p$, and typically $S \gg p$, the improvement is dominated by the penalty, so that overall the communication cost of PS is still significantly higher.



(a) (b)

Fig. 2: $\lambda = \kappa = \mu, p = m = n$. (a) Server communication cost vs. partition size, $L = 1$ and $X = 5$. (b) Server communication cost vs. batch size, $p = 2$ and $X = 5$.

IV. GCSA-NA Codes

A. Toy Example

Let us consider a toy example with parameters $\lambda = \kappa = \mu, m = n = 1, p = 2, l = 1, K_c = 2, X = 1$ and $S = R$. Suppose matrices $\mathbf{A}, \mathbf{B} \in \mathbb{F}^{\lambda\times\lambda}$, and we wish to

multiply matrix $\mathbf{A} = [\mathbf{A}_1 \ \mathbf{A}_2]$ with matrix $\mathbf{B} = \left[\mathbf{B}_1^T \ \mathbf{B}_2^T\right]^T$ to compute the product $\mathbf{AB} = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2$, where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{F}^{\lambda \times \frac{\lambda}{2}}, \mathbf{B}_1, \mathbf{B}_2 \in \mathbb{F}^{\frac{\lambda}{2} \times \lambda}$. For this toy example we summarize both the Polynomial Sharing approach [40], [43], [44], and our GCSA-NA approach.

1) Polynomial Sharing Solution: Polynomial sharing is based on EP code [5]. The given partitioning corresponds to EP code construction for $m = n = 1, p = 2$, and we have

$$P = \mathbf{A}_1 + \alpha\mathbf{A}_2, \qquad Q = \alpha\mathbf{B}_1 + \mathbf{B}_2.$$

To satisfy $X = 1$ security, PS includes noise with each share, i.e., $\widetilde{A} = P + \alpha^2\mathbf{Z}^A$, $\widetilde{B} = Q + \alpha^2\mathbf{Z}^B$, where $\alpha_1, \cdots, \alpha_S$ are distinct elements, and $\alpha, \widetilde{A}, \widetilde{B}$ are generic variables that should be replaced with $\alpha_s, \widetilde{A}^s, \widetilde{B}^s$ for Server s. Each server computes the product of the shares that it receives,

$$\widetilde{A}\widetilde{B} = \mathbf{A}_1\mathbf{B}_2 + \alpha(\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2) + \alpha^2(\mathbf{A}_2\mathbf{B}_1 + \mathbf{A}_1\mathbf{Z}^B + \mathbf{Z}^A\mathbf{B}_2) + \alpha^3(\mathbf{A}_2\mathbf{Z}^B + \mathbf{Z}^A\mathbf{B}_1) + \alpha^4\mathbf{Z}^A\mathbf{Z}^B.$$

To secure inputs from the master, PS requires that every server sends to the master only the desired term $\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2$ by using secret sharing scheme among servers. Since $deg_\alpha(\widetilde{A}\widetilde{B}) = 4$, $\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2$ can be calculated from 5 distinct $\widetilde{A}\widetilde{B}$ according to the Lagrange interpolation rules. In particular, there exist 5 constants $r_1, \cdots, r_5$, such that $\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 = \sum_{s\in[5]} r_s\widetilde{A}^s\widetilde{B}^s$. Consider Server s, it sends $M_{s\to j} = r_s\widetilde{A}^s\widetilde{B}^s + \alpha_j\mathbf{Z}_s$ to server j, where $\mathbf{Z}_1, \cdots, \mathbf{Z}_5$ are i.i.d. uniform noise matrices. After Server s collects all the shares $M_{j\to s}$, it sums them up $Y_s = \sum_{j\in[5]} M_{j\to s} = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 + \alpha_s\sum_{j\in[5]}\mathbf{Z}_j$, and sends Y_s to the master. Note that after receiving $M_{j\to s}$ for all $j \in [5]$, Server s still gains no information about the input data, which guarantees the security. However, it does not satisfy strong security, because $\mathbf{AB}$ can be decoded based on $M_{j\to s}$ for all $j, s \in [5]$.

The master can decode the desired $\mathbf{AB}$ after collecting 2 responses from servers.[1] Note that PS needs at least $S = R = 5$ servers, since 5 distinct $\widetilde{A}\widetilde{B}$ are required to obtain Y_s.

2) GCSA-NA Solution: GCSA codes [39] can handle batch processing, therefore let us consider batch size 2 ($\ell = 1, K_c = 2$). Denote the second instance by $\mathbf{A}', \mathbf{B}'$. Using CSA code,

$$P = \mathbf{A}_1 + (f-\alpha)\mathbf{A}_2, \qquad Q = (f-\alpha)\mathbf{B}_1 + \mathbf{B}_2,$$
$$P' = \mathbf{A}'_1 + (f'-\alpha)\mathbf{A}'_2, \qquad Q' = (f'-\alpha)\mathbf{B}'_1 + \mathbf{B}'_2,$$

and the shares are constructed as follows,

$$\widetilde{A} = \Delta\left(\frac{P}{(f-\alpha)^2} + \frac{P'}{(f'-\alpha)^2}\right), \widetilde{B} = \frac{Q}{(f-\alpha)^2} + \frac{Q'}{(f'-\alpha)^2},$$

where $\Delta = (f-\alpha)^2(f'-\alpha)^2$. $f, f', \alpha_1, \cdots, \alpha_S$ are distinct elements, and $\alpha, \widetilde{A}, \widetilde{B}$ are generic variables that should be replaced with $\alpha_s, \widetilde{A}^s, \widetilde{B}^s$ for Server s. Each server computes the product of the shares that it receives, i.e.,

$$\widetilde{A}\widetilde{B} = \frac{c_0PQ}{(f-\alpha)^2} + \frac{c_1PQ}{f-\alpha} + \frac{c'_0P'Q'}{(f'-\alpha)^2} + \frac{c'_1P'Q'}{f'-\alpha} + I_0 + \alpha I_1 + \alpha^2 I_2$$
$$= \frac{c_0\mathbf{A}_1\mathbf{B}_2}{(f-\alpha)^2} + \frac{c_0\mathbf{A}_1\mathbf{B}_1 + c_0\mathbf{A}_2\mathbf{B}_2 + c_1\mathbf{A}_1\mathbf{B}_2}{f-\alpha} + \frac{c'_0\mathbf{A}'_1\mathbf{B}'_2}{(f'-\alpha)^2} + \frac{c'_0\mathbf{A}'_1\mathbf{B}'_1 + c'_0\mathbf{A}'_2\mathbf{B}'_2 + c'_1\mathbf{A}'_1\mathbf{B}'_2}{f'-\alpha} + I_0 + \alpha I_1 + \alpha^2 I_2,$$

where I_0, I_1, I_2 are combinations of $PQ, P'Q', PQ', P'Q$ and c_0, c_1, c'_0, c'_1 are constants. This is the original GCSA code [39], and we need 7 responses to recover the desired product.

Next, let us modify the scheme to make it $X = 1$ secure by including noise with each share, i.e.,

$$\widetilde{A} = \Delta\left(\frac{P}{(f-\alpha)^2} + \frac{P'}{(f'-\alpha)^2} + \mathbf{Z}^A\right),$$
$$\widetilde{B} = \frac{Q}{(f-\alpha)^2} + \frac{Q'}{(f'-\alpha)^2} + \mathbf{Z}^B,$$
$$\widetilde{A}\widetilde{B} = \frac{c_0PQ}{(f-\alpha)^2} + \frac{c_1PQ}{f-\alpha} + \frac{c'_0P'Q'}{(f'-\alpha)^2} + \frac{c'_1P'Q'}{f'-\alpha} + \sum_{i=0}^{4}\alpha^i I_i.$$

Note that as a result of the added noise terms, the recovery threshold is now increased to 9. Also note that the term I_4 contains only contributions from $\Delta\mathbf{Z}^A\mathbf{Z}^B$, i.e., this term leaks no information about $\mathbf{A}, \mathbf{B}$ matrices.

If the servers directly return their computed values of $\widetilde{A}\widetilde{B}$ to the master, then besides the result of the computation some additional information about the input matrices $\mathbf{A}, \mathbf{B}$ may be leaked by the terms

$$\left(\frac{c_0}{(f-\alpha)^2} + \frac{c_1}{f-\alpha}\right)\mathbf{A}_1\mathbf{B}_2 + \left(\frac{c'_0}{(f'-\alpha)^2} + \frac{c'_1}{f'-\alpha}\right)\mathbf{A}'_1\mathbf{B}'_2 + \sum_{i=0}^{3}\alpha^i I_i,$$

which can be secured by the addition of *aligned noise* terms

$$\widetilde{Z} = \left(\frac{c_0}{(f-\alpha)^2} + \frac{c_1}{f-\alpha}\right)\mathbf{Z} + \left(\frac{c'_0}{(f'-\alpha)^2} + \frac{c'_1}{f'-\alpha}\right)\mathbf{Z}' + \sum_{i=0}^{3}\alpha^i\mathbf{Z}_i$$

at each server so that the answer returned by each server to the master is $\widetilde{A}\widetilde{B} + \widetilde{Z}$. Here $\mathbf{Z}, \mathbf{Z}', \mathbf{Z}_0, \mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3$ are i.i.d. uniform noise matrices, that can all be privately generated by one server, who can then share their aligned form $\widetilde{Z}$ with all other servers. This sharing of $\widetilde{Z}$ is the only inter-server communication needed in the GCSA-NA scheme.

B. General Construction of GCSA-NA

1) Partitioning: $L = \ell K_c$ instances of $\mathbf{A}$ and $\mathbf{B}$ matrices are split into ℓ groups. $\forall l \in [\ell], \forall k \in [K_c]$, denote

$$\mathbf{A}^{l,k} = \mathbf{A}^{(K_c(l-1)+k)}, \ \mathbf{B}^{l,k} = \mathbf{B}^{(K_c(l-1)+k)}.$$

Further, each $\mathbf{A}^{l,k}$ is partitioned into $m \times p$ blocks and each matrix $\mathbf{B}^{l,k}$ is partitioned into $p \times n$ blocks, where $\mathbf{A}^{l,k}_{i,j} \in \mathbb{F}^{\frac{\lambda}{m}\times\frac{\kappa}{p}}, \mathbf{B}^{l,k}_{j,k} \in \mathbb{F}^{\frac{\kappa}{p}\times\frac{\mu}{n}}, i \in [m], j \in [p], k \in [n]$.

Let $f_{1,1}, f_{1,2}, \cdots, f_{\ell,K_c}, \alpha_1, \alpha_2, \cdots, \alpha_S$ be $(S+L)$ distinct elements from the field $\mathbb{F}$. For convenience, define

$$D_E = \max(pm, pmn - pm + p) - 1,$$
$$\Delta^{l,K_c}_s = \prod_{k\in[K_c]}(f_{l,k} - \alpha_s)^{pmn}, \forall l \in [\ell], \forall s \in [S],$$
$$\mathcal{E} = \{p + p(m'-1) + pm(n''-1) \mid m' \in [m], n'' \in [n]\}.$$

$\forall l \in [\ell], \forall k \in [K_c]$, define $c_{l,k,i}, i \in \{0, 1, \cdots, pmn(K_c-1)\}$ to be the coefficients satisfying

$$\Psi_{l,k}(\alpha) = \prod_{k'\in[K_c]\setminus\{k\}}(\alpha + f_{l,k'} - f_{l,k})^{pmn} = \sum_{i=0}^{pmn(K_c-1)} c_{l,k,i}\alpha^i, \tag{1}$$

[1] In [44], for arbitrary polynomials, $M_{s\to j} = r_s\widetilde{A}^s\widetilde{B}^s + \alpha_j^2\mathbf{Z}_s$ because Y_s is forced to be casted in the form of entangled polynomial sharing.

i.e., they are the coefficients of the polynomial $\Psi_{l,k}(\alpha)$, which is defined by its roots. Note that all the coefficients $\alpha_s, f_{l,k}, c_{l,k,i}$ are globally known.

2) Sharing: Firstly, each source encodes each $\mathbf{A}^{l,k}$ and $\mathbf{B}^{l,k}$ with Entangled Polynomial code. For all $l \in [\ell], k \in [K_c]$,

$$P_s^{l,k} = \sum_{m' \in [m]} \sum_{p' \in [p]} \mathbf{A}_{m',p'}^{l,k} (f_{l,k} - \alpha_s)^{p'-1+p(m'-1)},$$
$$Q_s^{l,k} = \sum_{p'' \in [p]} \sum_{n'' \in [n]} \mathbf{B}_{p'',n''}^{l,k} (f_{l,k} - \alpha_s)^{p-p''+pm(n''-1)}.$$

Each source generates ℓX independent random matrices, i.e., $\mathcal{Z}^I = \left\{\mathbf{Z}_{1,1}^I, \cdots, \mathbf{Z}_{1,X}^I, \mathbf{Z}_{2,1}^I, \cdots, \mathbf{Z}_{\ell,X}^I\right\}, I \in \{A, B\}$. For all $s \in [S]$, the shares of matrices $\mathbf{A}$ and $\mathbf{B}$ at the s^{th} server are constructed as follows.

$$\widetilde{A}^s = (\widetilde{A}_1^s, \widetilde{A}_2^s, \ldots, \widetilde{A}_\ell^s),\ \widetilde{B}^s = (\widetilde{B}_1^s, \widetilde{B}_2^s, \ldots, \widetilde{B}_\ell^s),$$
$$\widetilde{A}_l^s = \Delta_s^{l,K_c} \left(\sum_{k \in [K_c]} \frac{P_s^{l,k}}{(f_{l,k} - \alpha_s)^{pmn}} + \sum_{x \in [X]} \alpha_s^{x-1} \mathbf{Z}_{l,x}^A \right),$$
$$\widetilde{B}_l^s = \sum_{k \in [K_c]} \frac{Q_s^{l,k}}{(f_{l,k} - \alpha_s)^{pmn}} + \sum_{x \in [X]} \alpha_s^{x-1} \mathbf{Z}_{l,x}^B.$$

for all $l \in [\ell]$. Then $\widetilde{A}^s, \widetilde{B}^s$ is sent to Server s, $\forall s \in [S]$.

3) Computation and Communication: One of the servers generates a set of $\frac{\lambda}{m} \times \frac{\mu}{n}$ independent matrices, denoted as $\mathcal{Z}^{server}$, which contains $pmn(K_c - 1) + X + D_E + \ell K_c(p-1)mn$ random matrices and $\ell K_c mn$ zero matrices. In particular, $\mathcal{Z}^{server} = \{\mathcal{Z}_1^{server}, \mathcal{Z}_2^{server}\}$, where $\mathcal{Z}_1^{server} = \{\mathbf{Z}_i' \mid i \in [pmn(K_c - 1) + X + D_E]\}$, and $\mathcal{Z}_2^{server} = \left\{\mathbf{Z}_{l,k,i}'' \mid l \in [\ell], k \in [K_c], i \in [pmn]\right\}$. Here,

$$\mathbf{Z}_{l,k,i}'' = \begin{cases} \mathbf{0}, & \text{if } i \in \mathcal{E} \\ \mathbf{Z}_{l,k,i}''', & \text{otherwise,} \end{cases} \quad \forall l \in [\ell], \forall k \in [K_c].$$

Without loss of generality, assume the 1^{st} server generates $\mathcal{Z}^{server}$, encodes them into

$$\widetilde{M}_s = \sum_{t \in [pmn(K_c-1)+X+D_E]} \alpha_s^{t-1} \mathbf{Z}_t' + \sum_{l \in [\ell]} \sum_{k \in [K_c]} \sum_{i=0}^{pmn-1} \frac{\sum_{i'=0}^{i} c_{l,k.i-i'} \mathbf{Z}_{l,k,i'+1}''}{(f_{l,k} - \alpha_s)^{pmn-i}},$$

and sends $\widetilde{M}_s$ to server $s, s \in [S] \setminus \{1\}$, where $c_{l,k.i}$ is defined in (1). The answer returned by the s^{th} server to the master is constructed as $Y_s = \sum_{l \in [\ell]} \widetilde{A}_l^s \widetilde{B}_l^s + \widetilde{M}_s$.

4) Reconstruction: From any R answers, the master decodes $\mathbf{AB}$.

5) Analysis: The proof of recovery threshold is identical to GCSA codes because the noise-alignment preserves the structure of both desired symbols and interference. Strong security is guaranteed because inter-server communication only involves $\mathcal{Z}^{server}$ matrices. Security is guaranteed by the inclusion of $\mathcal{Z}^A, \mathcal{Z}^B$ matrices into the shares sent to the servers by the source nodes. Privacy is guaranteed due to the aligned noise that is added to the answers by the servers.

Consider the communication cost. The source upload cost $U_A = \frac{S}{K_c pm}, U_B = \frac{S}{K_c pn}$. The server communication cost $CC = \frac{S-1}{\ell K_c mn}$. Note that the master is able to recover Lmn desired symbols from R downloaded symbols, the master download cost is $D = \frac{R}{Lmn} = \frac{pmn(\ell+1)K_c + 2X - 1}{\ell K_c mn}$. Thus the desired costs are achievable.

Now let us consider the computation complexity. Note that the source encoding procedure can be regarded as products of confluent Cauchy matrices by vectors [39]. By fast algorithms [45], the encoding complexity of $(\mathcal{C}_{eA}, \mathcal{C}_{eB}) = \left(\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S \log^2 S}{K_c pm}\right), \widetilde{\mathcal{O}}\left(\frac{\kappa\mu S \log^2 S}{K_c pn}\right)\right)$ is achievable. For the server computation complexity, each server multiplies the ℓ pairs of shares $\widetilde{A}_l^s, \widetilde{B}_l^s, l \in [\ell]$, and returns the sum of these ℓ products and structured noise $\widetilde{M}_s$. With straightforward matrix multiplication algorithms, each of the ℓ matrix products has a computation complexity of $\mathcal{O}\left(\frac{\lambda\kappa\mu}{pmn}\right)$ for a total of $\mathcal{O}\left(\frac{\ell\lambda\kappa\mu}{pmn}\right)$. The complexity of summation over the products and noise is $\mathcal{O}\left(\frac{\ell\lambda\mu}{mn}\right)$. To construct the aligned noise, one server needs to encode the noise, whose complexity is $\widetilde{\mathcal{O}}\left(\frac{\lambda\mu S \log^2 S}{mn}\right)$ by fast algorithms [45]. Normalized by the number of servers, it is $\widetilde{\mathcal{O}}\left(\frac{\lambda\mu \log^2 S}{mn}\right)$. Consider these 3 procedures, upon normalization by $L = \ell K_c$, it yields a complexity of $\mathcal{O}\left(\frac{\lambda\kappa\mu}{K_c pmn}\right) + \mathcal{O}\left(\frac{\lambda\mu}{K_c mn}\right) + \widetilde{\mathcal{O}}\left(\frac{\lambda\mu \log^2 S}{\ell K_c mn}\right)$ per server. Now let us consider the master decoding complexity. Note that the decoding procedure is identical to GCSA codes, so by fast algorithms [45], [46], the complexity of decoding is at most $\widetilde{\mathcal{O}}(\lambda\mu p \log^2 R)$. This completes the proof.

Remark: When $L = \ell = K_c = 1$, $S = R$, by setting $f_{1,1} = 0$, our construction of shares of $\widetilde{A}^s$ and $\widetilde{B}^s$ is indeed equivalent to the construction of shares in the PS code [40].

Remark: As explained in our full paper [47], noise alignment can also be applied to the scheme proposed in [48] .

V. Conclusion

For the problem of multiplication of two matrices, the class of GCSA codes is expanded by including noise-alignment, so that the resulting GCSA-NA code strictly generalizes PS [40] and outperforms it in several key aspects. However, while converse proofs remain unavailable, the fundamental limits of the SMBMM problem are open.

VI. Acknowledgement

This work is supported in part by funding from NSF grants CNS-1731384 and CCF-1907053, ONR grant N00014-18-1-2057 and ARO grant W911NF-19-1-0344.

References

[1] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[2] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication," *arXiv preprint arXiv:1705.10464*, 2017.

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

[4] S. Dutta, Z. Bai, H. Jeong, T. Low, and P. Grover, "A Unified Coded Deep Neural Network Training Strategy Based on Generalized PolyDot Codes for Matrix Multiplication," *ArXiv:1811.10751*, Nov. 2018.

[5] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[6] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. Avestimehr, "Lagrange Coded Computing: Optimal Design for Resiliency, Security and Privacy," *ArXiv:1806.00939*, 2018.

[7] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded Computation over Heterogeneous Clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

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

[9] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems*, 2016, pp. 2100–2108.

[10] ——, "Coded convolution for parallel and distributed computing within a deadline," *arXiv preprint arXiv:1705.03875*, 2017.

[11] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded fourier transform," *arXiv preprint arXiv:1710.06471*, 2017.

[12] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplications," *arXiv preprint arXiv:1812.10460*, 2018.

[13] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1993–1997.

[14] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 1271–1278.

[15] S. Wang, J. Liu, N. Shroff, and P. Yang, "Fundamental limits of coded linear transform," *arXiv preprint arXiv:1804.09791*, 2018.

[16] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, 2019.

[17] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," *arXiv preprint arXiv:1802.03430*, 2018.

[18] A. Severinson, A. G. i Amat, and E. Rosnes, "Block-diagonal and lt codes for distributed computing with straggling servers," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 1739–1753, 2018.

[19] F. Haddadpour and V. R. Cadambe, "Codes for distributed finite alphabet matrix-vector multiplication," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1625–1629.

[20] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, "An application of storage-optimal matdot codes for coded matrix multiplication: Fast k-nearest neighbors estimation," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1113–1120.

[21] H. Jeong, F. Ye, and P. Grover, "Locally recoverable coded matrix multiplication," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 715–722.

[22] M. Kim, J.-y. Sohn, and J. Moon, "Coded matrix multiplication on a group-based model," *arXiv preprint arXiv:1901.05162*, 2019.

[23] H. Park, K. Lee, J.-y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," *arXiv preprint arXiv:1801.04686*, 2018.

[24] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 34–40, 2017.

[25] W. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2018.

[26] J. Kakar, S. Ebadifar, and A. Sezgin, "On the capacity and straggler-robustness of distributed secure matrix multiplication," *IEEE Access*, vol. 7, pp. 45 783–45 799, 2019.

[27] R. G. D'Oliveira, S. E. Rouayheb, and D. Karpuk, "Gasp codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, 2020, early access, DOI: 10.1109/TIT.2020.2975021.

[28] M. Kim and J. Lee, "Private secure coded computation," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1918–1921, 2019.

[29] M. Aliasgari, O. Simeone, and J. Kliewer, "Distributed and private coded matrix computation with flexible communication load," *arXiv preprint arXiv:1901.07705*, 2019.

[30] H. Sun and S. A. Jafar, "The Capacity of Private Information Retrieval," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4075–4088, July 2017.

[31] ——, "The Capacity of Robust Private Information Retrieval with Colluding Databases," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2361–2370, April 2018.

[32] K. Banawan and S. Ulukus, "The Capacity of Private Information Retrieval from Coded Databases," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1945–1956, 2018.

[33] K. Banawan and S. Ulukus, "The capacity of private information retrieval from byzantine and colluding databases," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 1206–1219, Feb 2019.

[34] S. Kadhe, B. Garcia, A. Heidarzadeh, S. E. Rouayheb, and A. Sprintson, "Private information retrieval with side information," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2032–2043, 2020.

[35] Q. Wang and M. Skoglund, "Secure symmetric private information retrieval from colluding databases with adversaries," *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1083–1090, 2017.

[36] Z. Jia, H. Sun, and S. A. Jafar, "Cross Subspace Alignment and the Asymptotic Capacity of X -Secure T -Private Information Retrieval," *IEEE Trans. on Info. Theory*, vol. 65, no. 9, pp. 5783–5798, Sep. 2019.

[37] Z. Jia and S. A. Jafar, "On the Asymptotic Capacity of X-Secure T-Private Information Retrieval with Graph Based Replicated Storage," *ArXiv:1904.05906*, 2019.

[38] ——, "X-secure T-private Information Retrieval from MDS Coded Storage with Byzantine and Unresponsive Servers," *ArXiv:1908.10854*, 2019.

[39] Z. Jia and S. Jafar, "Cross-subspace alignment codes for coded distributed batch computation," *ArXiv:1909.13873*, 2019.

[40] H. A. Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *ArXiv:1908.04255*, 2019.

[41] A. C. Yao, "Protocols for secure computations (extended abstract)," *23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, 1982.

[42] W. Zhao, X. Ming, S. Mikael, and P. H. Vincent, "Secure degrees of freedom of wireless x networks using artificial noise alignment," *IEEE Transactions on communications*, vol. 63, no. 7, pp. 2632–2646, 2015.

[43] H. A. Nodehi and M. A. Maddah-Ali, "Limited-sharing multi-party computation for massive matrix operations," *IEEE International Symposium on Information Theory*, 2018.

[44] H. A. Nodehi, S. R. H. Najarkolaei, and M. A. Maddah-Ali, "Entangled polynomial coding in limited-sharing multi-party computation," *IEEE Information Theory Workshop*, 2018.

[45] V. Olshevsky and A. Shokrollahi, "A Superfast Algorithm for Confluent Rational Tangential Interpolation Problem via Matrix-Vector Multiplication for Confluent Cauchy-like Matrices," *Contemporary Mathematics*, vol. 280, pp. 31–46, 2001.

[46] I. Gohberg and V. Olshevsky, "Fast Algorithms with Preprocessing for matrix-Vector multiplication Problems," *Journal of Complexity*, vol. 10, no. 4, pp. 411–427, 1994.

[47] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "GCSA Codes with Noise Alignment for Secure Coded Multi-Party Batch Matrix Multiplication," *ArXiv:2002.07750*, 2020.

[48] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the "cubic" barrier," *ArXiv:2001.05101*, 2020.