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

Zhen Chen, *Graduate Student Member, IEEE*, Zhuqing Jia, *Graduate Student Member, IEEE*, Zhiying Wang, *Member, IEEE*, and Syed Ali Jafar, *Fellow, IEEE*

***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. The idea of noise alignment can also be combined with N-source Cross Subspace Alignment (N-CSA) codes and fast matrix multiplication algorithms like Strassen's construction. Moreover, noise alignment can be applied to symmetric secure private information retrieval to achieve the asymptotic capacity.**

***Index Terms*—Distributed computing, coded computing, matrix multiplication, secure multi-party computation, noise alignment.**

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]–[40]. These related ideas intersected recently in Generalized Cross Subspace Alignment (GCSA) codes presented in [40]. GCSA codes originated in the setting of secure private information retrieval [37] and have recently been developed further in [40] for applications to coded distributed batch computation problems. GCSA codes generalize and improve upon the state of art distributed computing schemes such as Polynomials codes [2], MatDot codes and PolyDot codes [3], Generalized PolyDot codes [4] and Entangled Polynomial (EP) Codes [5] that partition matrices into submatrices, as well as Lagrange Coded Computing [6], [7] that allows batch processing of multiple computations.

As the next step in the expanding scope of coding for distributed computing, recently in [41] Nodehi and Maddah-Ali explored its application to secure multiparty computation [42]. 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 is a concern because communication increases the risk for collusion. Furthermore, PS 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 and Maddah-Ali [41].

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 secure multiplication of two matrices. Such a problem may arise, e.g., in correlation analysis between privately held genomic datasets to determine genetic connections without revealing anything else. 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 [40]. 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

Manuscript received August 14, 2020; revised November 17, 2020; accepted January 14, 2021. Date of publication January 22, 2021; date of current version March 16, 2021. This work was supported in part by NSF Grant CNS-1731384 and Grant CCF-1907053; in part by the Office of Naval Research (ONR) under Grant N00014-18-1-2057; and in part by the Army Research Office (ARO) under Grant W911NF-19-1-0344. *(Corresponding author: Zhen Chen.)*

The authors are with the Center for Pervasive Communications and Computing, Department of Electrical Engineering and Computer Science, University of California at Irvine, Irvine, CA 92697 USA (e-mail: zhenc4@uci.edu; zhuqingj@uci.edu; zhiying@uci.edu; syed@uci.edu).

Digital Object Identifier 10.1109/JSAIT.2021.3052934

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 [43].

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 [41]. The only reason any inter-server communication is needed in a GCSA-NA scheme is to share the aligned noise terms 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. 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. In fact, the GCSA-NA scheme works even if inter-server communication is *entirely* disallowed, because the aligned noise can be equivalently generated by either of the source nodes and sent to the servers. By disallowing communication among servers, GCSA-NA may reduce the probability of collusion among servers relative to PS where all servers must communicate with each other.

The rest of this article is organized as follows. Section II presents the problem statement. In Section III we state the main result and compare it with previous approaches. A toy example is presented in Section IV. The construction and proof of GCSA-NA are shown in Section V. Section VI concludes this article.

Notation: For positive integers M, N ($M < N$), $[N]$ stands for the set $\{1, 2, \ldots, N\}$ and $[M : N]$ stands for the set $\{M, M+1, \ldots, N\}$. For a set $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$, $X_{\mathcal{I}}$ denotes the set $\{X_{i_1}, X_{i_2}, \ldots, X_{i_N}\}$. The notation $\otimes$ denotes the Kronecker product of two matrices. $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. $\mathbf{T}(X_1, X_2, \ldots, X_N)$ denotes the $N \times N$ lower triangular Toeplitz matrix, i.e.,

$$\mathbf{T}(X_1, X_2, \ldots, X_N) = \begin{bmatrix} X_1 & & & \\ X_2 & X_1 & & \\ \vdots & \ddots & \ddots & \\ X_N & \cdots & X_2 & X_1 \end{bmatrix}.$$

For a matrix M, $|M|$ denotes the number of elements in M. For a polynomial P, $deg_\alpha(P)$ denotes the degree with respect to a variable α. Define the degree of the zero polynomial as -1. The notation $\widetilde{\mathcal{O}}(a \log^2 b)$ suppresses polylog terms for computation complexity. It may be replaced with $\mathcal{O}(a \log^2 b)$ if the field $\mathbb{F}$ supports the Fast Fourier Transform (FFT), and with $\mathcal{O}(a \log^2 b \log\log(b))$ if it does not.

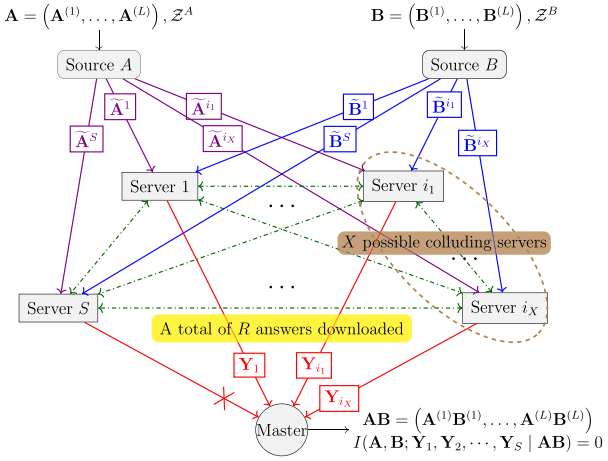

Fig. 1. The SMBMM problem. Sources generate matrices $\mathbf{A} = (\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(L)})$ with separate noise $\mathcal{Z}^A$ and $\mathbf{B} = (\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)})$ with separate noise $\mathcal{Z}^B$, and upload information to S distributed servers in coded form $\widetilde{\mathbf{A}}^{[S]}$, $\widetilde{\mathbf{B}}^{[S]}$, respectively. Servers may communicate with each other via dash-dotted links. For security, any X colluding servers (e.g., Servers i_1 to i_X in the figure) learn nothing about $\mathbf{A}, \mathbf{B}$. The s^{th} server computes the answer $\mathbf{Y}_s$, which is a function of all information available to it. For effective straggler (e.g., Server S in the figure) mitigation, upon downloading answers from any R servers, where $R < S$, the master must be able to recover the product $\mathbf{AB} = (\mathbf{A}^{(1)}\mathbf{B}^{(1)}, \mathbf{A}^{(2)}\mathbf{B}^{(2)}, \ldots, \mathbf{A}^{(L)}\mathbf{B}^{(L)})$. For privacy, the master must not gain any additional information about $\mathbf{A}, \mathbf{B}$ beyond the desired product $\mathbf{AB}$.

II. Problem Statement

Consider a system including 2 sources (A and B), 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,[1] and all of the servers are connected to the master. All of these links are secure and error free.

Source A and B independently generate sequences[2] of L matrices, denoted as $\mathbf{A} = (\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(L)})$, and $\mathbf{B} = (\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)})$, respectively, such that $\forall l \in [L]$, $\mathbf{A}^{(l)} \in \mathbb{F}^{\lambda \times \kappa}$ and $\mathbf{B}^{(l)} \in \mathbb{F}^{\kappa \times \mu}$. The master is interested in the sequence of product matrices, $\mathbf{AB} = (\mathbf{A}^{(1)}\mathbf{B}^{(1)}, \mathbf{A}^{(2)}\mathbf{B}^{(2)}, \ldots, \mathbf{A}^{(L)}\mathbf{B}^{(L)})$. The system operates in three phases: 1) sharing, 2) computation and communication, and 3) reconstruction.

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

2) *Computation and Communication:* Servers may send messages to other servers, and process what they received from both the sources and other servers. Denote the communication from Server s to Server s' as $\mathbf{M}_{s \to s'}$. Define

[1] While we *allow* these links (dash-dotted lines in Figure 1) for the sake of consistency with the original formulation in [41], these links are not necessary for our solution. See the remark following the definition of security & strong security.

[2] The batch size L can be chosen to be arbitrarily large by the coding algorithm.

$\mathcal{M}_s \triangleq \{\mathbf{M}_{s'\to s} | s' \in [S] \setminus \{s\}\}$ as the messages that Server s receives from other servers, and $\mathcal{M} \triangleq \{\mathcal{M}_s | s \in [S]\}$ as the total messages that all servers receive. After the communication among servers, each server s computes a response $\mathbf{Y}_s$ and sends it to the master. $\mathbf{Y}_s$ is a function of $\widetilde{\mathbf{A}}^s$, $\widetilde{\mathbf{B}}^s$ and $\mathcal{M}_s$, i.e., $\mathbf{Y}_s = h_s(\widetilde{\mathbf{A}}^s, \widetilde{\mathbf{B}}^s, \mathcal{M}_s)$, 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 $\boldsymbol{d} = \{d_{\mathcal{R}} | \mathcal{R} \subset [S]\}$ 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.,

$$H(\mathbf{AB}|\mathbf{Y}_{\mathcal{R}}) = 0, \tag{1}$$

or equivalently $\mathbf{AB} = d_{\mathcal{R}}(\mathbf{Y}_{\mathcal{R}})$, for some $\mathcal{R}$.

Security & Strong Security: We first define *security* which is called privacy for workers in [41]. The servers must remain oblivious to the content of the data $\mathbf{A}, \mathbf{B}$, even if X of them collude. Formally, $\forall \mathcal{X} \subset [S], |\mathcal{X}| \leq X$,

$$I\Big(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}^{\mathcal{X}}, \widetilde{\mathbf{B}}^{\mathcal{X}}, \mathcal{M}_{\mathcal{X}}\Big) = 0. \tag{2}$$

In this article, *strong security* is also considered. It requires that the information transmitted among servers is independent of data $\mathbf{A}, \mathbf{B}$ and all the shares $\widetilde{\mathbf{A}}^{[\mathcal{S}]}, \widetilde{\mathbf{B}}^{[\mathcal{S}]}$, i.e.,

$$I\Big(\mathbf{A}, \mathbf{B}, \widetilde{\mathbf{A}}^{[\mathcal{S}]}, \widetilde{\mathbf{B}}^{[\mathcal{S}]}; \mathcal{M}\Big) = 0. \tag{3}$$

This property makes it possible that inter-server communications happen before receiving data from sources, and makes the server communication network topology more flexible. Note that PS does not satisfy strong security because $H(\mathbf{AB}|\mathcal{M}) = 0$ in the PS scheme.

Remark: $\mathcal{M}$ can be shared among servers in various ways that satisfy strong security. For example, the servers can share $\mathcal{M}$ *a-priori* during an initial setup phase, so that there is no communication among servers during the actual computation phase. Alternatively, $\mathcal{M}$ can come from a service provider whose sole purpose is to generate structured noise and send it to each server. Finally, $\mathcal{M}$ can also be separately generated by either of the source nodes (independent of the input matrices and their coded shares) and sent to each server. This only makes uses of existing communication links between the source and server nodes, and requires no communication between servers.

Privacy: The master must not gain any additional information about $\mathbf{A}, \mathbf{B}$, beyond the required product. Precisely,

$$I(\mathbf{A}, \mathbf{B}; \mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_S | \mathbf{AB}) = 0. \tag{4}$$

This is the privacy for the master in [41].

Remark: There is another setting for secure distributed matrix multiplication that appears in the literature, where the input matrices originate at the master node itself [26]–[30], [44]. In that case, while the solution presented in this work will still apply, the privacy aspect would be irrelevant because the master already knows both $\mathbf{A}$ and $\mathbf{B}$. Correspondingly, our solution degenerates to a special case called X-secure GCSA codes (see [40, Remark 2, Appendix A]). Since privacy is an important aspect of this work, we assume that the source nodes are distinct from the master node as shown in Figure 1.

We say that $(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{d})$ form an SMBMM (**S**ecure coded **M**ulti-party **B**atch **M**atrix **M**ultiplication) 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 of an SMBMM code is comprised of these parts: upload cost of the sources, communication cost among the servers, and download cost of the master. The (normalized)[3] upload costs U_A and U_B are defined as follows.

$$U_A = \frac{\sum_{s\in[S]} |\widetilde{\mathbf{A}}^s|}{L\lambda\kappa}, \quad U_B = \frac{\sum_{s\in[S]} |\widetilde{\mathbf{B}}^s|}{L\kappa\mu}. \tag{5}$$

Similarly, the (normalized) server communication cost CC and download cost D are defined as follows.

$$CC = \frac{\sum_{\mathbf{M}\in\mathcal{M}} |\mathbf{M}|}{L\lambda\mu}, \quad D = \max_{\mathcal{R}, \mathcal{R}\subset[S], |\mathcal{R}|=R} \frac{\sum_{s\in\mathcal{R}} |\mathbf{Y}_s|}{L\lambda\mu}. \tag{6}$$

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 $\widetilde{\mathbf{A}}^{[S]}$ and $\mathcal{C}_{eB}$ for $\widetilde{\mathbf{B}}^{[S]}$ 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 the order of the number of arithmetic operations required to compute $d_{\mathcal{R}}(\mathbf{Y}_{\mathcal{R}})$, maximized over $\mathcal{R}, \mathcal{R} \subset [S], |\mathcal{R}| = R$, and normalized by L. Note that normalization by batch-size L is needed to have fair comparisons between batch processing approaches and individual matrix-partitioning solutions *per matrix multiplication*.

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 $m|\lambda$, $p|\kappa$, $n|\mu$ and $L = \ell K_c \leq |\mathbb{F}| - S$, the GCSA-NA scheme

[3]We normalize source upload cost with the number of elements contained in the constituent matrices $\mathbf{A}, \mathbf{B}$. The server communication cost and master download cost are normalized by the number of elements contained in the desired product $\mathbf{AB}$.

TABLE I
PERFORMANCE COMPARISON OF PS AND *GCSA-NA*

	Polynomial Sharing (PS [41])	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 ($\mathcal{C}_d$)	$\widetilde{\mathcal{O}}\left(\lambda\mu\log^2(mn+X)\right)$	$\widetilde{\mathcal{O}}\left(\lambda\mu p\log^2(R)\right)$

presented in Section V is a solution, and its recovery threshold, cost, and complexity are listed as follows.

Recovery Threshold: $R = pmn(\ell+1)K_c + 2X - 1$,

Source Upload Cost of $\widetilde{A}^{[S]}, \widetilde{B}^{[S]}$: $(U_A, U_B) = \left(\frac{S}{K_c pm}, \frac{S}{K_c pn}\right)$,

Server Communication Cost: $CC = \frac{S-1}{\ell K_c mn}$,

Master Download Cost: $D = \frac{R}{\ell K_c mn}$,

Source Encoding Complexity for $\widetilde{A}^{[S]}, \widetilde{B}^{[S]}$:

$$(\mathcal{C}_{eA}, \mathcal{C}_{eB}) = \left(\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_c pm}\right), \widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_c pn}\right)\right),$$

Server Computation Complexity: $\mathcal{C}_s = \mathcal{O}\left(\frac{\lambda\kappa\mu}{K_c pmn}\right)$,

Master Decoding Complexity: $\mathcal{C}_d = \widetilde{\mathcal{O}}\left(\lambda\mu p\log^2 R\right)$.

The following observations place the result of Theorem 1 in perspective.

1. GCSA-NA codes are based on the construction of GCSA codes from [40], combined with the idea of noise-alignment (e.g., [43]). In turn, GCSA codes are based on a combination of CSA codes for batch processing [40] and EP codes for matrix partitioning [5]. CSA codes are themselves based on the idea of Cross-Subspace Alignment (CSA) that was introduced in the context of secure Private Information Retrieval (PIR) [37]. It is a remarkable coincidence that while the idea of CSA originated in the context of PIR [37], and Lagrange Coded Computing (LCC) was introduced in parallel independently in [6] for the context of coded computing, the two approaches are essentially identical, with CSA codes being slightly more powerful in the context of coded distributed matrix multiplication (CSA codes offer additional improvements over LCC codes in terms of download cost [40]). Indeed, LCC codes for batch matrix multiplication are recovered as a special case of CSA codes.

2. The idea of noise alignment can be applied to the N-CSA codes [40], for N-source secure coded multi-party batch matrix computation. In [7], Strassen's construction [45], combined with LCC, are introduced for batch distributed matrix multiplication for better computation complexity. Noise alignment is also applicable to Strassen's constructions (see Section VI). By setting $K_c = 1$, $\ell = L$ and $S = R$, the construction of GCSA-NA codes, with a straightforward generalization, can be further modified to settle the asymptotic (the number of messages go to infinity) capacity of symmetric X-secure T-private computation (and also symmetric X-secure T-private information retrieval XSTPIR) [37]. However, the amount of randomness required by the construction is not necessarily optimal. For example, it is shown in [37] that by the achievable scheme for XSTPIR, symmetric security (privacy) is automatically satisfied when $T = 1$, i.e., no randomness among servers is required.

3. A side-by-side comparison of the GCSA-NA solution with polynomial sharing (PS) appears in Table I. Because all inter-server communication is independent of input data, 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. 2(a). 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). This advantage is particularly significant for large number

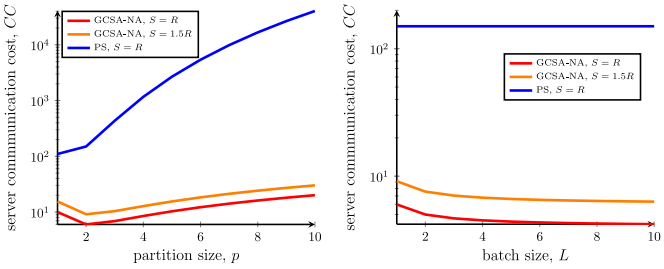

Fig. 2. $\lambda = \kappa = \mu$, $p = m = n$. (a) Server communication cost vs. partition size, given $L = 1, X = 5$. (b) Server communication cost vs. batch size, given $p = 2, X = 5$.

of servers. The GCSA-NA scheme naturally allows robustness to stragglers, which is particularly important for massive matrix multiplications. Stragglers can be an especially significant concern for PS because of the strongly sequential nature of multi-round computation that is central to PS. This is because server failures between computation rounds disrupt the computation sequence. Remarkably, Fig. 2(a) shows that the inter-server communication cost of GCSA-NA is significantly better than PS even when GCSA-NA accommodates stragglers (while PS does not).

When restricted to batch size 1, i.e., with $\ell = K_c = 1$, GCSA-NA has the same recovery threshold as PS. 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. 2(b) 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.

IV. 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 $\mathbf{B} = [\mathbf{B}_1^T \ \mathbf{B}_2^T]^T$ 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 [41], [46], [47], and our GCSA-NA approach.

A. 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

$$\mathbf{P} = \mathbf{A}_1 + \alpha\mathbf{A}_2, \quad \mathbf{Q} = \alpha\mathbf{B}_1 + \mathbf{B}_2. \tag{7}$$

$$\implies \mathbf{PQ} = \mathbf{A}_1\mathbf{B}_2 + \alpha(\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2) + \alpha^2\mathbf{A}_2\mathbf{B}_1. \tag{8}$$

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

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{B}} = \mathbf{PQ} + \alpha^2 P\mathbf{Z}^B + \alpha^2\mathbf{Z}^A\mathbf{Q} + \alpha^4\mathbf{Z}^A\mathbf{Z}^B \tag{9}$$

$$= \mathbf{A}_1\mathbf{B}_2 + \alpha(\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2) + \alpha^2\Big(\mathbf{A}_2\mathbf{B}_1 + \mathbf{A}_1\mathbf{Z}^B + \mathbf{Z}^A\mathbf{B}_2\Big) + \alpha^3\Big(\mathbf{A}_2\mathbf{Z}^B + \mathbf{Z}^A\mathbf{B}_1\Big) + \alpha^4\mathbf{Z}^A\mathbf{Z}^B. \tag{10}$$

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 $deg_\alpha(\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}) = 4$, $\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2$ can be calculated from 5 distinct $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}$ according to the Lagrange interpolation rules. In particular, there exist 5 constants $r_1, \ldots, r_5$, such that $\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 = \sum_{s\in[5]} r_s\widetilde{\mathbf{A}}^s\widetilde{\mathbf{B}}^s$. Consider Server s, it sends $\mathbf{M}_{s\to j} = r_s\widetilde{\mathbf{A}}^s\widetilde{\mathbf{B}}^s + \alpha_j\mathbf{Z}_s$ to Server j, where $\mathbf{Z}_1, \ldots, \mathbf{Z}_5$ are i.i.d. uniform noise matrices. After Server s collects all the shares $M_{j\to s}$, it sums them up

$$\mathbf{Y}_s = \sum_{j\in[5]} \mathbf{M}_{j\to s} = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 + \alpha_s \sum_{j\in[5]} \mathbf{Z}_j \tag{11}$$

and sends $\mathbf{Y}_s$ to the master. Note that after receiving $\mathbf{M}_{j\to s}$ 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 $\mathbf{M}_{j\to s}, j, s \in [5]$.

The master can decode $\mathbf{AB}$ after collecting 2 responses from servers.[4] Note that PS needs at least $S = R = 5$ servers, since 5 distinct $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}$ are required to obtain $\mathbf{Y}_s$.

B. GCSA-NA Solution

GCSA codes [40] 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,

$$\mathbf{P} = \mathbf{A}_1 + (f - \alpha)\mathbf{A}_2, \quad \mathbf{Q} = (f - \alpha)\mathbf{B}_1 + \mathbf{B}_2. \tag{12}$$

$$\mathbf{P}' = \mathbf{A}_1' + \big(f' - \alpha\big)\mathbf{A}_2', \quad \mathbf{Q}' = \big(f' - \alpha\big)\mathbf{B}_1' + \mathbf{B}_2'. \tag{13}$$

And the shares are constructed as follows,

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

where $\Delta = (f-\alpha)^2(f'-\alpha)^2$, and $\alpha, \widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}$ are generic variables that should be replaced with $\alpha_s, \widetilde{\mathbf{A}}^s, \widetilde{\mathbf{B}}^s$ for Server s. Furthermore, $f, f', \alpha_1, \alpha_2, \ldots, \alpha_S$ are distinct elements. Each server computes the product of the shares that it receives, i.e.,

[4] In [47], for arbitrary polynomials, $\mathbf{M}_{s\to j} = r_s\widetilde{\mathbf{A}}^s\widetilde{\mathbf{B}}^s + \alpha_j^2\mathbf{Z}_s$ because $\mathbf{Y}_s$ is forced to be casted in the form of entangled polynomial sharing.

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{B}} = \frac{(f'-\alpha)^2}{(f-\alpha)^2}\mathbf{PQ} + \frac{(f-\alpha)^2}{(f'-\alpha)^2}\mathbf{P'Q'} + \mathbf{P'Q} + \mathbf{PQ'} \tag{14}$$

$$= \frac{\big((f'-f)+(f-\alpha)\big)^2}{(f-\alpha)^2}\mathbf{PQ} + \frac{\big((f-f')+(f'-\alpha)\big)^2}{(f'-\alpha)^2}\mathbf{P'Q'} + \mathbf{P'Q} + \mathbf{PQ'} \tag{15}$$

$$= \frac{(f'-f)^2 + 2(f'-f)(f-\alpha) + (f-\alpha)^2}{(f-\alpha)^2}\mathbf{PQ} + \frac{(f-f')^2 + 2(f-f')(f'-\alpha) + (f'-\alpha)^2}{(f'-\alpha)^2}\mathbf{P'Q'} + \mathbf{P'Q} + \mathbf{PQ'} \tag{16}$$

$$= \frac{(f'-f)^2}{(f-\alpha)^2}\mathbf{PQ} + \frac{2(f'-f)}{f-\alpha}\mathbf{PQ} + \frac{(f-f')^2}{(f'-\alpha)^2}\mathbf{P'Q'} + \frac{2(f-f')}{f'-\alpha}\mathbf{P'Q'} + \mathbf{PQ} + \mathbf{P'Q'} + \mathbf{P'Q} + \mathbf{PQ'} \tag{17}$$

$$= \frac{c_0}{(f-\alpha)^2}\mathbf{PQ} + \frac{c_1}{f-\alpha}\mathbf{PQ} + \frac{c_0'}{(f'-\alpha)^2}\mathbf{P'Q'} + \frac{c_1'}{f'-\alpha}\mathbf{P'Q'} + \mathbf{I}_0 + \alpha\mathbf{I}_1 + \alpha^2\mathbf{I}_2 \tag{18}$$

$$= \frac{c_0\mathbf{A}_1\mathbf{B}_2}{(f-\alpha)^2} + \frac{c_0\mathbf{A}_1\mathbf{B}_1 + c_0\mathbf{A}_2\mathbf{B}_2 + c_1\mathbf{A}_1\mathbf{B}_2}{f-\alpha} + \frac{c_0'\mathbf{A}_1'\mathbf{B}_2'}{(f'-\alpha)^2} + \frac{c_0'\mathbf{A}_1'\mathbf{B}_1' + c_0'\mathbf{A}_2'\mathbf{B}_2' + c_1'\mathbf{A}_1'\mathbf{B}_2'}{f'-\alpha} + \mathbf{I}_0 + \alpha\mathbf{I}_1 + \alpha^2\mathbf{I}_2, \tag{19}$$

where $\mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2$ are various linear combinations of $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2, \mathbf{A'}_1, \mathbf{A'}_2, \mathbf{B'}_1, \mathbf{B'}_2$ and c_0, c_1, c_0', c_1' are constants. Their exact forms can be found by performing partial fraction decomposition. This is the original GCSA code [40], and we need $R = pmn((\ell+1)K_c - 1) + p - 1 = 7$ 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.,

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

$$\widetilde{\mathbf{B}} = \frac{\mathbf{Q}}{(f-\alpha)^2} + \frac{\mathbf{Q}'}{(f'-\alpha)^2} + \mathbf{Z}^B.$$

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{B}} = \frac{c_0\mathbf{PQ}}{(f-\alpha)^2} + \frac{c_1\mathbf{PQ}}{f-\alpha} + \frac{c_0'\mathbf{P'Q'}}{(f'-\alpha)^2} + \frac{c_1'\mathbf{P'Q'}}{f'-\alpha} + \sum_{i=0}^{4}\alpha^i\mathbf{I}_i.$$

As a result of the added noise terms, the recovery threshold is now increased to 9. Note that the term $\mathbf{I}_4$ 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 $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}$ 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 interference terms $(\frac{c_0}{(f-\alpha)^2} + \frac{c_1}{f-\alpha})\mathbf{A}_1\mathbf{B}_2 + (\frac{c_0'}{(f'-\alpha)^2} + \frac{c_1'}{f'-\alpha})\mathbf{A}_1'\mathbf{B}_2' + \sum_{i=0}^{3}\alpha^i\mathbf{I}_i$, which can be secured by the addition of *aligned noise* terms $\widetilde{\mathbf{Z}} = (\frac{c_0}{(f-\alpha)^2} + \frac{c_1}{f-\alpha})\mathbf{Z} + (\frac{c_0'}{(f'-\alpha)^2} + \frac{c_1'}{f'-\alpha})\mathbf{Z}' + \sum_{i=0}^{3}\alpha^i\mathbf{Z}_i$ at each server so that the answer returned by each server to the master is $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}} + \widetilde{\mathbf{Z}}$. 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 $\widetilde{\mathbf{Z}}$ with all other servers. This sharing of $\widetilde{\mathbf{Z}}$ is the only inter-server communication needed in GCSA-NA. Since it is independent of the inputs, it can be done during off-peak hours, thereby reducing the latency of server computation. The strong security is also automatically satisfied.

V. Construction of GCSA-NA

Now let us present the general construction. $L = \ell K_c$ instances of $\mathbf{A}$ and $\mathbf{B}$ matrices are split into ℓ groups. $\forall l \in [\ell], \forall k \in [K_c]$, denote

$$\mathbf{A}^{l,k} = \mathbf{A}^{(K_c(l-1)+k)}, \ \mathbf{B}^{l,k} = \mathbf{B}^{(K_c(l-1)+k)}. \tag{20}$$

Further, each matrix $\mathbf{A}^{l,k}$ is partitioned into $m \times p$ blocks and each matrix $\mathbf{B}^{l,k}$ is partitioned into $p \times n$ blocks, i.e.,

$$\mathbf{A}^{l,k} = \begin{bmatrix} \mathbf{A}_{1,1}^{l,k} & \cdots & \mathbf{A}_{1,p}^{l,k} \\ \mathbf{A}_{2,1}^{l,k} & \cdots & \mathbf{A}_{2,p}^{l,k} \\ \vdots & \vdots & \vdots \\ \mathbf{A}_{m,1}^{l,k} & \cdots & \mathbf{A}_{m,p}^{l,k} \end{bmatrix}, \mathbf{B}^{l,k} = \begin{bmatrix} \mathbf{B}_{1,1}^{l,k} & \cdots & \mathbf{B}_{1,n}^{l,k} \\ \mathbf{B}_{2,1}^{l,k} & \cdots & \mathbf{B}_{2,n}^{l,k} \\ \vdots & \vdots & \vdots \\ \mathbf{B}_{p,1}^{l,k} & \cdots & \mathbf{B}_{p,n}^{l,k} \end{bmatrix},$$

where $(\mathbf{A}_{i,j}^{l,k})_{i\in[m],j\in[p]} \in \mathbb{F}^{\frac{\lambda}{m}\times\frac{\kappa}{p}}$ and $(\mathbf{B}_{i,j}^{l,k})_{i\in[m],j\in[p]} \in \mathbb{F}^{\frac{\kappa}{p}\times\frac{\mu}{n}}$.

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

$$R' = pmn, \quad D_E = \max(pm, pmn - pm + p) - 1, \tag{21}$$

$$\mathcal{E} = \big\{p + p(m'-1) + pm(n''-1) | m' \in [m], n'' \in [n]\big\}, \tag{22}$$

$$\Delta_s^{l,K_c} = \prod_{k\in[K_c]}(f_{l,k} - \alpha_s)^{R'}, \forall l \in [\ell], \forall s \in [S]. \tag{23}$$

Define $c_{l,k,i}, i \in \{0, 1, \ldots, R'(K_c - 1)\}$ to be the coefficients satisfying

$$\Psi_{l,k}(\alpha) = \prod_{k'\in[K_c]\setminus\{k\}}\big(\alpha + (f_{l,k'} - f_{l,k})\big)^{R'} = \sum_{i=0}^{R'(K_c-1)} c_{l,k,i}\alpha^i, \forall l \in [\ell], \forall k \in [K_c], \tag{24}$$

i.e., they are the coefficients of the polynomial $\Psi_{l,k}(\alpha) = \prod_{k'\in[K_c]\setminus\{k\}}(\alpha + (f_{l,k'} - f_{l,k}))^{R'}$, which is defined by its roots. Note that all the coefficients $(c_{l,k,i})_{l\in[L],k\in[K_c],i\in\{0,1,\ldots,R'(K_c-1)\}}$, $\alpha_{[S]}$, $(f_{l,k})_{l\in[L],k\in[K]}$ are globally known.

A. Sharing

Firstly, each source encodes each constituent matrix blocks $\mathbf{A}^{l,k}$ and $\mathbf{B}^{l,k}$ with Entangled Polynomial code [5]. For all $l \in [\ell], k \in [K_c]$, define

$$\mathbf{P}_s^{l,k} = \sum_{m'\in[m]}\sum_{p'\in[p]}\mathbf{A}_{m',p'}^{l,k}(f_{l,k} - \alpha_s)^{p'-1+p(m'-1)}, \tag{25}$$

$$\mathbf{Q}_s^{l,k} = \sum_{p''\in[p]}\sum_{n''\in[n]}\mathbf{B}_{p'',n''}^{l,k}(f_{l,k} - \alpha_s)^{p-p''+pm(n''-1)}. \tag{26}$$

Note that the original Entangled Polynomial code can be regarded as polynomials of α_s, and here for each (l, k),

Entangled Polynomial code is constructed as polynomials of $(f_{l,k} - \alpha_s)$.

Each source generates ℓX independent random matrices, $\mathcal{Z}^A = \{\mathbf{Z}^A_{1,1}, \ldots, \mathbf{Z}^A_{\ell,X}\}$ and $\mathcal{Z}^B = \{\mathbf{Z}^B_{1,1}, \ldots, \mathbf{Z}^B_{\ell,X}\}$. The independence is established as follows.

$$H\Big(\mathcal{Z}^A, \mathcal{Z}^B, \mathbf{A}, \mathbf{B}\Big) = H(\mathbf{A}) + H(\mathbf{B}) + \sum_{l\in[\ell], x\in[X]} H\Big(\mathbf{Z}^A_{l,x}\Big) + \sum_{l\in[\ell], x\in[X]} H\big(\mathbf{Z}^B_{l,x}\big). \tag{27}$$

For all $s \in [S]$, the shares of matrices $\mathbf{A}$ and $\mathbf{B}$ at the s^{th} server are constructed as $\widetilde{\mathbf{A}}^s = (\widetilde{\mathbf{A}}^s_1, \widetilde{\mathbf{A}}^s_2, \ldots, \widetilde{\mathbf{A}}^s_\ell)$, $\widetilde{\mathbf{B}}^s = (\widetilde{\mathbf{B}}^s_1, \widetilde{\mathbf{B}}^s_2, \ldots, \widetilde{\mathbf{B}}^s_\ell)$, where for all $l \in [\ell]$,

$$\widetilde{\mathbf{A}}^s_l = \Delta^{l,K_c}_s \left(\sum_{k\in[K_c]} \frac{\mathbf{P}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} + \sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^A_{l,x} \right),$$

$$\widetilde{\mathbf{B}}^s_l = \sum_{k\in[K_c]} \frac{\mathbf{Q}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} + \sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^B_{l,x}.$$

Then each pair of shares $\widetilde{\mathbf{A}}^s, \widetilde{\mathbf{B}}^s$ is sent to the corresponding server.

B. Computation and Communication

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

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

Here $\mathbf{Z}'_i$ and $\mathbf{Z}'''_{l,k,i}$ are the independent random matrices. The independence is established as follows.

$$H\big(\mathcal{Z}^{server}, \mathbf{A}, \mathbf{B}\big) = H(\mathbf{A}) + H(\mathbf{B}) + \sum_{i=1}^{R'(K_c-1)+X+D_E} H\big(\mathbf{Z}'_i\big) + \sum_{\substack{l\in[\ell], k\in[K_c], \\ i\in[R']}} H\big(\mathbf{Z}''_{l,k,i}\big). \tag{28}$$

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

$$\widetilde{\mathbf{M}}_s = \sum_{x=1}^{R'(K_c-1)+X+D_E} \alpha^{x-1}_s \mathbf{Z}'_x + \sum_{l\in[\ell]} \sum_{k\in[K_c]} \sum_{i=0}^{R'-1} \frac{\sum_{i'=0}^{i} c_{l,k,i-i'} \mathbf{Z}''_{l,k,i'+1}}{\big(f_{l,k} - \alpha_s\big)^{R'-i}} \tag{29}$$

and sends $\widetilde{\mathbf{M}}_s$ to server s, $s \in [S] \setminus \{1\}$, where $c_{l,k,i}$ is defined in (24). The answer returned by the s^{th} server to the master is constructed as $\mathbf{Y}_s = \sum_{l\in[\ell]} \widetilde{\mathbf{A}}^s_l \widetilde{\mathbf{B}}^s_l + \widetilde{\mathbf{M}}_s$.

C. Reconstruction

After the master collects any R answers, it decodes the desired products $\mathbf{AB}$.

D. Proof of Theorem 1

To begin, let us recall the standard result for Confluent Cauchy-Vandermonde matrices [48], replicated here for the sake of completeness.

Lemma 1: If $f_{1,1}, f_{1,2}, \ldots, f_{\ell,K_c}, \alpha_1, \alpha_2, \ldots, \alpha_R$ are $R + L$ distinct elements of $\mathbb{F}$, with $|\mathbb{F}| \geq R + L$, $L = \ell K_c$ and $R = R'(\ell + 1)K_c + 2X - 1$, then the $R \times R$ Confluent Cauchy-Vandermonde matrix (30) (shown at the bottom of the page) is invertible over $\mathbb{F}$.

Firstly, let us prove that the GCSA-NA codes are $R = pmn(\ell + 1)K_c + 2X - 1$ recoverable. Rewrite $\mathbf{Y}_s$ as follows.

$$\mathbf{Y}_s = \widetilde{\mathbf{A}}^s_1 \widetilde{\mathbf{B}}^s_1 + \widetilde{\mathbf{A}}^s_2 \widetilde{\mathbf{B}}^s_2 + \cdots + \widetilde{\mathbf{A}}^s_\ell \widetilde{\mathbf{B}}^s_\ell + \widetilde{\mathbf{M}}_s \tag{31}$$

$$= \sum_{l\in[\ell]} \Delta^{l,K_c}_s \left(\sum_{k\in[K_c]} \frac{\mathbf{P}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} + \sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^A_{l,x} \right) \times \left(\sum_{k\in[K_c]} \frac{\mathbf{Q}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} + \sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^B_{l,x} \right) + \widetilde{\mathbf{M}}_s \tag{32}$$

$$= \sum_{l\in[\ell]} \Delta^{l,K_c}_s \left(\sum_{k\in[K_c]} \frac{\mathbf{P}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} \right) \left(\sum_{k\in[K_c]} \frac{\mathbf{Q}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} \right) + \underbrace{\sum_{l\in[\ell]} \Delta^{l,K_c}_s \left(\sum_{k\in[K_c]} \frac{\mathbf{P}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} \right) \left(\sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^B_{l,x} \right)}_{\Gamma_2} + \underbrace{\sum_{l\in[\ell]} \Delta^{l,K_c}_s \left(\sum_{k\in[K_c]} \frac{\mathbf{Q}^{l,k}_s}{\big(f_{l,k} - \alpha_s\big)^{R'}} \right) \left(\sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^A_{l,x} \right)}_{\Gamma_3} + \underbrace{\sum_{l\in[\ell]} \Delta^{l,K_c}_s \left(\sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^A_{l,x} \right) \left(\sum_{x\in[X]} \alpha^{x-1}_s \mathbf{Z}^B_{l,x} \right)}_{\Gamma_4} + \widetilde{\mathbf{M}}_s \tag{33}$$

$$\hat{\mathbf{V}}_{\ell,K_c,R',X,R} \triangleq \begin{bmatrix} \frac{1}{(f_{1,1}-\alpha_1)^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_1} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_1)^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_1} & 1 & \cdots & \alpha_1^{R'K_c+2X-2} \\ \frac{1}{(f_{1,1}-\alpha_2)^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_2} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_2)^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_2} & 1 & \cdots & \alpha_2^{R'K_c+2X-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{(f_{1,1}-\alpha_R)^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_R} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_R)^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_R} & 1 & \cdots & \alpha_R^{R'K_c+2X-2} \end{bmatrix} \tag{30}$$

$$= \sum_{l\in[\ell]}\sum_{k\in[K_c]} \frac{\prod_{k'\in[K_c]\setminus\{k\}}\left(f_{l,k'}-\alpha_s\right)^{R'}}{\left(f_{l,k}-\alpha_s\right)^{R'}}\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k} + \underbrace{\sum_{l\in[\ell]}\sum_{\substack{k,k'\in[K_c]\\k\neq k'}}\left(\prod_{k''\in[K_c]\setminus\{k,k'\}}\left(f_{l,k''}-\alpha_s\right)^{R'}\right)\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k'}}_{\Gamma_1} + \Gamma_2+\Gamma_3+\Gamma_4+\widetilde{\mathbf{M}}_s. \tag{34}$$

Consider the first term in (34). For each $l\in[\ell], k\in[K_c]$, we have

$$\frac{\prod_{k'\in[K_c]\setminus\{k\}}\left(f_{l,k'}-\alpha_s\right)^{R'}}{\left(f_{l,k}-\alpha_s\right)^{R'}}\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k} = \frac{\prod_{k'\in[K_c]\setminus\{k\}}\left(\left(f_{l,k}-\alpha_s\right)+\left(f_{l,k'}-f_{l,k}\right)\right)^{R'}}{\left(f_{l,k}-\alpha_s\right)^{R'}}\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k} \tag{35}$$

$$= \frac{\Psi_{l,k}\left(f_{l,k}-\alpha_s\right)}{\left(f_{l,k}-\alpha_s\right)^{R'}}\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k} \tag{36}$$

$$= \left(\frac{c_{l,k,0}}{\left(f_{l,k}-\alpha_s\right)^{R'}}+\frac{c_{l,k,1}}{\left(f_{l,k}-\alpha_s\right)^{R'-1}}+\cdots+\frac{c_{l,k,R'-1}}{f_{l,k}-\alpha_s}\right)\times\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k} + \underbrace{\left(\sum_{i=R'}^{R'(K_c-1)}c_{l,k,i}\left(f_{l,k}-\alpha_s\right)^{i-R'}\right)\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k}}_{\Gamma_5}, \tag{37}$$

where (36) results from the definition of $\Psi_{l,k}(\cdot)$ as in (24) and in (37) the polynomial $\Psi_{l,k}(f_{l,k}-\alpha_s)$ is rewritten in terms of its coefficients.

By the construction of Entangled Polynomial code (25) (26), the product $\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k}$ can be written as weighted sums of the terms $1,(f_{l,k}-\alpha_s),\ldots,(f_{l,k}-\alpha_s)^{R'+p-2}$, i.e.,

$$\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k} = \sum_{i=0}^{R'+p-2}\mathbf{C}_{i+1}^{l,k}\left(f_{l,k}-\alpha_s\right)^i, \tag{38}$$

where $\mathbf{C}_1^{l,k},\mathbf{C}_2^{l,k},\ldots,\mathbf{C}_{R'+p-1}^{l,k}$ are various linear combinations of products of blocks of $\mathbf{A}^{l,k}$ and blocks of $\mathbf{B}^{l,k}$. Consider the first term in (37).

$$\left(\frac{c_{l,k,0}}{\left(f_{l,k}-\alpha_s\right)^{R'}}+\cdots+\frac{c_{l,k,R'-1}}{f_{l,k}-\alpha_s}\right)\mathbf{P}_s^{l,k}\mathbf{Q}_s^{l,k}$$

$$\overset{(38)}{=}\left(\frac{c_{l,k,0}}{\left(f_{l,k}-\alpha_s\right)^{R'}}+\cdots+\frac{c_{l,k,R'-1}}{f_{l,k}-\alpha_s}\right)\sum_{i=0}^{R'+p-2}\mathbf{C}_{i+1}^{l,k}\left(f_{l,k}-\alpha_s\right)^i \tag{39}$$

$$= \sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k,i-i'}\mathbf{C}_{i'+1}^{l,k}}{\left(f_{l,k}-\alpha_s\right)^{R'-i}} + \underbrace{\sum_{i=0}^{p-2}\left(f_{l,k}-\alpha_s\right)^i\left(\sum_{i'=i+1}^{R'+i'}c_{l,k,R'-i'+i}\mathbf{C}_{i'+1}^{l,k}\right)}_{\Gamma_6}$$

$$+ \underbrace{\sum_{i=p-1}^{R'+p-3}\left(f_{l,k}-\alpha_s\right)^i\left(\sum_{i'=i+1}^{R'+p-2}c_{l,k,R'-i'+i}\mathbf{C}_{i'+1}^{l,k}\right)}_{\Gamma_7}. \tag{40}$$

Note that if $K_c=1$, $\forall i\neq 0, c_{l,k,i}=0$, then Γ_5 and Γ_7 are zero polynomials. Now let us consider the degree with respect to α_s of $\Gamma_1,\ldots,\Gamma_7$.

$$deg_{\alpha_s}(\Gamma_1) = \begin{cases} R'(K_c-1)+p-2, & \text{if } K_c>1\\ -1, & \text{otherwise,}\end{cases}$$
$$deg_{\alpha_s}(\Gamma_2) = R'(K_c-1)+pm+X-2,$$
$$deg_{\alpha_s}(\Gamma_3) = R'(K_c-1)+pmn-pm+p+X-2,$$
$$deg_{\alpha_s}(\Gamma_4) = R'K_c+2X-2,\quad deg_{\alpha_s}(\Gamma_6)=p-2,$$
$$deg_{\alpha_s}(\Gamma_5) = \begin{cases} R'(K_c-1)+p-2, & \text{if } K_c>1\\ -1, & \text{otherwise,}\end{cases}$$
$$deg_{\alpha_s}(\Gamma_7) = \begin{cases} R'+p-3, & \text{if } K_c>1\\ -1, & \text{otherwise.}\end{cases}$$

Recall X,p,m,n,K_c are positive integers. If $K_c>1$, it is easy to see that $R'K_c+2X-2$ is the largest. If $K_c=1$, $R'=pmn\geq p>p-2$, $R'K_c+2X-2$ is also the largest. Therefore the sum of $\Gamma_1,\ldots,\Gamma_7$ can be expanded into weighted sums of the terms $1,\alpha_s,\ldots,\alpha_s^{R'K_c+2X-2}$. Note that the weights of terms $\alpha_s^{R'(K_c-1)+X+D_E+1},\ldots,\alpha_s^{R'K_c+2X-2}$ are functions of $\mathcal{Z}^A,\mathcal{Z}^B$. $\mathbf{Y}_s$ can be rewritten as

$$\mathbf{Y}_s = \sum_{l\in[\ell]}\sum_{k\in[K_c]}\sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k,i-i'}\mathbf{C}_{i'+1}^{l,k}}{\left(f_{l,k}-\alpha_s\right)^{R'-i}} + \sum_{x=1}^{R'K_c+2X-1}\alpha_s^{x-1}\mathbf{I}_x+\widetilde{\mathbf{M}}_s \tag{41}$$

$$\overset{(29)}{=} \sum_{l\in[\ell]}\sum_{k\in[K_c]}\sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k,i-i'}\mathbf{C}_{i'+1}^{l,k}}{\left(f_{l,k}-\alpha_s\right)^{R'-i}} + \sum_{x=1}^{R'K_c+2X-1}\alpha_s^{x-1}\mathbf{I}_x + \sum_{x=1}^{R'(K_c-1)+X+D_E}\alpha_s^{x-1}\mathbf{Z}'_x + \sum_{l\in[\ell]}\sum_{k\in[K_c]}\sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k.i-i'}\mathbf{Z}''_{l,k,i'+1}}{\left(f_{l,k}-\alpha_s\right)^{R'-i}} \tag{42}$$

$$= \sum_{l\in[\ell]}\sum_{k\in[K_c]}\sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k,i-i'}\left(\mathbf{C}_{i'+1}^{l,k}+\mathbf{Z}''_{l,k,i'+1}\right)}{\left(f_{l,k}-\alpha_s\right)^{R'-i}} + \sum_{x=1}^{R'(K_c-1)+X+D_E}\alpha_s^{x-1}\left(\mathbf{I}_x+\mathbf{Z}'_x\right) + \sum_{x=R'(K_c-1)+X+D_E+1}^{R'K_c+2X-1}\alpha_s^{x-1}\mathbf{I}_x \tag{43}$$

$$= \sum_{l\in[\ell]}\sum_{k\in[K_c]}\sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k,i-i'}\mathbf{D}_{i'+1}^{l,k}}{\left(f_{l,k}-\alpha_s\right)^{R'-i}} + \sum_{x\in[R'K_c+2X-1]}\alpha_s^{x-1}\mathbf{J}_x, \tag{44}$$

where $\mathbf{D}_i^{l,k}=\mathbf{C}_i^{l,k}+\mathbf{Z}''_{l,k,i}, l\in[\ell], k\in[K_c], i\in[R']$, $\mathbf{J}_x=\mathbf{I}_x+\mathbf{Z}'_x, x\in[R'(K_c-1)+X+D_E]$ and $\mathbf{J}_x=\mathbf{I}_x, x\in$

$[R'(K_c-1)+X+D_E+1 : R'K_c+2X-1]$. In the matrix form, answers from any $R = R'K_c+2X-1+R'L = pmn(\ell+1)K_c+2X-1$ servers, whose indices are denoted as $s_1, s_2, \ldots, s_R$, can be written as (45), shown at the bottom of the page. Since $f_{1,1}, f_{1,2}, \ldots, f_{\ell,K_c}$ are distinct, for all $l \in [\ell], k \in [K_c]$, $c_{l,k,0} = \prod_{k' \in [K_c]\setminus\{k\}}(f_{l,k'} - f_{l,k})^{R'}$ are non-zero. Hence, the lower triangular toeplitz matrices $\mathbf{T}(c_{1,1,0}, \ldots, c_{1,1,R'-1})$, , $\ldots$, $\mathbf{T}(c_{\ell,K_c,0}, \ldots, c_{\ell,K_c,R'-1})$ are non-singular, and the block diagonal matrix $\hat{\mathbf{V}}'_{\ell,K_c,R',X,R}$ is invertible. Guaranteed by Lemma 1 and the fact that the Kronecker product of non-singular matrices is non-singular, the matrix $(\hat{\mathbf{V}}_{\ell,K_c,R',X,R}\hat{\mathbf{V}}'_{\ell,K_c,R',X,R}) \otimes \mathbf{I}_{\lambda/m}$ is invertible. Therefore, the master is able to recover $(\mathbf{D}_i^{l,k})_{l\in[\ell],k\in[K_c],i\in[R']}$ by inverting the matrix. Note that $\mathbf{Z}''_{l,k,i} = \mathbf{0}, l \in [\ell], k \in [K_c], i \in \mathcal{E}$, therefore $(\mathbf{C}_i^{l,k})_{l\in[\ell],k\in[K_c],i\in\mathcal{E}} = (\mathbf{D}_i^{l,k})_{l\in[\ell],k\in[K_c],i\in\mathcal{E}}$. The desired products $(\mathbf{A}^{(l)}\mathbf{B}^{(l)})_{l\in[L]}$ are recoverable from $(\mathbf{C}_i^{l,k})_{l\in[\ell],k\in[K_c],i\in\mathcal{E}}$, guaranteed by the correctness of Entangled Polynomial code [5]. This completes the proof of recovery threshold $R = pmn(\ell+1)K_c + 2X - 1$.

Consider the strong security property. According to the construction, $\mathcal{M}_1 = 0$, $\mathcal{M}_s = \widetilde{\mathbf{M}}_s, s \in [S] \setminus \{1\}$, and $\mathcal{M} = \{\widetilde{\mathbf{M}}_s | s \in [S] \setminus \{1\}\}$. Since $\widetilde{\mathbf{M}}_s$ is a function of $\mathcal{Z}^{server}$,

$$I\Big(\mathbf{A}, \mathbf{B}, \widetilde{\mathbf{A}}^{[\mathcal{S}]}, \widetilde{\mathbf{B}}^{[\mathcal{S}]}; \mathcal{M}\Big) \le I\Big(\mathbf{A}, \mathbf{B}, \widetilde{\mathbf{A}}^{[\mathcal{S}]}, \widetilde{\mathbf{B}}^{[\mathcal{S}]}; \mathcal{Z}^{server}\Big) = 0.$$

Strong security is satisfied. Security is guaranteed because $\forall \mathcal{X} \subset [S], |\mathcal{X}| = X$,

$$\begin{aligned} &I\Big(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}^{\mathcal{X}}, \widetilde{\mathbf{B}}^{\mathcal{X}}, \mathcal{M}_{\mathcal{X}}\Big) \\ &= I(\mathbf{A}, \mathbf{B}; \mathcal{M}_{\mathcal{X}}) + I\Big(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}^{\mathcal{X}}, \widetilde{\mathbf{B}}^{\mathcal{X}} | \mathcal{M}_{\mathcal{X}}\Big) \qquad (46)\\ &= I(\mathbf{A}, \mathbf{B}; \mathcal{M}_{\mathcal{X}}) + I\Big(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}^{\mathcal{X}}, \widetilde{\mathbf{B}}^{\mathcal{X}}\Big) = 0, \qquad (47)\end{aligned}$$

where (47) is due to (27), (28) and the facts that each share is encoded with (X, S) Reed-Solomon code with uniformly and independently distributed noise.

Consider the privacy property,

$$\begin{aligned} &I(\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_S; \mathbf{A}, \mathbf{B} | \mathbf{AB})\\ &= I\Big(\Big(\mathbf{D}_i^{l,k}\Big)_{l\in[\ell],k\in[K_c],i\in[R']}, (\mathbf{J}_x)_{x\in[R'K_c+2X-1]}; \mathbf{A}, \mathbf{B} | \mathbf{AB}\Big) \qquad (48)\\ &= I\Big(\Big(\mathbf{D}_i^{l,k}\Big)_{l\in[\ell],k\in[K_c],i\in[R']}; \mathbf{A}, \mathbf{B} | \mathbf{AB}\Big)\\ &\quad + I\Big((\mathbf{J}_x)_{x\in[R'K_c+2X-1]}; \mathbf{A}, \mathbf{B} | \mathbf{AB}, \Big(\mathbf{D}_i^{l,k}\Big)_{l\in[\ell],k\in[K_c],i\in[R']}\Big) \qquad (49)\\ &= I\Big((\mathbf{J}_x)_{x\in[R'K_c+2X-1]}; \mathbf{A}, \mathbf{B} | \mathbf{AB}, \Big(\mathbf{D}_i^{l,k}\Big)_{l\in[\ell],k\in[K_c],i\in[R']}\Big) \qquad (50)\\ &\le I\Big((\mathbf{J}_x)_{x\in[R'K_c+2X-1]}; \mathbf{A}, \mathbf{B}, \mathbf{AB}, \Big(\mathbf{D}_i^{l,k}\Big)_{l\in[\ell],k\in[K_c],i\in[R']}\Big) \qquad (51)\\ &\le I\Big(\mathcal{Z}_1^{server}, \mathcal{Z}^A, \mathcal{Z}^B; \mathbf{A}, \mathbf{B}, \Big(\mathbf{D}_i^{l,k}\Big)_{l\in[\ell],k\in[K_c],i\in[R']}\Big) = 0, \qquad (52)\end{aligned}$$

where (48) holds because the mapping from $((\mathbf{D}_i^{l,k})_{l\in[\ell],k\in[K_c],i\in[R']}, (\mathbf{J}_x)_{x\in[R'K_c+2X-1]})$ to $(\mathbf{Y}_1, \ldots, \mathbf{Y}_S)$ is bijective. Equation (50) holds due to (28) and the fact $(\mathbf{C}_i^{l,k})_{l\in[\ell],k\in[K_c],i\in\mathcal{E}}$ are functions of $\mathbf{AB}$.

Consider the communication cost. The source upload cost $U_A = \frac{S}{K_c pm}$ and $U_B = \frac{S}{K_c pn}$. The server communication cost $CC = \frac{S-1}{\ell K_c mn}$. Note that the master is able to recover

$$\begin{bmatrix}\mathbf{Y}_{s_1}\\ \mathbf{Y}_{s_2}\\ \vdots\\ \mathbf{Y}_{s_R}\end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{(f_{1,1}-\alpha_{s_1})^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_{s_1}} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_{s_1})^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_1}} & 1 & \cdots & \alpha_{s_1}^{R'K_c+2X-2}\\ \frac{1}{(f_{1,1}-\alpha_{s_2})^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_{s_2}} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_{s_2})^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_2}} & 1 & \cdots & \alpha_{s_2}^{R'K_c+2X-2}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \frac{1}{(f_{1,1}-\alpha_{s_R})^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_{s_R}} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_{s_R})^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_R}} & 1 & \cdots & \alpha_{s_R}^{R'K_c+2X-2}\end{bmatrix}}_{\hat{\mathbf{V}}_{\ell,K_c,R',X,R}}$$
$$\times \underbrace{\begin{bmatrix}\mathbf{T}(c_{1,1,0}, \cdots, c_{1,1,R'-1}) & & \\ & \ddots & \\ & & \mathbf{T}(c_{\ell,K_c,0}, \cdots, c_{\ell,K_c,R'-1})\\ & & & \mathbf{I}_{R-R'L}\end{bmatrix}}_{\hat{\mathbf{V}}'_{\ell,K_c,R',X,R}} \otimes \mathbf{I}_{\lambda/m} \begin{bmatrix}\mathbf{D}_1^{1,1}\\ \vdots\\ \mathbf{D}_{R'}^{1,1}\\ \vdots\\ \mathbf{D}_1^{\ell,K_c}\\ \vdots\\ \mathbf{D}_{R'}^{\ell,K_c}\\ \mathbf{J}_1\\ \vdots\\ \mathbf{J}_{R'K_c+2X-1}\end{bmatrix} \qquad (45)$$

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. So by fast algorithms [49], the encoding complexity of $(\mathcal{C}_{eA}, \mathcal{C}_{eB}) = (\widetilde{\mathcal{O}}(\frac{\lambda\kappa S\log^2 S}{K_c pm}), \widetilde{\mathcal{O}}(\frac{\kappa\mu S\log^2 S}{K_c pn}))$ 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 $\mathcal{O}(\frac{\lambda\kappa\mu}{pmn})$ for a total of $\mathcal{O}(\frac{\ell\lambda\kappa\mu}{pmn})$. The complexity of summation over the products and noise is $\mathcal{O}(\frac{\ell\lambda\mu}{mn})$. To construct the noise, one server needs to encode the noise, whose complexity is $\widetilde{\mathcal{O}}(\frac{\lambda\mu S\log^2 S}{mn})$ by fast algorithms [49]. Normalized by the number of servers, it is $\widetilde{\mathcal{O}}(\frac{\lambda\mu\log^2 S}{mn})$. Considering these 3 procedures, upon normalization by $L = \ell K_c$, it yields a complexity of $\mathcal{O}(\frac{\lambda\kappa\mu}{K_c pmn}) + \mathcal{O}(\frac{\lambda\mu}{K_c mn}) + \widetilde{\mathcal{O}}(\frac{\lambda\mu\log^2 S}{\ell K_c mn})$ per server. The master decoding complexity is inherited from that of GCSA codes [40], which is at most $\widetilde{\mathcal{O}}(\lambda\mu p\log^2 R)$. This completes the proof of Theorem 1.

Remark: When $L = \ell = K_c = 1, S = R$, by setting $f_{1,1} = 0$, our construction of shares of $\widetilde{A}^s$ and $\widetilde{B}^s$ essentially recovers the construction of shares in [41].

VI. Discussion and Conclusion

In this article, the class of GCSA codes is expanded by including noise-alignment, so that the resulting GCSA-NA code is a solution for secure coded multi-party computation of massive matrix multiplication. For two sources and matrix multiplication, GCSA-NA strictly generalizes PS [41] and outperforms it in several key aspects. This construction also settles the asymptotic capacity of symmetric X-secure T-private information retrieval. The idea of noise-alignment can be applied to construct a scheme for N sources based on N-CSA codes, and be combined with Strassen's construction. As open problems, exploring the optimal amount of randomness and finding the communication efficient schemes for arbitrary polynomial are interesting directions.

Since Strassen's algorithm [45] is an important fast matrix multiplication approach, it is interesting to show noise alignment can be combined with it for secure multi-party matrix multiplication. Consider an example with two 2×2 block matrices $\mathbf{A}, \mathbf{B}$ and $X = 1$. It can be shown that the general recursive Strassen's algorithm also works similarly. The desired product is denoted by $\mathbf{C} = \begin{bmatrix}\mathbf{C}_{1,1} & \mathbf{C}_{1,2}\\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2}\end{bmatrix}$. The Strassen's constuction constructs 14 matrices $\mathbf{P}_i, \mathbf{Q}_i, i \in [7]$ ($\mathbf{P}_i$ only depends on $\mathbf{A}$ and $\mathbf{Q}_i$ only depends on $\mathbf{B}$) and

$$\begin{bmatrix}\mathbf{C}_{1,1}\\ \mathbf{C}_{1,2}\\ \mathbf{C}_{2,1}\\ \mathbf{C}_{2,2}\end{bmatrix} = \begin{bmatrix}0 & -1 & 0 & 1 & 1 & 1 & 0\\ 1 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 1 & 0 & 0 & 0\\ 1 & 0 & -1 & 0 & 1 & 0 & -1\end{bmatrix}\begin{bmatrix}\mathbf{P}_1\mathbf{Q}_1\\ \mathbf{P}_2\mathbf{Q}_2\\ \vdots\\ \mathbf{P}_7\mathbf{Q}_7\end{bmatrix}. \quad (53)$$

This is the basic Strassen algorithm. Now let us see how we apply CSA and noise alignment to it. Each share is constructed based on CSA code principles with noise, i.e., $\widetilde{\mathbf{A}} = \Delta(\sum_{i\in[7]}\frac{\mathbf{P}_i}{f_i-\alpha} + \mathbf{Z}^A), \widetilde{\mathbf{B}} = \sum_{i\in[7]}\frac{\mathbf{Q}_i}{f_i-\alpha} + \mathbf{Z}^B, \widetilde{\mathbf{A}}\widetilde{\mathbf{B}} = \sum_{i\in[7]}\frac{c_i}{f_i-\alpha}\mathbf{P}_i\mathbf{Q}_i + \sum_{i=0}^{7}\alpha^i\mathbf{I}_i$.

If the servers directly return $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}$ to the master, additional information about the input may be leaked due to interference terms $\mathbf{P}_1\mathbf{Q}_1, \ldots, \mathbf{P}_7\mathbf{Q}_7$ and $\sum_{i=0}^{6}\alpha^i I_i$. We secure the scheme by the addition of noise. The idea is that we want the master to decode $\mathbf{T}_1, \ldots, \mathbf{T}_7$ instead of $\mathbf{P}_1\mathbf{Q}_1, \ldots, \mathbf{P}_7\mathbf{Q}_7$, such that

$$H(\mathbf{C}|\mathbf{T}_1, \ldots, \mathbf{T}_7) = I(\mathbf{A}, \mathbf{B}; \mathbf{T}_1, \ldots, \mathbf{T}_7|\mathbf{C}) = 0. \quad (54)$$

$\mathbf{T}_1, \ldots, \mathbf{T}_v$ are constructed as follows.

$$\mathbf{T}_1 = \mathbf{P}_1\mathbf{Q}_1 - \mathbf{Z}_1 - \mathbf{Z}_2 + \mathbf{Z}_3, \quad \mathbf{T}_2 = \mathbf{P}_2\mathbf{Q}_2 - \mathbf{Z}_1 + \mathbf{Z}_2 - \mathbf{Z}_3,$$
$$\mathbf{T}_3 = \mathbf{P}_3\mathbf{Q}_3 - \mathbf{Z}_1, \quad \mathbf{T}_4 = \mathbf{P}_4\mathbf{Q}_4 + \mathbf{Z}_1, \quad \mathbf{T}_5 = \mathbf{P}_5\mathbf{Q}_5 + \mathbf{Z}_2,$$
$$\mathbf{T}_6 = \mathbf{P}_6\mathbf{Q}_6 - \mathbf{Z}_3, \quad \mathbf{T}_7 = \mathbf{P}_7\mathbf{Q}_7 + \mathbf{Z}_3,$$

where $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3$ are i.i.d. uniform noise matrices. To align the noise, we construct $\widetilde{\mathbf{Z}}$,

$$\begin{aligned}\widetilde{\mathbf{Z}} = &\left(-\frac{c_1}{f_1-\alpha} - \frac{c_2}{f_2-\alpha} - \frac{c_3}{f_3-\alpha} + \frac{c_4}{f_4-\alpha}\right)\mathbf{Z}_1\\ &+ \left(-\frac{c_1}{f_1-\alpha} + \frac{c_2}{f_2-\alpha} + \frac{c_5}{f_5-\alpha}\right)\mathbf{Z}_2\\ &+ \left(\frac{c_1}{f_1-\alpha} - \frac{c_2}{f_2-\alpha} - \frac{c_6}{f_6-\alpha} + \frac{c_7}{f_7-\alpha}\right)\mathbf{Z}_3\\ &+ \sum_{i=0}^{6}\alpha^i\mathbf{Z}_{i+4},\end{aligned}$$

where $\mathbf{Z}_4, \ldots, \mathbf{Z}_{10}$ are i.i.d. uniform noise matrices. The answer returned by each server to the master is $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}} + \widetilde{\mathbf{Z}}$. The correctness and privacy are easily proved.

References

[1] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[2] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," 2017. [Online]. Available: arXiv:1705.10464.

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

[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," Nov. 2018. [Online]. Available: ArXiv:1811.10751.

[5] 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, Mar. 2020.

[6] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and A. S. Avestimehr, "Lagrange coded computing: Optimal Design for resiliency, security and privacy," 2018. [Online]. Available: ArXiv:1806.00939.

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

[8] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

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

[10] 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.*, 2016, pp. 2100–2108.

[11] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," 2017. [Online]. Available: arXiv:1705.03875.

[12] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded fourier transform," 2017. [Online]. Available: arXiv:1710.06471.

[13] T. Jahani-Nezhad and M. A. Maddah-Ali, "CodedSketch: A coding scheme for distributed computation of approximated matrix multiplications," 2018. [Online]. Available: arXiv:1812.10460.

[14] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1993–1997.

[15] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *Proc. 55th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2017, pp. 1271–1278.

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

[17] 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, p. 58, 2019.

[18] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," 2018. [Online]. Available: arXiv:1802.03430.

[19] A. Severinson, A. G. I. Amat, and E. Rosnes, "Block-diagonal and LT codes for distributed computing with straggling servers," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 1739–1753, Mar. 2019.

[20] F. Haddadpour and V. R. Cadambe, "Codes for distributed finite alphabet matrix-vector multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1625–1629.

[21] U. Sheth *et al.*, "An application of storage-optimal matdot codes for coded matrix multiplication: Fast K-nearest neighbors estimation," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2018, pp. 1113–1120.

[22] H. Jeong, F. Ye, and P. Grover, "Locally recoverable coded matrix multiplication," in *Proc. 56th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2018, pp. 715–722.

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

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

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

[26] W. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[27] J. Kakar, S. Ebadifar, and A. Sezgin, "On the capacity and straggler-robustness of distributed secure matrix multiplication," *IEEE Access*, vol. 7, pp. 45783–45799, 2019.

[28] 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, Jul. 2020, doi: 10.1109/TIT.2020.2975021.

[29] M. Kim and J. Lee, "Private secure coded computation," *IEEE Commun. Lett.*, vol. 23, no. 11, pp. 1918–1921, Nov. 2019.

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

[31] H. Sun and S. A. Jafar, "The capacity of private information retrieval," *IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4075–4088, Jul. 2017.

[32] H. Sun and S. A. Jafar, "The capacity of robust private information retrieval with colluding databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2361–2370, Apr. 2018.

[33] K. Banawan and S. Ulukus, "The capacity of private information retrieval from coded databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1945–1956, Mar. 2018.

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

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

[36] Q. Wang and M. Skoglund, "Secure symmetric private information retrieval from colluding databases with adversaries," in *Proc. 55th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2017, pp. 1083–1090.

[37] Z. Jia, H. Sun, and S. A. Jafar, "Cross subspace alignment and the asymptotic capacity of X-secure T-private information retrieval," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5783–5798, Sep. 2019.

[38] Z. Jia and S. A. Jafar, "On the asymptotic capacity of X-secure T-private information retrieval with graph based replicated storage," 2019. [Online]. Available: ArXiv:1904.05906.

[39] Z. Jia and S. A. Jafar, "X-secure T-private information retrieval from MDS coded storage with byzantine and unresponsive servers," 2019. [Online]. Available: ArXiv:1908.10854.

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

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

[42] A. C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.

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

[44] R. G. L. D'Oliveira, S. E. Rouayheb, D. Heinlein, and D. Karpuk, "Notes on communication and computation in secure distributed matrix multiplication," 2020. [Online]. Available: arXiv:2001.05568.

[45] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[46] H. A. Nodehi and M. A. Maddah-Ali, "Limited-sharing multi-party computation for massive matrix operations," in *Proc. IEEE Int. Symp. Inf. Theory*, 2018, pp. 1231–1235.

[47] H. A. Nodehi, S. R. H. Najarkolaei, and M. A. Maddah-Ali, "Entangled polynomial coding in limited-sharing multi-party computation," in *Proc. IEEE Inf. Theory Workshop*, 2018, pp. 1–5.

[48] M. Gasca, J. Martinez, and G. Mühlbach, "Computation of rational interpolants with prescribed poles," *J. Comput. Appl. Math.*, vol. 26, no. 3, pp. 297–309, 1989.

[49] V. Olshevsky and A. Shokrollahi, "A superfast algorithm for confluent rational tangential interpolation problem via matrix-vector multiplication for confluent cauchy-like matrices," in *Structured Matrices in Mathematics, Computer Science, and Engineering I* (Contemporary Mathematics), vol. 280, Amer. Math. Soc., 2001, pp. 32–46.