UNIVERSITY OF CALIFORNIA,
IRVINE

Cross Subspace Alignment and Its Applications to Private Information Retrieval and Coded Distributed Computation

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Zhuqing Jia

Dissertation Committee:
Chancellor's Professor Syed Ali Jafar, Chair
Assistant Professor Zhiying Wang
Assistant Professor Yanning Shen

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Syed Jafar. I have benefited tremendously from his guidance and countless discussions with him. Without his insights and immense knowledge, this dissertation would not have been possible. His motivation and enthusiasm in regard to research and scholarship continually encourage me to advance on the road of academic.

I would like to thank Professor Zhiying Wang and Professor Yanning Shen for serving on my dissertation committee, and Professor Ender Ayanoglu, and Professor Zhaoxia Yu for serving on my qualifying examination committee. I appreciate all these valuable suggestions from my committee members.

Let me also express my gratitude to Hua Sun, Zhen Chen and Yuxiang Lu for collaborating and co-authoring papers with me. I am also grateful to have worked with my former and current colleagues including Arash Gholami Davoodi, Bofeng Yuan, Yao-Chia Chan, Junge Wang and Nilab İsmailoğlu. It is a great honor to work with all of you.

Last but not the least, let me express my gratefulness to my parents, my family and my friends. It is my pleasure to dedicate this dissertation to them.

VITA

Zhuqing Jia

EDUCATION

Doctor of Philosophy in Electrical Engineering **2021**
University of California, Irvine *Irvine, California*

Master of Science in Electrical Engineering **2019**
University of California, Irvine *Irvine, California*

Bachelor of Science in Electronic Information Engineering **2015**
Beijing University of Posts and Telecommunications *Beijing, China*

RESEARCH EXPERIENCE

Graduate Student Researcher **2016–2021**
University of California, Irvine *Irvine, California*

ABSTRACT OF THE DISSERTATION

Cross Subspace Alignment and Its Applications to Private Information Retrieval and Coded Distributed Computation

By

Zhuqing Jia

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2021

Chancellor's Professor Syed Ali Jafar, Chair

Originating from the construction of the asymptotic-capacity achieving scheme for X-secure T-private information retrieval (XSTPIR), the technique of cross-subspace alignment (CSA) emerges as the natural solution to secure and private information retrieval, secure distributed matrix multiplication, and coded distributed batch computation. Characterized by a Cauchy-Vandermonde structure that facilitates interference alignment along Vandermonde terms, while the desired signals remain resolvable along the Cauchy terms, the idea of CSA is shown to be the essential ingredient in the optimal/asymptotically optimal/state-of-art approaches that minimize the download and/or communication cost of these independently introduced but closely related problems.

In this dissertation we will first introduce the idea of CSA to the applications of XSTPIR, XSTPIR with graph-based replicated storage (GXSTPIR) and XSTPIR with MDS coded storage (MDS-XSTPIR). The CSA solution to XSTPIR exploits the Cauchy-Vandermonde structured answer strings that align interference symbols guaranteeing T-privacy and X-security to achieve the asymptotic capacity. The achievability scheme for GXSTPIR reveals a non-trivial generalization of CSA that takes the advantage of a special structure inspired by dual Generalized Reed Solomon (GRS) codes to allow interference alignment for arbitrary storage patterns. For MDS-XSTPIR, we propose a novel scheme based on *confluent* Cauchy-Vandermonde storage structure and a strategy of successive decoding with interference cancellation. Next, by characterizing a connection between a form of XSTPIR problem known as multi-message XSTPIR and the problem of secure distributed matrix multiplication (SDMM), we characterize a series of capacity regions of SDMM, as well as several of its variants. The idea of CSA serves as an essential component of the construction of several achievability schemes. Given the insights from all these results, next, we construct CSA/GCSA codes based on (*confluent*) Cauchy-Vandermonde structures for coded distributed batch computation (CDBC) that unify, generalize and improve upon the state-of-art codes for distributed computing such as LCC codes for multivariate polynomial evaluations and EP codes for matrix multiplication. Finally, we study the problem of X-secure T-private federated submodel learning (XSTPFSL), which is a non-trivial generalization of the XSTPIR problem where private writes are needed. The proposed ACSA-RW scheme achieves the desired private read and write functionality with elastic dropout resilience, takes the advantage of the synergistic gain from the joint design of private read and write for low communication costs, and sheds light on a striking symmetry between upload and download costs. Intuitively, private read and write functionalities, as well as their synergistic gain, rely on secure distributed matrix multiplication so that the idea of CSA emerges as the core of the solution.

Chapter 1

Introduction

1.1 Background

The era of big data, machine learning, cloud computing, and massive parallelization is marked by the rise of fundamental tradeoffs between competing concerns such as privacy, security, robustness, and efficiency. These concerns have been extensively studied independently, for example, the problem of (information-theoretic) private information retrieval (PIR)[102] seeks for a solution that efficiently retrieves one out of K messages stored in a distributed storage system with N servers such that any server can learn nothing about the identity of the desired message. Coded distributed computing (CDC)[68, 138] as another example, is to design distributed computation approaches that are resilient to stragglers through coding techniques. Although a large body of literature devotes to the solutions and the characterization of fundamental limits of these closely related problems (see the exhaustive list of reference for PIR [102, 106, 103, 120, 117, 12, 109, 119, 111, 116, 36, 105, 54, 143, 130, 57, 7, 128, 4, 127, 112, 71, 126, 22, 104, 134, 9, 97, 123, 115, 107, 81, 21, 53, 75, 52, 50] and CDC [140, 31, 28, 141, 138, 91, 69, 68, 29, 30, 139, 48, 6, 100, 122, 77, 121, 94, 44, 96, 49, 65, 86, 72]), there is a shortage of an unified understanding of the fundamental tradeoffs and a concentrated methodology that combines all.

The central of this dissertation is an interference alignment technique/a coding design named *cross-subspace alignment* (CSA) that tries to characterize the fundamental tradeoffs between the aforementioned concerns. Originating from the construction of the asymptotic-capacity achieving scheme of X-secure T-private information retrieval (XSTPIR), CSA is shown to be the essential component of the optimal/asymptotic optimal/state-of-art approaches that minimize the download and/or communication cost of independently introduced but closely related problems including XSTPIR, XSTPIR with graph-based replicated storage (GXSTPIR), XSTPIR with MDS coded storage (MDS-XSTPIR), secure distributed matrix multiplication (SDMM), coded distributed batch computation (CDBC) and X-secure T-private federated submodel learning (XSTPFSL). In this dissertation, we will introduce the idea of CSA through the solutions to the aforementioned problems and finally build up a unified understanding of the fundamental tradeoffs between privacy, security, robustness, and efficiency through the lens of CSA.

1.2 Overview of the Dissertation

We start with Chapter 2, which devotes to characterizing the asymptotic capacity of XSTPIR. In particular, the asymptotic capacity (i.e., the capacity in the limit as the number of messages K approaches infinity) of XSTPIR is characterized and the CSA-based solution emerges as the asymptotic capacity-achieving scheme. The basic element of CSA, i.e., a Cauchy-Vandermonde structure that facilitates interference alignment along Vandermonde terms, while the desired signals remain resolvable along the Cauchy terms, is also introduced. In Chapter 3, we consider the problem of GXSTPIR where instead of *fully* replicated storage, i.e., every message is replicated at every server, we assume that each message is replicated only among a subset of servers. A solution based on CSA and specialized storage and query construction inspired by the structure of dual of GRS codes shows that the PIR rate of $(\rho_{\min}-X-T)/N$ is achievable where N is the number of servers and each message is replicated at least $\rho_{\min}$ times. This is our first non-trivial generalization of CSA techniques. Next in Chapter 4, we study the problem of MDS-XSTPIR where the data is stored among the N servers according to an X-secured MDS code with the storage overhead N/K_c. We present a scheme built upon a *confluent* Cauchy-Vandermonde storage structure and a successive decoding and interference cancellation strategy that achieves the conjectured asymptotic capacity.

Given the above results, in Chapter 5 we characterize a connection between a special form of secure and private information retrieval known as *multi-message* XSTPIR and the problem of SDMM which enables us to obtain a series of capacity results of the latter. In Chapter 6, we further seek the application of the idea of CSA to coded distributed batch computation and construct CSA/GCSA ("G" stands for "generalized") codes that unify, generalize and improve upon the state-of-art codes for distributed computing such as LCC codes for multivariate polynomial evaluations and EP codes for matrix multiplication. It is worth noting that Cauchy-Vandermonde structure, as well as confluent Cauchy-Vandermonde structure, is essential in these codes.

In Chapter 7, we study the problem of XSTPFSL, which is a non-trivial generalization of the XSTPIR problem where private writes are needed. The proposed ACSA-RW scheme achieves the desired private read and write functionality with elastic dropout resilience, improves significantly upon available baselines for private-write, and reveals a striking symmetry between upload and download costs. Finally, we conclude this dissertation.

1.3 Notation

Bold symbols are used to denote vectors and matrices, while calligraphic symbols denote sets. By convention, let the empty product be the multiplicative identity, and the empty sum be the additive identity. For two positive integers M,N such that $M \le N$, $[M:N]$ denotes the set $\{M,M+1,\cdots,N\}$. We use the shorthand notation $[N]$ for $[1:N]$. $\mathbb{N}$ denotes the set of positive integers $\{1,2,3,\cdots\}$, and $\mathbb{Z}^*$ denotes the set $\mathbb{N}\cup\{0\}$. For sake of simplicity, let $X_{[m:n]}$ denote the set of random variables $\{X_m,X_{m+1},\ldots,X_n\}$. For an index set $\mathcal{I}=\{i_1,i_2,\ldots,i_n\}$, let $X_{\mathcal{I}}$ denote the set $\{X_{i_1},X_{i_2},\ldots,X_{i_n}\}$. For variables $a_n, n \in [1:N]$ and an arbitrary function $f(\cdot)$, we denote the $N \times 1$ vector whose n^{th} term is $f(a_n)$, as $\overrightarrow{f(a)}$. Similarly, $\overrightarrow{g(b)}$ denotes the vector $(g(b_1),\cdots,g(b_n))^T$ for variables $b_n, n \in [1:N]$ and a function $g(\cdot)$. For such $N \times 1$ vectors $\overrightarrow{f(a)}$ and $\overrightarrow{g(b)}$, let $\overrightarrow{f(a)} \circ \overrightarrow{g(b)}$ denote their Hadamard product, i.e., the $N \times 1$ vector whose n^{th} term is $f(a_n) \times g(b_n)$. The notation $X \sim Y$ is used to indicate that X and Y are identically distributed. Mutual informations between sets of random variables are similarly defined. For tuples such as $A=(a_1,a_2,\cdots,a_n)$ we allow set theoretic notions of inclusion. For example, $b \in A$ denotes the relationship $b \in \{a_1,a_2,\cdots,a_n\}$. Similarly, $b \in A \setminus \{a_1\}$ denotes $b \in \{a_2,a_3,\cdots,a_n\}$. For a subset of integers $\mathcal{N} \subset \mathbb{N}$, $\mathcal{N}(i), i \in [|\mathcal{N}|]$ denotes its i^{th} element, sorted in ascending order. The notation $\text{diag}(\mathbf{D}_1,\mathbf{D}_2,\cdots,\mathbf{D}_n)$ denotes the block diagonal matrix, i.e., the main-diagonal blocks are square matrices $(\mathbf{D}_1,\mathbf{D}_2,\cdots,\mathbf{D}_n)$ and all off-diagonal blocks are zero matrices. For a positive integer K, $\mathbf{I}_K$ denotes the $K \times K$ identity matrix. For two positive integers k,K such that $k \le K$, $\mathbf{e}_K(k)$ denotes the k^{th} column of the $K \times K$ identity matrix. The notation $\otimes$ is used to denote the Kronecker product of two matrices, i.e., for two matrices $\mathbf{A}$ and $\mathbf{B}$, where $(\mathbf{A})_{r,s}=a_{rs}$ and $(\mathbf{B})_{v,w}=b_{vw}$, $(\mathbf{A}\otimes\mathbf{B})_{p(r-1)+v,q(s-1)+w}=a_{rs}b_{vw}$. $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. $\mathbf{T}(X_1,X_2,\cdots,X_N)$ denotes the $N \times N$ lower triangular

Toeplitz matrix, i.e.,

$$\mathbf{T}(X_1,X_2,\cdots,X_N)=\begin{bmatrix} X_1 & & & & & \\ X_2 & X_1 & & & & \\ X_3 & X_2 & \ddots & & & \\ \vdots & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & X_2 & X_1 & \\ X_N & \cdots & \cdots & X_3 & X_2 & X_1 \end{bmatrix}. \tag{1.1}$$

The notation $\widetilde{\mathcal{O}}(a\log^2 b)$ suppresses[1] polylog terms. It may be replaced with $\mathcal{O}(a\log^2 b)$ if the field supports the Fast Fourier Transform (FFT), and with $\mathcal{O}(a\log^2 b\log\log(b))$ if it does not[2].

[1] There is another standard definition of the notation $\widetilde{\mathcal{O}}$ which fully suppresses polylog terms, i.e, $\mathcal{O}(a\text{polylog}(b))$ is represented by $\widetilde{\mathcal{O}}(a)$, regardless of the exact form of polylog(b). The definition used in this chapter emphasizes the dominant factor in the polylog term.

[2] If the FFT is not supported by the field, Schönhage–Strassen algorithm[92] can be used for fast algorithms that require convolutions, with an extra factor of $\log\log b$ in the complexity.

Chapter 2

X-Secure T-Private Information Retrieval

X-secure and T-private information retrieval (XSTPIR) is a form of private information retrieval where data security is guaranteed against collusion among up to X servers and the user's privacy is guaranteed against collusion among up to T servers. The capacity of XSTPIR is characterized for arbitrary number of servers N, and arbitrary security and privacy thresholds X and T, in the limit as the number of messages $K \to \infty$. Capacity is also characterized for any number of messages if either $N = 3, X = T = 1$ or if $N \leq X + T$. Insights are drawn from these results, about aligning versus decoding noise, dependence of PIR rate on field size, and robustness to symmetric security constraints. In particular, the idea of cross subspace alignment, i.e., introducing a subspace dependence between Reed-Solomon code parameters, emerges as the optimal way to align undesired terms while keeping desired terms resolvable.

2.1 Introduction

Motivated by the importance of security and privacy in the era of big data and distributed storage, in this chapter we explore the information theoretic capacity of private information retrieval (PIR) in a secure distributed storage system. Specifically, our focus in this chapter is on the X-secure and T-private information retrieval problem (XSTPIR). A PIR scheme is said to be T-private if it allows a user to retrieve a desired message from a database of K messages stored at N distributed servers, without revealing any information about the identity of the desired message to any group of up to T colluding servers. Similarly, a distributed storage scheme is said to be X-secure[1] if it guarantees that any group of up to X colluding servers learn nothing about the stored data. The T and X parameters may be chosen arbitrarily depending on the relative importance of security and privacy for any given application.

The rate of a PIR scheme is the ratio of the number of bits retrieved by the user to the total number of bits downloaded from all servers. The supremum of achievable rates is called the capacity of PIR. The capacity of the basic PIR setting was found in [102] to be

$$C_{\text{PIR}}(N,K)=(1+1/N+1/N^2+\cdots+1/N^{K-1})^{-1}. \tag{2.1}$$

The result was generalized subsequently in [106] to the T-PIR setting, as

$$C_{\text{TPIR}}(N,K,T)=\begin{cases}\left(1+T/N+T^2/N^2+\cdots+T^{K-1}/N^{K-1}\right)^{-1}, & T<N\\ 1/K, & T\geq N.\end{cases} \tag{2.2}$$

Further generalizations of T-privacy, e.g., when privacy is required only against certain specified collusion patterns [110, 56] have also been explored. In particular, capacity is

[1] In other words, everything that is stored at any X servers must be independent of the K messages. Besides X-security, no other constraints are imposed on the storage. The storage and the PIR scheme are jointly optimized to maximize the capacity of XSTPIR.

known for disjoint colluding sets [56].

The rapidly growing body of literature in this area has produced capacity results for PIR under a rich variety of constraints [102, 106, 103, 120, 117, 12, 109, 119, 111, 116, 36, 105, 54, 143, 130, 57, 7, 128, 4, 127, 112, 71, 126, 22, 104, 134, 9, 97, 123, 115, 107, 81, 21, 53, 75, 52, 50]. However, the capacity for the natural setting of secure storage remains unknown, and relatively unexplored. While a number of efforts are motivated by security concerns, such efforts have focused largely on other models, e.g., wiretap models where data security is desired against eavesdroppers listening to the communication between the user and the servers [14, 120], Byzantine models where the servers may respond incorrectly by introducing erasures or errors in their response to the user's queries [8, 142, 109, 119, 135], and so called symmetric security models [103, 116, 118] that allow the user to learn nothing about the data besides his desired message. An exception in this regard is the recent work in [130] where PIR with distributed storage is explored and the asymptotic (large K) capacity for the $X=T=1$ setting is bounded as

$$\left(1-\frac{1}{\sqrt{N}}\right)^2 \leq \lim_{K\rightarrow\infty} C_{\text{XSTPIR}}(N,K,X=1,T=1) \leq \left(1-\frac{1}{N}\right). \tag{2.3}$$

As the main result of this chapter, we close this gap and characterize the asymptotic capacity of XSTPIR for all N,X,T as follows.

$$\lim_{K\rightarrow\infty} C_{\text{XSTPIR}}(N,K,X,T) = \begin{cases} 1-\left(\frac{X+T}{N}\right), & N>X+T \\ 0, & N\leq X+T. \end{cases} \tag{2.4}$$

The asymptotic capacity characterization leads us to supplementary results which include a general upper bound on the capacity of XSTPIR, the exact capacity characterization for any number of messages K if $N\leq X+T$, and the exact capacity characterization for any K if $X=T=1, N=3$. The results also lead us to interesting observations about aligning versus decoding noise, dependence of PIR rate on field size, robustness to symmetric security constraints, and a particularly useful idea called cross subspace alignment. When privately retrieving multiple symbols from a desired message in a secure distributed storage system, the structure (say, $1,\beta,\beta^2,\cdots$, for one symbol and $1,\gamma,\gamma^2,\cdots$ for another, as in Reed-Solomon (RS) codes) of storage and queries for each symbol determines the number of dimensions occupied by interference and the resolvability of desired symbols. Choosing identical RS codes ($\beta=\gamma$) for each symbol of the same message would cause desired signals to align among themselves, while making the RS codes insufficiently dependent would cause interference to occupy too many dimensions. Cross subspace alignment is achieved by drawing the code parameters as linear combinations from the same subspace (say, $\beta=1-\alpha,\gamma=2-\alpha$), which turns out to be the optimal way to align interference while keeping desired symbols resolvable. For a summary of results and a better explanation of the main observations we refer the reader directly to Section 2.3.

2.2 Problem Statement: XSTPIR

Consider data that is stored at N distributed servers. The data consists of K independent messages, $W_1,W_2,\cdots,W_K$, and each message is represented[2] by L random symbols from the finite field $\mathbb{F}_q$.

$$H(W_1)=H(W_2)=\cdots=H(W_K)=L, \tag{2.5}$$

$$H(W_1,W_2,\ldots,W_K)=KL, \tag{2.6}$$

[2] As usual for an information theoretic formulation, the actual size of each message is allowed to approach infinity. The parameters L and q partition the data into blocks and may be chosen freely by the coding scheme to match the code dimensions. Since the coding scheme for a block can be repeated for each successive block of data with no impact on rate, it suffices to consider one block of data subject to optimization over L and q.

in q-ary units. There are N servers. The information stored at the n^{th} server is denoted by $S_n, n \in [N]$. An X-secure scheme, $0 \leq X < N$, guarantees that any X (or fewer) colluding servers learn nothing about the data.

$$[X\text{-Security}] \qquad I(S_{\mathcal{X}}; W_1, \ldots, W_K) = 0, \qquad \forall \mathcal{X} \subset [N], |\mathcal{X}| = X. \tag{2.7}$$

Besides X-security, we place no other constraint[3] on the amount of storage or the storage code used at each server, all of which is jointly optimized to maximize the capacity of XSTPIR. To ensure information retrieval is possible, note that the set of messages $W_1, \cdots, W_K$ must be a function of $S_{[N]}$.

$$H(W_1, \cdots, W_K \mid S_{[N]}) = 0. \tag{2.8}$$

The user generates a desired message index θ privately and uniformly from $[K]$. In order to retrieve W_θ privately, the user generates N queries, $Q_1^{[\theta]}, Q_2^{[\theta]}, \ldots, Q_N^{[\theta]}$. The query $Q_n^{[\theta]}$ is sent to the n^{th} server. The user has no prior knowledge of the information stored at the servers, i.e.,

$$I(S_{[N]}; Q_{[N]}^{[\theta]}, \theta) = 0. \tag{2.9}$$

T-privacy, $1 \leq T \leq N$, guarantees that any T (or fewer) colluding servers learn nothing about θ.

$$[T\text{-Privacy}] \qquad I(Q_{\mathcal{T}}^{[\theta]}, S_{\mathcal{T}}; \theta) = 0, \qquad \forall \mathcal{T} \subset [N], |\mathcal{T}| = T. \tag{2.10}$$

[3] The amount of storage at each server is not constrained *a priori*, however, it is remarkable that none of the XSTPIR schemes in this chapter end up storing more than KL symbols at each server. Thus the amount of storage used is not worse than a data replication scheme in the absence of security constraints.

Upon receiving the query $Q_n^{[\theta]}$, the n^{th} server generates an answering string $A_n^{[\theta]}$, as a function of the query $Q_n^{[\theta]}$ and its stored information S_n.

$$H(A_n^{[\theta]}|Q_n^{[\theta]},S_n)=0. \tag{2.11}$$

From all the answers the user must be able to recover the desired message W_θ,

$$\text{[Correctness]} \qquad H(W_\theta|A_{[N]}^{[\theta]},Q_{[N]}^{[\theta]},\theta)=0. \tag{2.12}$$

The rate of an XSTPIR scheme characterizes how many bits of desired message are retrieved per downloaded bit, (equivalently, how many q-ary symbols of desired message are retrieved per downloaded q-ary symbol),

$$R=\frac{L}{D}, \tag{2.13}$$

where D is the expected value (with respect to the random queries) of the number of q-ary symbols downloaded by the user from all servers. The capacity of XSTPIR, denoted $C_{\text{XSTPIR}}(N,K,X,T)$, is the supremum of achievable rates.

Finally, note that setting $X=0$ and $T=1$ reduces the XSTPIR problem to the basic PIR setting where data storage is not secure and the user's privacy is only guaranteed if no collusion takes place among servers. Setting $X=0$ for arbitrary T, reduces XSTPIR to the T-PIR problem. Setting $T=0$ for arbitrary X reduces XSTPIR to an X-secure storage scheme with no privacy constraint.

2.3 Capacity of XSTPIR: Results and Observations

The results of this chapter are presented in this section, followed by some observations.

2.3.1 Results

Our first result, presented in the following theorem, is an upper bound on the capacity of XSTPIR.

Theorem 2.1.

$$C_{XSTPIR}(N,K,X,T) \leq \left(\frac{N-X}{N}\right) C_{TPIR}(N-X,K,T). \tag{2.14}$$

The proof of Theorem 2.1 appears in Section 2.4. The intuition behind Theorem 2.1 may be understood through a thought experiment as follows. Without loss of generality, suppose the expected number of bits downloaded from each server is the same. Now, relax the constraints so that $S_{[X]}$, i.e., the stored information at the first X servers is made available globally (to all servers and to the user) for free, the messages $W_1,W_2,\cdots,W_K$ are made available to all servers, and the data-security constraint is eliminated. None of this can hurt capacity because any XSTPIR scheme from before can still be used with the relaxed constraints. So any upper bound on capacity of this relaxed setting is still an upper bound on the capacity of the original XSTPIR setting. The relaxed setting is analogous to the T-PIR problem with K messages and $N-X$ servers, for which we already know the optimal download per server from the existing capacity results for T-PIR. Thus, the statement of Theorem 2.1 follows. However, formalizing this intuition into a proof is not trivial because of the correlated side-information generated at the user and servers in the process of relaxing the constraints. Indeed, the formal proof presented in Section 2.4 takes a less direct approach.

It turns out the bound in Theorem 2.1 is quite powerful. In fact, we suspect that this bound might be tight in general. An immediate observation is that if we set $X=0$, i.e., remove the data storage security constraint, then the bound is tight because it gives us the capacity of T-PIR. Similarly, if we set $T=0$, i.e., the privacy constraint is removed, then the bound is also tight, and the capacity in the absence of privacy constraints is easily seen to be $C_{\text{XSTPIR}}(N,K,X,T=0)=1-\frac{X}{N}$, which is achievable by a simple secret-sharing scheme. We further prove the tightness of this bound for the cases identified in our next set of results. The first setting identifies a somewhat degenerate extreme where it is optimal to download everything.

Theorem 2.2. *If $N\leq X+T$, then[4] for arbitrary K,*

$$C_{XSTPIR}(N,K,X,T)=\left(\frac{N-X}{N}\right)C_{TPIR}(N-X,K,T) \tag{2.15}$$

$$=\frac{N-X}{NK}. \tag{2.16}$$

The proof of Theorem 2.2 is presented in Section 2.5. Since the upper bound is already provided by Theorem 2.1, only a proof of achievability is needed. Furthermore, since retrieving the desired message in this setting amounts to downloading everything stored at all servers regardless of which message is desired, the only thing required for the achievable scheme is a secure storage scheme, which is readily achieved by including X uniformly random noise symbols for every $N-X$ symbols of each message.

Next, the main result of this chapter is the asymptotic capacity characterization presented in the following theorem.

Theorem 2.3. *As the number of messages $K\to\infty$, for arbitrary N,X,T,*

$$\lim_{K\to\infty}C_{XSTPIR}(N,K,X,T)=\lim_{K\to\infty}\left(\frac{N-X}{N}\right)C_{TPIR}(N-X,K,T) \tag{2.17}$$

$$=\begin{cases}1-\left(\frac{X+T}{N}\right), & N>X+T\\ 0, & N\leq X+T.\end{cases} \tag{2.18}$$

The proof of Theorem 2.3 appears in Section 2.6. Theorem 2.3 is significant for two reasons.

[4] Note that $N>X$ by definition.

First, asymptotic capacity results are particularly relevant for PIR problems because the capacity approaches its asymptotic value extremely quickly — the gap is negligible even for moderate values of K, and K is typically a large value. Second, the asymptotic capacity result showcases a new idea, *cross subspace alignment*, that is interesting by itself.

Insights from the asymptotically optimal scheme allow us to settle the exact capacity of XSTPIR with $X=T=1$, $N=3$ and arbitrary K.

Theorem 2.4. *If the number of servers, $N=3$, and $X=T=1$, then for arbitrary number of messages, K,*

$$C_{XSTPIR}(N=3,K,X=1,T=1)=\left(\frac{N-X}{N}\right)C_{TPIR}(N-X,K,T) \tag{2.19}$$

$$=\frac{2}{3}\left(1+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{K-1}}\right)^{-1}. \tag{2.20}$$

Theorem 2.4 is proved in Section 2.7. The capacity achieving scheme introduces a new insight. For almost all PIR settings studied so far, asymptotic capacity achieving schemes have been found that send a uniformly random query vector to each server and download a product of the query vector and information stored at the server. Suppose the query vector is uniform over $\mathbb{F}_q^M$. Then with probability $1/q^M$ the query vector is all zero, and the scheme requests nothing from the server. Typically M depends on the number of messages K. As K approaches infinity the probability of requesting nothing approaches zero, so this does not help in the asymptotic sense. However, if the same scheme is used for finite K, then M is also finite, $1/q^M>0$, and the average download is reduced by the factor $(1-1/q^M)$, which improves the achieved rate of the scheme. It is remarkable that the rate achieved in this way depends on the field size. This idea is essential to the capacity achieving scheme for Theorem 2.4.

Next we present some observations that place our results in perspective.

2.3.2 Observations

Alignment of Noise and Interference

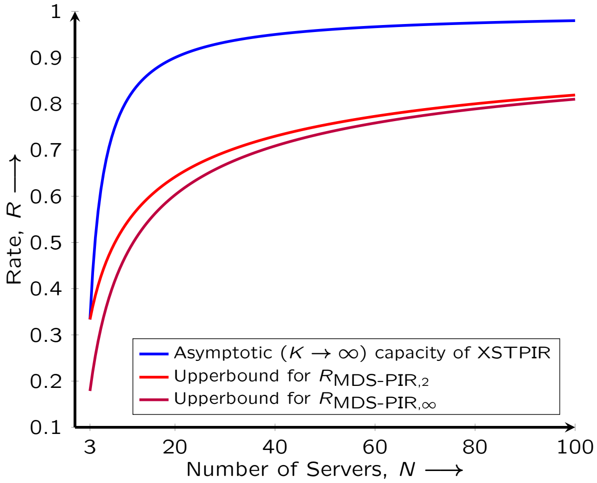


Figure 2.1: Suboptimality of the rate achieved by the $X = 1$ secure MDS-PIR alternative that allows the user to decode noise relative to the rate achieved with the asymptotically optimal XSTPIR scheme where the noise is aligned with other interference.

Consider the simplest non-trivial setting for XSTPIR, where $X = 1, T = 1$, and the number of servers, $N \geq 3$. A natural idea for providing $X = 1$ secure storage is to include 1 independent uniformly random noise symbol along with the L symbols of each message, creating a new message with $M = L+1$ symbols. This new message is stored across N servers according to an (N, M) MDS code, essentially storing a linear combination of the M message symbols at each server, where the coefficients for the noise symbol at each server must be non-zero. Capacity is known for PIR with coded storage (MDS-PIR [11]), and one might wonder if such an MDS-PIR scheme might suffice to achieve capacity with secure storage. It is not difficult to see that the best rate achievable with such an MDS-PIR scheme is

$$R_{\text{MDS-PIR}} = \frac{M-1}{M}\left(1+\left(\frac{M}{N}\right)+\cdots+\left(\frac{M}{N}\right)^{K-1}\right)^{-1}. \tag{2.21}$$

The $\frac{M-1}{M}$ penalty appears because one of the M symbols of the *decoded* message is the noise symbol. As $K\to\infty$, the rate approaches $R_{\text{MDS-PIR},\infty}=\frac{M-1}{M}\left(1-\left(\frac{M}{N}\right)\right)$. This expression takes its maximum value when $M=\sqrt{N}$, so it can be bounded as,

$$R_{\text{MDS-PIR},\infty}\leq\frac{\sqrt{N}-1}{\sqrt{N}}\left(1-\left(\frac{\sqrt{N}}{N}\right)\right)=\left(1-\frac{1}{\sqrt{N}}\right)^2. \tag{2.22}$$

Note that this expression matches the achievable rate bound of [130]. However, it is strictly smaller than, $1-2/N$, the asymptotic capacity of XSTPIR for this setting. Evidently, the natural MDS-PIR solution, and the secret sharing based scheme of [130], are asymptotically suboptimal. In fact, the MDS-PIR solution falls short of the asymptotic $(K\to\infty)$ capacity of XSTPIR, even if the MDS-PIR scheme is only required to deal with $K=2$ messages. Denoting the corresponding rate of the MDS-PIR scheme as $R_{\text{MDS-PIR},2}$, we have,

$$R_{\text{MDS-PIR},2}\leq\frac{\sqrt{N+1}}{\sqrt{N+1}+1}\left(1+\left(\frac{\sqrt{N+1}+1}{N}\right)\right)^{-1}\leq 1-\frac{2}{N}. \tag{2.23}$$

Figure 2.1 shows that the gap between the $X=1$ secure MDS-PIR alternative and the XSTPIR scheme is significant. Intuitively, the reason for this gap is the following. The secure MDS-PIR alternative allows the user to *decode* the artificial noise symbol which is added to the message to guarantee security. However, in the XSTPIR scheme, the user is able to decode only the desired message, and not the noise protecting it. In fact this noise is *aligned* with other interfering symbols, e.g., the noise terms protecting other message symbols, thus creating a more efficient solution. Incidentally, the alignment of noise provides another unexpected benefit, in some cases it automatically makes the scheme symmetrically secure, as explained next.

Symmetric Security: Capacity of Sym-XSPIR

Let us fix $T=1$, thereby relaxing the T-privacy constraint to its minimum value for PIR. Now, suppose in addition to X-secure storage, we also include the so called 'symmetric' security constraint, that the user should learn nothing about the data besides his desired message, i.e.,

$$\text{[Sym-Security]} \qquad I(W_{[K]};A_{[N]}^{[\theta]} \mid Q_{[N]}^{[\theta]},W_\theta,\theta)=0. \tag{2.24}$$

Capacity of the basic ($X=0,T=1,K>1$) Sym-PIR setting was shown in [103] to be

$$C_{\text{Sym-PIR}}(K,N)=1-\frac{1}{N}. \tag{2.25}$$

Note that there is a loss of capacity due to the additional symmetric security constraint. Furthermore, the capacity without the symmetric security constraint depends on the number of messages K while the capacity with the symmetric security constraint does not.

XSTPIR with the symmetric security constraint and with $T=1$, in short the Sym-XSPIR setting (note that we drop the T because $T=1$ is the degenerate case for T-privacy), reveals a surprising aspect of our XSTPIR schemes, that imposing the symmetric security constraint does not affect[5] our capacity results for $T=1$. This is made explicit in the following corollaries for Sym-XSPIR, that match the corresponding theorems for XSTPIR.

Corollary 2.1.

$$C_{\textit{Sym-XSPIR}}(N,K,X)\leq\left(\frac{N-X}{N}\right)C_{\textit{PIR}}(N-X,K). \tag{2.26}$$

[5] For $T>1$ our XSTPIR schemes are not symmetrically secure.

Corollary 2.2. *If $N=X+1$, then*[6] *for arbitrary K,*

$$C_{\textit{Sym-XSPIR}}(N,K,X)=\left(\frac{N-X}{N}\right)C_{\textit{PIR}}(N-X,K) \tag{2.27}$$

$$=\frac{1}{NK}. \tag{2.28}$$

Corollary 2.3. *As the number of messages $K\to\infty$, for arbitrary N,X,*

$$\lim_{K\to\infty} C_{\textit{Sym-XSPIR}}(N,K,X)=\lim_{K\to\infty}\left(\frac{N-X}{N}\right)C_{\textit{PIR}}(N-X,K) \tag{2.29}$$

$$=\begin{cases} 1-\left(\frac{X+1}{N}\right), & N>X+1 \\ 0, & N\leq X+1. \end{cases} \tag{2.30}$$

Corollary 2.4. *If the number of servers, $N=3$, and $X=1$, then for arbitrary number of messages, K,*

$$C_{\textit{Sym-XSPIR}}(N=3,K,X=1)=\left(\frac{N-X}{N}\right)C_{\textit{PIR}}(N-X,K) \tag{2.31}$$

$$=\frac{2}{3}\left(1+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{K-1}}\right)^{-1}. \tag{2.32}$$

The proofs of all 4 corollaries appear in Appendix A.2. Surprisingly, note that there is no loss of capacity in each case due to the additional symmetric security constraint. Also note that according to Corollary 2.4, unlike Sym-PIR, the capacity of Sym-XSPIR depends on the number of messages K for all $K>1$.

[6] Note that since $X<N$ by definition, and $T=1$ for XSPIR, the condition $N\leq X+T$ is equivalent to $N=X+1$.

Cross Subspace Alignment

Conceptually, the most intriguing aspect of the asymptotically optimal XSTPIR scheme is the extent to which it is able to align interference. Interference alignment is central to PIR [101, 102], and nearly all existing PIR constructions use some form of interference alignment. The strength of XSTPIR lies in the novel idea of cross subspace alignment, that we explain intuitively in this section through an example. Consider the setting of $X=2$ secure and $T=1$ PIR with $N=5$ servers. Let w_1 be a symbol from a desired message W. For simplicity (and because identical alignments are applied to all messages), it suffices to focus on only this message for the purpose of this explanation. In order to guarantee $X=2$ security, w_1 is mixed with 2 random noise symbols z_{11}, z_{12}, according to the following RS Code, so that the n^{th} row is stored at the n^{th} server, $n \in [5]$.

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} w_1 + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} z_{11} + \begin{bmatrix} \beta_1^2 \\ \beta_2^2 \\ \beta_3^2 \\ \beta_4^2 \\ \beta_5^2 \end{bmatrix} z_{12}$$

$$\triangleq \overrightarrow{1} w_1 + \overrightarrow{\beta} z_{11} + \overrightarrow{\beta^2} z_{12}.$$

To ensure privacy, the query symbol q_θ ($q_\theta = 1$, i.e., this message is desired) is similarly mixed with a noise symbol z_1'.

$$\overrightarrow{1} q_\theta + \overrightarrow{\beta} z_1' = \overrightarrow{1} + \overrightarrow{\beta} z_1'$$

and the n^{th} row of this query vector is sent to the n^{th} server. Each server returns the product of the noisy query symbol and the noisy stored symbol, so that the user receives the

5 answers.

$$\left(\overrightarrow{1}w_1+\overrightarrow{\beta}z_{11}+\overrightarrow{\beta^2}z_{12}\right)\circ\left(\overrightarrow{1}+\overrightarrow{\beta}z_1'\right)$$
$$=\overrightarrow{1}w_1+\overrightarrow{\beta}(w_1z_1'+z_{11})+\overrightarrow{\beta^2}(z_{11}z_1'+z_{12})+\overrightarrow{\beta^3}z_{12}z_1'.$$

The desired symbol w_1 appears along the vector $\overrightarrow{1}$ while the remaining 5 undesired symbols align along 3 dimensions. Specifically, the undesired symbols w_1z_1' and z_{11} align along the vector $\overrightarrow{\beta}$; undesired symbols $z_{11}z_1'$ and z_{12} align along the vector $\overrightarrow{\beta^2}$ and undesired symbol $z_{12}z_1'$ appears along the vector $\overrightarrow{\beta^3}$. This type of alignment, enabled by using the same $\overrightarrow{\beta}$ in the storage and query, is indeed very useful and has been used previously by Freij-Hollanti et al. for MDS-TPIR [36]. However, note that we have a 5 dimensional space (all vectors are 5×1) and we are so far only using 4 dimensions (one desired, three interference), so there is room for improvement.

In order to improve the efficiency of the retrieval scheme, suppose we try to retrieve another symbol, w_2, from the same desired message $W=(w_1,w_2)$. The challenge is that because of the $X=2$ security requirement w_2 is mixed with new (independent) noise symbols z_{21},z_{22} according to an RS code parameterized by γ,

$$\overrightarrow{1}w_2+\overrightarrow{\gamma}z_{21}+\overrightarrow{\gamma^2}z_{22}, \tag{2.33}$$

so any attempt to retrieve w_2 will add new interference terms. Since we already have 3 dimensions of interference, the new interference added due to the noise protecting w_2 must align completely within the existing interference. This will be accomplished by cross-alignment, i.e., introducing additional structure across the storage and query codes for the different symbols to be retrieved. In particular, we will use the query vector $\overrightarrow{\gamma}\circ\left(\overrightarrow{1}+\overrightarrow{\beta}z_1'\right)$ to multiply with the stored variables containing w_1 (i.e., $\overrightarrow{1}w_1+\overrightarrow{\beta}z_{11}+\overrightarrow{\beta^2}z_{12}$) and the query vector $\overrightarrow{\beta}\circ\left(\overrightarrow{1}+\overrightarrow{\gamma}z_2'\right)$ to multiply with the stored variables containing w_2 (i.e.,

$\overrightarrow{1}w_2+\overrightarrow{\gamma}z_{21}+\overrightarrow{\gamma}^2z_{22})$. The sum of the two multiplications is returned as the answer. Note that Hadamard products are commutative and associative. The answers from the 5 servers are now expressed as follows.

$$\overrightarrow{\gamma}\circ\left(\overrightarrow{1}w_1+\overrightarrow{\beta}z_{11}+\overrightarrow{\beta^2}z_{12}\right)\circ\left(\overrightarrow{1}+\overrightarrow{\beta}z_1'\right)+\overrightarrow{\beta}\circ(\overrightarrow{1}w_2+\overrightarrow{\gamma}z_{21}+\overrightarrow{\gamma}^2z_{22})\circ\left(\overrightarrow{1}+\overrightarrow{\gamma}z_2'\right) \tag{2.34}$$

$$=\overrightarrow{\gamma}w_1+\overrightarrow{\beta}w_2+\overrightarrow{\beta}\circ\overrightarrow{\gamma}(w_1z_1'+z_{11}+w_2z_2'+z_{21})$$
$$+\overrightarrow{\beta^2}\circ\overrightarrow{\gamma}(z_{11}z_1'+z_{12})+\overrightarrow{\beta}\circ\overrightarrow{\gamma}^2(z_{21}z_2'+z_{22})+\overrightarrow{\beta^3}\circ\overrightarrow{\gamma}z_{12}z_1'+\overrightarrow{\beta}\circ\overrightarrow{\gamma}^3z_{22}z_2'. \tag{2.35}$$

Note that we cannot choose $\overrightarrow{\beta}=\overrightarrow{\gamma}$, because the two desired symbols (w_1,w_2) must not align in the same dimension. Also note that by cross-multiplying the first set of answers with $\overrightarrow{\gamma}$ and the second with $\overrightarrow{\beta}$ we have achieved *cross alignment* of 4 terms along $\overrightarrow{\beta}\circ\overrightarrow{\gamma}$. However, we now have 5 dimensions occupied by interference, along the 5 vectors, $\overrightarrow{\beta}\circ\overrightarrow{\gamma},\overrightarrow{\beta^2}\circ\overrightarrow{\gamma},\overrightarrow{\beta}\circ\overrightarrow{\gamma}^2,\overrightarrow{\beta^3}\circ\overrightarrow{\gamma},\overrightarrow{\beta}\circ\overrightarrow{\gamma}^3$. Since the overall space is only 5 dimensional and we need two dimensions for desired symbols, we need to restrict interference to no more than 3 dimensions. Surprisingly, it is possible to do this by cross *subspace* alignment as we show next. Let us introduce a structural relationship between β and γ. In particular, let us set,

$$\overrightarrow{\beta}=\overrightarrow{f_1-\alpha} \tag{2.36}$$

$$\overrightarrow{\gamma}=\overrightarrow{f_2-\alpha}, \tag{2.37}$$

where $f_1,f_2,\alpha_1,\cdots,\alpha_N$ are distinct, so that the answers from the 5 servers are now expressed as,

$$\left(\overrightarrow{f_2-\alpha}\right)w_1+\left(\overrightarrow{f_1-\alpha}\right)w_2+\left(\overrightarrow{f_1-\alpha}\right)\circ\left(\overrightarrow{f_2-\alpha}\right)I \tag{2.38}$$

where the interference I is

$$
\begin{aligned}
I = & \overrightarrow{1}(w_1 z'_1 + z_{11} + w_2 z'_2 + z_{21}) + \left(\overrightarrow{f_1 - \alpha}\right)(z_{11} z'_1 + z_{12}) + \left(\overrightarrow{f_2 - \alpha}\right)(z_{21} z'_1 + z_{22}) \\
& + \left(\overrightarrow{f_1^2 + -2 f_1 \alpha - \alpha^2}\right) z_{12} z'_1 + \left(\overrightarrow{f_2^2 - 2 f_2 \alpha - \alpha^2}\right) z_{22} z'_2.
\end{aligned}
\tag{2.39}
$$

Note that there are still 5 interference vectors, no two of which align directly with each other. However, the 5 interference vectors align into a 3 dimensional subspace of the 5 dimensional vector space. This is what we mean by *cross subspace alignment* and it is essential to this chapter. To see explicitly how the interference aligns into a 3 dimensional subspace, we can rewrite I as,

$$
\begin{aligned}
I = & \overrightarrow{1}(w_1 z'_1 + z_{11} + w_2 z'_2 + z_{21} + f_1 z_{11} z'_1 + f_1 z_{12} + f_2 z_{21} z'_1 + f_2 z_{22} + f_1^2 z_{12} z'_1 + f_2^2 f_2^2 z_{22} z'_2) \\
& + \overrightarrow{\alpha}(-z_{11} z'_1 - z_{12} - z_{21} z'_1 - z_{22} - 2 f_1 z_{12} z'_1 - 4 f_2 z_{22} z'_2) \\
& + \overrightarrow{\alpha^2}(z_{12} z'_1 + z_{22} z'_2).
\end{aligned}
\tag{2.40}
$$

Thus, due to cross subspace alignment, all of I aligns within a 3 dimensional space, leaving the remaining 2 dimensions interference-free for the desired symbols. Exactly the same alignments apply to all messages as explained in the formal descriptions of the schemes provided in this chapter.

2.4 Proof of Theorem 2.1

Let us start with two useful lemmas. The first one shows that the desired message index is independent of the messages, stored variables, queries and answers.

Lemma 2.1. *For all $k, k' \in [K], \forall \mathcal{T} \subset [N], |\mathcal{T}| = T$, we have*

$$
(Q_{\mathcal{T}}^{[k]}, A_{\mathcal{T}}^{[k]}, S_{[N]}, W_1, \cdots, W_K) \sim (Q_{\mathcal{T}}^{[k']}, A_{\mathcal{T}}^{[k']}, S_{[N]}, W_1, \cdots, W_K)
\tag{2.41}
$$

Proof: Since $W_1,\cdots,W_K$ is a function of $S_{[N]}$ and $A_{\mathcal{T}}^{[\theta]}$ is a function of $(Q_{\mathcal{T}}^{[\theta]},S_{\mathcal{T}})$ (refer to (2.11)), it suffices to prove $I(\theta;Q_{\mathcal{T}}^{[\theta]},S_{[N]})=0$. From (2.9), we have

$$I(Q_{[N]}^{[\theta]},\theta;S_{[N]})=0 \tag{2.42}$$

$$\Rightarrow \quad I(Q_{\mathcal{T}}^{[\theta]},\theta;S_{[N]})=0 \tag{2.43}$$

$$\Rightarrow \quad I(Q_{\mathcal{T}}^{[\theta]};S_{[N]})=I(Q_{\mathcal{T}}^{[\theta]};S_{[N]}|\theta)=0 \tag{2.44}$$

Next, we have,

$$\begin{aligned}
I(\theta;Q_{\mathcal{T}}^{[\theta]},S_{[N]}) &\overset{(2.9)}{=} I(\theta;Q_{\mathcal{T}}^{[\theta]}|S_{[N]}) && (2.45)\\
&= H(Q_{\mathcal{T}}^{[\theta]}|S_{[N]})-H(Q_{\mathcal{T}}^{[\theta]}|S_{[N]},\theta) && (2.46)\\
&\overset{(2.44)}{=} H(Q_{\mathcal{T}}^{[\theta]})-H(Q_{\mathcal{T}}^{[\theta]}|\theta) && (2.47)\\
&\overset{(2.10)}{=} 0 && (2.48)
\end{aligned}$$

□

The second lemma is a statement of conditional independence of answers from one set of servers from the queries to the rest of the servers.

Lemma 2.2. *For all $\mathcal{T},\mathcal{X}\subset[N],\forall k\in[K],\forall\mathcal{K}\in[K]$, we have*

$$H(A_{\mathcal{T}}^{[k]}|S_{\mathcal{X}},Q_{[N]}^{[k]},W_{\mathcal{K}})=H(A_{\mathcal{T}}^{[k]}|S_{\mathcal{X}},Q_{\mathcal{T}}^{[k]},W_{\mathcal{K}}) \tag{2.49}$$

Proof: It suffices to prove that $I(A_{\mathcal{T}}^{[k]};Q_{[N]}^{[k]}|S_{\mathcal{X}},Q_{\mathcal{T}}^{[k]},W_{\mathcal{K}})=0$. This proof is presented as follows.

$$\begin{aligned}
I(A_{\mathcal{T}}^{[k]};Q_{[N]}^{[k]}|S_{\mathcal{X}},Q_{\mathcal{T}}^{[k]},W_{\mathcal{K}}) &\leq I(A_{\mathcal{T}}^{[k]},S_{\mathcal{X}},W_{\mathcal{K}};Q_{[N]}^{[k]}|Q_{\mathcal{T}}^{[k]}) && (2.50)\\
&\leq I(A_{\mathcal{T}}^{[k]},S_{[N]},W_{\mathcal{K}};Q_{[N]}^{[k]}|Q_{\mathcal{T}}^{[k]}) && (2.51)
\end{aligned}$$

$$\overset{(2.8)(2.11)}{=} I(S_{[N]};Q_{[N]}^{[k]}|Q_{\mathcal{T}}^{[k]}) \tag{2.52}$$

$$\overset{(2.9)}{=} 0 \tag{2.53}$$

□

The next lemma formalizes the intuition that because of the security constraint, the answers from any X servers are, in some sense, not very useful. Specifically, after conditioning on the information contained in any X servers, the answers from the remaining $N-X$ servers must still contain at least L more bits than the interference that is included in those answers. For a set $\mathcal{X}$, its complement set is denoted as $\overline{\mathcal{X}}$, i.e., $\overline{\mathcal{X}}=\{n|n\in[N],n\notin\mathcal{X}\}$. We use D_n to denote the expected number of symbols downloaded from Server n.

Lemma 2.3. *For all $\mathcal{X}\subset[N],|\mathcal{X}|=X$, we have*

$$L\leq\sum_{n\in\overline{\mathcal{X}}}D_n-H(A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]},W_1) \tag{2.54}$$

Proof:

$$\begin{aligned}
L=H(W_1) &\overset{(2.12)}{=} I(W_1;A_{[N]}^{[1]}|Q_{[N]}^{[1]}) && (2.55)\\
&\leq I(W_1;A_{[N]}^{[1]},S_{\mathcal{X}}|Q_{[N]}^{[1]}) && (2.56)\\
&= I(W_1;S_{\mathcal{X}}|Q_{[N]}^{[1]})+I(W_1;A_{\mathcal{X}}^{[1]},A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]}) && (2.57)\\
&\overset{(2.11)}{=} I(W_1;S_{\mathcal{X}}|Q_{[N]}^{[1]})+I(W_1;A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]}) && (2.58)\\
&\overset{(2.9)}{=} I(W_1,Q_{[N]}^{[1]};S_{\mathcal{X}})+I(W_1;A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]}) && (2.59)\\
&\overset{(2.7)}{=} I(Q_{[N]}^{[1]};S_{\mathcal{X}}|W_1)+I(W_1;A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]}) && (2.60)\\
&\leq I(Q_{[N]}^{[1]};S_{\mathcal{X}},W_1)+I(W_1;A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]}) && (2.61)\\
&\overset{(2.9)}{=} I(W_1;A_{\overline{\mathcal{X}}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]}) && (2.62)
\end{aligned}$$

$$\leq \sum_{n \in \overline{\mathcal{X}}} D_n - H(A_{\mathcal{X}}^{[1]} | S_{\mathcal{X}}, Q_{[N]}^{[1]}, W_1) \tag{2.63}$$

□

We may interpret the second term of the RHS of (2.54) as the interference term. To bound it, we need the following recursive relation, stated in a lemma.

Lemma 2.4. *For all $\mathcal{X} \subset [N], |\mathcal{X}| = X$ and for all $k \in [K]$, we have*

$$H(A_{\mathcal{X}}^{[k]} | S_{\mathcal{X}}, Q_{[N]}^{[k]}, W_{[k]}) \geq \frac{T}{N-X} \left(L + H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k+1]}) \right), \text{ if } N > X + T. \tag{2.64}$$

$$H(A_{\mathcal{X}}^{[k]} | S_{\mathcal{X}}, Q_{[N]}^{[k]}, W_{[k]}) \geq L + H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k+1]}), \text{ if } N \leq X + T. \tag{2.65}$$

Proof: First consider $N > X + T$. Consider any set $\mathcal{T} \subset \overline{\mathcal{X}}, |\mathcal{T}| = T$.

$$\begin{aligned} H(A_{\mathcal{X}}^{[k]} | S_{\mathcal{X}}, Q_{[N]}^{[k]}, W_{[k]}) &\geq H(A_{\mathcal{T}}^{[k]} | S_{\mathcal{X}}, Q_{[N]}^{[k]}, W_{[k]}) && (2.66) \\ &\overset{(2.49)}{=} H(A_{\mathcal{T}}^{[k]} | S_{\mathcal{X}}, Q_{\mathcal{T}}^{[k]}, W_{[k]}) && (2.67) \\ &\overset{(2.41)}{=} H(A_{\mathcal{T}}^{[k+1]} | S_{\mathcal{X}}, Q_{\mathcal{T}}^{[k+1]}, W_{[k]}) && (2.68) \\ &\overset{(2.49)}{=} H(A_{\mathcal{T}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) && (2.69) \end{aligned}$$

Averaging (2.69) over all choices of $\mathcal{T}$ and applying Han's inequality, we have

$$\begin{aligned} & H(A_{\mathcal{X}}^{[k]} | S_{\mathcal{X}}, Q_{[N]}^{[k]}, W_{[k]}) \\ &\geq \frac{T}{N-X} H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) && (2.70) \\ &\overset{(2.11)(2.12)}{=} \frac{T}{N-X} H(A_{\mathcal{X}}^{[k+1]}, W_{k+1} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) && (2.71) \\ &= \frac{T}{N-X} \left(H(W_{k+1} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) + H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k+1]}) \right) && (2.72) \\ &= \frac{T}{N-X} \left(L + H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k+1]}) \right) && (2.73) \end{aligned}$$

where the last step uses $L = H(W_{k+1})$ and $I(W_{k+1}; S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) = 0$, proved as follows.

$$\begin{aligned}
I(W_{k+1}; S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) &\overset{(2.5)(2.6)}{=} I(W_{k+1}; S_{\mathcal{X}}, Q_{[N]}^{[k+1]} \,|\, W_{[k]}) && (2.74)\\
&\le I(W_{[k+1]}; S_{\mathcal{X}}, Q_{[N]}^{[k+1]}) && (2.75)\\
&\overset{(2.7)}{=} I(W_{[k+1]}; Q_{[N]}^{[k+1]} | S_{\mathcal{X}}) && (2.76)\\
&\le I(W_{[k+1]}, S_{\mathcal{X}}; Q_{[N]}^{[k+1]}) && (2.77)\\
&\le I(S_{[N]}; Q_{[N]}^{[k+1]}) && (2.78)\\
&\overset{(2.9)}{=} 0 && (2.79)
\end{aligned}$$

Next, consider $N \le X + T$. The proof is similar to that presented above. Note that $|\mathcal{X}| = N - X \le T$.

$$\begin{aligned}
H(A_{\mathcal{X}}^{[k]} | S_{\mathcal{X}}, Q_{[N]}^{[k]}, W_{[k]}) &\overset{(2.49)}{=} H(A_{\mathcal{X}}^{[k]} | S_{\mathcal{X}}, Q_{\mathcal{X}}^{[k]}, W_{[k]}) && (2.80)\\
&\overset{(2.41)}{=} H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{\mathcal{X}}^{[k+1]}, W_{[k]}) && (2.81)\\
&\overset{(2.49)}{=} H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) && (2.82)\\
&\overset{(2.11)(2.12)}{=} H(A_{\mathcal{X}}^{[k+1]}, W_{k+1} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) && (2.83)\\
&= H(W_{k+1} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k]}) + H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k+1]}) && (2.84)\\
&\overset{(2.79)}{=} L + H(A_{\mathcal{X}}^{[k+1]} | S_{\mathcal{X}}, Q_{[N]}^{[k+1]}, W_{[k+1]}) && (2.85)
\end{aligned}$$

This completes the proof of Lemma 2.4. □

Now let us apply Lemma 2.4 repeatedly for $k = 1, 2, \cdots$. When $N > X + T$, we have

$$\begin{aligned}
H(A_{\mathcal{X}}^{[1]} | S_{\mathcal{X}}, Q_{[N]}^{[1]}, W_1) &\ge \frac{T}{N-X}\left(L + H(A_{\mathcal{X}}^{[2]} | S_{\mathcal{X}}, Q_{[N]}^{[2]}, W_{[2]}\right) && (2.86)\\
&\ge \frac{T}{N-X}\left(L + \frac{T}{N-X}\left(L + H(A_{\mathcal{X}}^{[3]} | S_{\mathcal{X}}, Q_{[N]}^{[3]}, W_{[3]}\right)\right) && (2.87)\\
&\ge \cdots && (2.88)
\end{aligned}$$

$$\geq L\left(\frac{T}{N-X}+\left(\frac{T}{N-X}\right)^2+\cdots+\left(\frac{T}{N-X}\right)^{K-1}\right) \tag{2.89}$$

Similarly, when $N \leq X+T$, we have

$$\begin{aligned}
H(A_{\mathcal{X}}^{[1]}|S_{\mathcal{X}},Q_{[N]}^{[1]},W_1) &\geq L+H(A_{\mathcal{X}}^{[2]}|S_{\mathcal{X}},Q_{[N]}^{[2]},W_{[2]}) && (2.90)\\
&\geq \cdots && (2.91)\\
&\geq L(K-1) && (2.92)
\end{aligned}$$

Substituting (2.89), (2.92) into (2.54), we have

$$\begin{aligned}
&L \leq \sum_{n\in\overline{\mathcal{X}}} D_n - L\left(\frac{T}{N-X}+\left(\frac{T}{N-X}\right)^2+\cdots+\left(\frac{T}{N-X}\right)^{K-1}\right), \text{ if } N > X+T. && (2.93)\\
&L \leq \sum_{n\in\overline{\mathcal{X}}} D_n - L(K-1), \text{ if } N \leq X+T. && (2.94)
\end{aligned}$$

Averaging over all $\mathcal{X}$, we have

$$L \leq \left(\frac{N-X}{N}\right)D - L\left(\frac{T}{N-X}+\left(\frac{T}{N-X}\right)^2+\cdots+\left(\frac{T}{N-X}\right)^{K-1}\right), \text{ if } N > X+T. \tag{2.95}$$

$$L \leq \left(\frac{N-X}{N}\right)D - L(K-1), \text{ if } N \leq X+T. \tag{2.96}$$

Finally since the rate is defined as $R=L/D$, we arrive at the final bound.

$$\begin{aligned}
&R \leq \frac{N-X}{N}\left(1+\frac{T}{N-X}+\left(\frac{T}{N-X}\right)^2+\cdots+\left(\frac{T}{N-X}\right)^{K-1}\right)^{-1}, \text{ if } N > X+T. && (2.97)\\
&R \leq \frac{N-X}{N}\times\frac{1}{K}, \text{ if } N \leq X+T. && (2.98)
\end{aligned}$$

Thus

$$C_{\text{XSTPIR}}(N,K,X,T) \leq \left(\frac{N-X}{N}\right) C_{\text{TPIR}}(N-X,K,T), \tag{2.99}$$

and the proof of Theorem 2.1 is complete. □

2.5 Proof of Theorem 2.2

Let each message consist of $L = N - X$ symbols in $\mathbb{F}_q$, $q \geq N$, and append X instances of 0 symbols, to create artificial messages of length N,

$$\bar{W}_k = (W_{k1}, W_{k_2}, \cdots, W_{k(N-X)}, \underbrace{0,0,\cdots,0}_{X}), \qquad \forall k \in [K]. \tag{2.100}$$

Corresponding to each message W_k, let $Z_k = (Z_{k1}, Z_{k2}, \cdots, Z_{kX}) \in \mathbb{F}_q^X$ be X independent uniform noise symbols, to be used for X-security. Let Z_k be encoded with an (N,X) MDS code to produce $\bar{Z}_k \in \mathbb{F}_q^N$. For each $k \in [K]$ and $n \in [N]$, the n^{th} server stores the n^{th} symbol of $\bar{W}_k + \bar{Z}_k$. Thus, each server stores a total of K symbols. The MDS property of $\bar{Z}_k$ ensures that the data storage is X-secure. Retrieval is trivial — in order to retrieve the desired message W_θ, the user simply downloads everything from all servers. Since the queries do not depend on the desired message, the scheme is N-private, so it is also T-private. The rate achieved is $\frac{N-X}{NK}$ which matches the capacity for this setting. □

2.6 Proof of Theorem 2.3

Let us present an XSTPIR scheme for arbitrary X, T, N, K, that is asymptotically optimal (as $K \to \infty$). The asymptotic capacity is zero for $N \leq X + T$, so we only need to consider

$N > X+T$. Throughout this scheme we will set

$$L = N - X - T \tag{2.101}$$

and we will use the compact notation,

$$\Delta = \prod_{i=1}^{L} (f_i - \alpha). \tag{2.102}$$

Δ_n will represent the value of Δ when α is replaced with α_n.

Each message $W_k, k \in [K]$, consists of $L = N - X - T$ symbols, $W_k = (W_{k1}, W_{k2}, \cdots, W_{kL})$ from a finite field $\mathbb{F}_q$. The field $\mathbb{F}_q$ is assumed to have size[7] $q \geq L+N$. For the design of this scheme, we will need a total of $L+N$ constants $\alpha_n, n \in [N], f_l, l \in [L]$ that are distinct elements of $\mathbb{F}_q$.

Let us split the messages into L vectors, so that $\mathbf{W}_l = (W_{1l}, W_{2l}, \cdots, W_{Kl})$, $l \in [L]$, contains the l^{th} symbol of every message. Let $\mathbf{Z}_{lx}, l \in [L], x \in [X]$, be independent uniformly random noise vectors from $\mathbb{F}_q^{1\times K}$, that are used to guarantee security. Similarly, let $\mathbf{Z}'_{lt}, l \in [L], t \in [T]$, be independent uniformly random noise vectors from $\mathbb{F}_q^{K\times 1}$, that are used to guarantee privacy. The independence between noise vectors, messages, and the user's desired message index θ is specified as follows.

$$H\Big((\mathbf{W}_l)_{l\in[L]}, (\mathbf{Z}_{lx})_{l\in[L], x\in[X]}, (\mathbf{Z}'_{lt})_{l\in[L], t\in[T]}, \theta \Big) \tag{2.103}$$

$$= H((\mathbf{W}_l)_{l\in[L]}) + H(\theta) + KL(X+T) \tag{2.104}$$

[7] In other words, we set $q = p^n$ for a prime number p and an integer $n \geq 1$ such that $p^n \geq L+N$. While this makes the scheme more general, let us note that for simplicity it may be desirable to choose $n=1$ and $q = p \geq L+N$. On the other hand, the general scheme is useful for extensions of results in this chapter, say to private computation (see Footnote 8), where the choice of field may be fixed by the functions that need to be computed.

in q-ary units. Let $\mathbf{Q}_\theta$ represent[8] the θ^{th} column of the $K\times K$ identity matrix, so it contains a 1 in the θ^{th} position and zeros everywhere else. Note that

$$(\mathbf{W}_1\mathbf{Q}_\theta,\mathbf{W}_2\mathbf{Q}_\theta,\cdots,\mathbf{W}_L\mathbf{Q}_\theta)=(W_{\theta 1},W_{\theta 2},\cdots,W_{\theta L})=W_\theta \tag{2.105}$$

is the message desired by the user. A succinct summary of the storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server, is provided below.

	Server 'n' (Replace α,Δ with α_n,Δ_n)
Storage (S_n)	$\mathbf{W}_1+(f_1-\alpha)\mathbf{Z}_{11}+\cdots+(f_1-\alpha)^X\mathbf{Z}_{1X}$, $\mathbf{W}_2+(f_2-\alpha)\mathbf{Z}_{21}+\cdots+(f_2-\alpha)^X\mathbf{Z}_{2X}$, $\vdots$ $\mathbf{W}_L+(f_L-\alpha)\mathbf{Z}_{L1}+\cdots+(f_L-\alpha)^X\mathbf{Z}_{LX}$
Query $(Q_n^{[\theta]})$	$\frac{\Delta}{f_1-\alpha}\Big(\mathbf{Q}_\theta+(f_1-\alpha)\mathbf{Z}'_{11}+\cdots+(f_1-\alpha)^T\mathbf{Z}'_{1T}\Big)$, $\frac{\Delta}{f_2-\alpha}\Big(\mathbf{Q}_\theta+(f_2-\alpha)\mathbf{Z}'_{21}+\cdots+(f_2-\alpha)^T\mathbf{Z}'_{2T}\Big)$, $\vdots$ $\frac{\Delta}{f_L-\alpha}\Big(\mathbf{Q}_\theta+(f_L-\alpha)\mathbf{Z}'_{L1}+\cdots+(f_L-\alpha)^T\mathbf{Z}'_{LT}\Big)$
	Desired symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{(f_2-\alpha)^{-1}},\cdots,\overrightarrow{(f_L-\alpha)^{-1}}\Big)$ Interference appears along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{1},\overrightarrow{(f_1-\alpha)},\cdots,\overrightarrow{(f_1-\alpha)^{X+T-1}},\overrightarrow{(f_2-\alpha)},\cdots,\overrightarrow{(f_2-\alpha)^{X+T-1}},\cdots,$ $\cdots,\overrightarrow{(f_L-\alpha)},\cdots,\overrightarrow{(f_L-\alpha)^{X+T-1}}\Big)$

Table 2.1: A summary of the XSTPIR scheme showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

Initially, the user knows only his desired message index θ and the noise terms $\mathbf{Z}'_{lt},l\in[L],t\in$

[8] Note that the XSTPIR scheme described in this section works even if $\mathbf{Q}_\theta$ is an arbitrary vector, i.e., if instead of retrieving one of the K messages, the user wishes to compute an arbitrary linear function of the K messages over $\mathbb{F}_q$. Thus, the scheme automatically settles the asymptotic capacity of the natural X-secure and T-private generalization of the linear private computation problem introduced in [107] (also known as linear private function retrieval [81]).

$[T]$, all of which are privately generated by the user. Each server $n \in [N]$ knows only its stored information S_n. The storage S_n at Server n may be viewed as a $1 \times LK$ row vector formed by concatenating the L row vectors, $\mathbf{W}_l + \sum_{x=1}^{X}(f_l - \alpha_n)^x \mathbf{Z}_{lx}$, $l \in [L]$. Similarly, the query $Q_n^{[\theta]}$ may be viewed as an $LK \times 1$ column vector formed by concatenating the L column vectors, $\frac{\Delta_n}{f_l - \alpha_n}\left(\mathbf{Q}_\theta + \sum_{t=1}^{T}(f_l - \alpha_n)^t \mathbf{Z}'_{lt}\right)$, $l \in [L]$.

Upon receiving the query $Q_n^{[\theta]}$ from the user, Server n responds with the answer $A_n^{[\theta]}$ that is exactly one symbol in $\mathbb{F}_q$, found by multiplying S_n with $Q_n^{[\theta]}$.

$$A_n^{[\theta]} = S_n Q_n^{[\theta]}. \tag{2.106}$$

This produces a single equation in a total of $L(X+1)(T+1)$ terms. Out of these, L terms are desired message symbols $\mathbf{W}_l \mathbf{Q}_\theta$, $l \in [L]$, and the remaining $L(X+1)(T+1) - L$ terms are undesired, or interference terms. The interference terms include LT terms of the type $\mathbf{W}_l \mathbf{Z}'_{lt}$, LX terms of the type $\mathbf{Z}_{lx}\mathbf{Q}_\theta$, and LXT terms of the type $\mathbf{Z}_{lx}\mathbf{Z}'_{lt}$. The user obtains one such equation from each server, for a total of N equations, from which he must be able to retrieve his L desired symbols. The key to this is the alignment of $L(X+1)(T+1) - L$ interference terms into $N - L$ dimensions, leaving L dimensions free from interference from which the L desired symbols can be decoded.

First let us identify the desired signal dimensions, i.e., the vectors along which desired symbols are seen by the user. Each answer $A_n^{[\theta]}$ contains the desired symbols $\frac{\Delta_n}{f_l - \alpha_n}\mathbf{W}_l \mathbf{Q}_\theta = \frac{\Delta_n}{f_l - \alpha_n} W_{\theta l}$, $l \in [L]$. These L desired symbols appear along the following L vectors.

$$\begin{bmatrix} \frac{\Delta_1}{f_1 - \alpha_1} \\ \frac{\Delta_2}{f_1 - \alpha_2} \\ \vdots \\ \frac{\Delta_N}{f_1 - \alpha_N} \end{bmatrix}, \begin{bmatrix} \frac{\Delta_1}{f_2 - \alpha_1} \\ \frac{\Delta_2}{f_2 - \alpha_2} \\ \vdots \\ \frac{\Delta_N}{f_2 - \alpha_N} \end{bmatrix}, \cdots, \begin{bmatrix} \frac{\Delta_1}{f_L - \alpha_1} \\ \frac{\Delta_2}{f_L - \alpha_2} \\ \vdots \\ \frac{\Delta_N}{f_L - \alpha_N} \end{bmatrix} \triangleq \overrightarrow{\Delta} \circ \left(\overrightarrow{(f_1 - \alpha)^{-1}}, \overrightarrow{(f_2 - \alpha)^{-1}}, \cdots, \overrightarrow{(f_L - \alpha)^{-1}}\right). \tag{2.107}$$

Recall that $\circ$ represents the Hadamard product. Similarly, the vectors along which interference symbols appear are identified as follows.

$$\overrightarrow{\Delta}\circ\Big(\overrightarrow{1},\overrightarrow{(f_1-\alpha)},\cdots,\overrightarrow{(f_1-\alpha)^{X+T-1}},\overrightarrow{(f_2-\alpha)},\cdots,\overrightarrow{(f_2-\alpha)^{X+T-1}},\cdots,$$
$$\cdots,\overrightarrow{(f_L-\alpha)},\cdots,\overrightarrow{(f_L-\alpha)^{X+T-1}}\Big). \tag{2.108}$$

Thus, the vector of answers from all N servers can be expressed as

$$\overrightarrow{A^{[\theta]}}=\sum_{l=1}^{L}W_{\theta l}\overrightarrow{\Delta}\circ\overrightarrow{(f_l-\alpha)^{-1}}+\sum_{l=1}^{L}\sum_{i=0}^{X+T-1}\overrightarrow{\Delta}\circ\overrightarrow{(f_l-\alpha)^{i}}I_{li} \tag{2.109}$$

for some interference terms I_{li} that are sums of various $\mathbf{W}_l\mathbf{Z}'_{lt}$, $\mathbf{Z}_{lx}\mathbf{Q}_\theta$, and $\mathbf{Z}_{lx}\mathbf{Z}'_{lt}$ terms. The exact form of I_{li} terms is not important for our analysis. Using binomial expansion to write each $\overrightarrow{(f_l-\alpha)^{i}}$ vector as $\sum_{j=0}^{i}\binom{i}{j}f_l^j\overrightarrow{\alpha^{i-j}}$, and grouping terms by the vectors $\overrightarrow{\alpha^{i}}$, we can write,

$$\overrightarrow{A^{[\theta]}}=\sum_{l=1}^{L}W_{\theta l}\overrightarrow{\Delta}\circ\overrightarrow{(f_l-\alpha)^{-1}}+\sum_{i=0}^{X+T-1}\overrightarrow{\Delta}\circ\overrightarrow{\alpha^{i}}I'_i. \tag{2.110}$$

Thus, all interference is aligned within the subspace spanned by vectors $\overrightarrow{\Delta}$, $\overrightarrow{\Delta}\circ\overrightarrow{\alpha}$, ..., $\overrightarrow{\Delta}\circ\overrightarrow{\alpha^{X+T-1}}$. As explained in Section 2.3.2, this is because of *cross subspace alignment.*

In matrix notation, we have,

$$\overrightarrow{A^{[\theta]}}=\begin{bmatrix}A_1^{[\theta]}\\A_2^{[\theta]}\\\vdots\\A_N^{[\theta]}\end{bmatrix}=\mathbf{M}_N\begin{bmatrix}W_{\theta1}\\\vdots\\W_{\theta L}\\I'_0\\\vdots\\I'_{(X+T-1)}\end{bmatrix} \tag{2.111}$$

where the $N \times N$ square matrix (note that $L+X+T=N$)

$$\mathbf{M}_N = \begin{bmatrix} \frac{\Delta_1}{f_1-\alpha_1} & \cdots & \frac{\Delta_1}{f_L-\alpha_1} & \Delta_1 & \Delta_1\alpha_1 & \cdots & \Delta_1\alpha_1^{X+T-1} \\ \frac{\Delta_2}{f_1-\alpha_2} & \cdots & \frac{\Delta_2}{f_L-\alpha_2} & \Delta_2 & \Delta_2\alpha_2 & \cdots & \Delta_2\alpha_2^{X+T-1} \\ \vdots & & & & & & \\ \frac{\Delta_N}{f_1-\alpha_N} & \cdots & \frac{\Delta_N}{f_L-\alpha_N} & \Delta_N & \Delta_N\alpha_N & \cdots & \Delta_N\alpha_N^{X+T-1} \end{bmatrix} \tag{2.112}$$

$$= \left[\overrightarrow{\Delta} \circ \overrightarrow{(f_1-\alpha)^{-1}} \cdots \overrightarrow{\Delta} \circ \overrightarrow{(f_L-\alpha)^{-1}} \; \overrightarrow{\Delta} \; \overrightarrow{\Delta} \circ \overrightarrow{\alpha} \cdots \overrightarrow{\Delta} \circ \overrightarrow{\alpha^{X+T-1}} \right] \tag{2.113}$$

is called the decoding matrix. Evidently, if the decoding matrix is invertible, then the user can recover his L desired message symbols. We show that if $\alpha_n, n \in [N], f_l, l \in [L]$ are distinct elements of $\mathbb{F}_q$, then M_N is invertible. Indeed, the invertibility of the matrix $\mathbf{M}_N$ follows immediately from the determinant of Cauchy-Vandermonde matrices (see, e.g., [37]). For the sake of completeness, the result, as well as a compact proof, is stated in the following lemma. Note that in our design, we have chosen α_n, f_l as distinct elements, so Lemma 2.5 guarantees that the scheme satisfies the correctness constraint. Fixing distinct values of $\alpha_1, \cdots, \alpha_N, f_1, \cdots, f_l$ completes the design of the scheme.

Lemma 2.5. *The decoding matrix $\mathbf{M}_N$ is invertible if all $\alpha_n, n \in [N], f_l, l \in [L]$ are distinct.*

Proof. To set up the proof by contradiction, suppose on the contrary that $\mathbf{M}_N$ is singular. Then there must exist $c_n \in \mathbb{F}_q, n \in [N]$, at least one of which is non-zero, such that

$$c_1 \overrightarrow{\Delta} \circ \overrightarrow{(f_1-\alpha)^{-1}} + \cdots + c_L \overrightarrow{\Delta} \circ \overrightarrow{(f_L-\alpha)^{-1}} + c_{L+1} \overrightarrow{\Delta} + c_{L+2} \overrightarrow{\Delta} \circ \overrightarrow{\alpha} + \cdots + c_N \overrightarrow{\Delta} \circ \overrightarrow{\alpha^{X+T-1}} = \overrightarrow{0} \tag{2.114}$$

where $\overrightarrow{0}$ is the vector whose elements are all 0. Now consider n-th row of (2.114).

$$c_1 \frac{\Delta_n}{f_1-\alpha_n} + \cdots + c_L \frac{\Delta_n}{f_L-\alpha_n} + c_{L+1}\Delta_n + c_{L+2}\Delta_n\alpha_n + \cdots + c_N \Delta_n \alpha_n^{X+T-1} = 0. \tag{2.115}$$

From (2.102), we know that $\Delta_n \neq 0$. Then α_n must be the root of the following polynomial

$$g(\alpha) = \sum_{i=1}^{L} c_i \left(\frac{\Delta}{f_i - \alpha} \right) + \sum_{i=L+1}^{N} c_i \Delta \alpha^{i-(L+1)} \tag{2.116}$$

Note that Δ (as a function of α) has order L and $f_i - \alpha$ is a factor of Δ (refer to (2.102)), so $g(\alpha)$ has order *at most* $N-1$. If $g(\alpha)$ is a non-zero polynomial, then it can have at most $N-1$ roots over $\mathbb{F}_q$. Now $\alpha_n, n \in [N]$ are N distinct roots of $g(\alpha)$, thus $g(\alpha)$ must be the zero polynomial, i.e., the coefficients of all monomials in $g(\alpha)$ must be zero. The coefficient of α^{N-1} is c_N so we must have $c_N = 0$. Then, the remaining coefficient of α^{N-2} is c_{N-1}, so we must have $c_{N-1} = 0$. Similarly, we find $c_{L+1} = c_{L+2} = \cdots = c_N = 0$, leaving us with

$$g(\alpha) = \sum_{i=1}^{L} c_i \left(\frac{\Delta}{f_i - \alpha} \right). \tag{2.117}$$

Now, if this $g(\alpha)$ is the zero polynomial, then it must be zero for every $\alpha \in \mathbb{F}_q$. Choosing α such[9] that $(f_i - \alpha) = 0$, gives us $c_i = 0$ for every $i \in [L]$. Thus, we have $c_1 = c_2 = \cdots = c_N = 0$. This is a contradiction since we assumed that at least one of $c_n, n \in [N]$ is non-zero. Thus, the proof is complete. □

Now consider the security guarantee. For any X colluding servers, $i_1, i_2, \cdots, i_X$, the X observations, $U_{kl1}, \cdots, U_{klX}$, of each message symbol $W_{kl}, k \in [K], l \in [L]$, are protected by noise

[9] Note that $\frac{\Delta}{f_i - \alpha}$ is simply a compact notation for $\prod_{l \in [L], l \neq i} (l - \alpha)$, i.e., it only means that the $(f_i - \alpha)$ factor is eliminated from Δ, so there is no 'division by 0' when we set $i - \alpha = 0$ in $\frac{\Delta}{f_i - \alpha}$.

terms as follows.

$$\begin{bmatrix} U_{kl1} \\ \vdots \\ U_{klX} \end{bmatrix} = \begin{bmatrix} W_{kl} \\ \vdots \\ W_{kl} \end{bmatrix} + \underbrace{\begin{bmatrix} f_l-\alpha_{i_1} & (f_l-\alpha_{i_1})^2 & \cdots & (f_l-\alpha_{i_1})^X \\ f_l-\alpha_{i_2} & (f_l-\alpha_{i_2})^2 & \cdots & (f_l-\alpha_{i_2})^X \\ \vdots & \vdots & & \vdots \\ f_l-\alpha_{i_X} & (f_l-\alpha_{i_X})^2 & \cdots & (f_l-\alpha_{i_X})^X \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} \mathbf{Z}_{l1}(k) \\ \vdots \\ \mathbf{Z}_{lX}(k) \end{bmatrix}}_{Z} \tag{2.118}$$

$$= W_{kl}\mathbf{1} + \begin{bmatrix} l-\alpha_{i_1} & 0 & \cdots & 0 \\ 0 & l-\alpha_{i_2} & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & l-\alpha_{i_X} \end{bmatrix} \begin{bmatrix} 1 & f_l-\alpha_{i_1} & \cdots & (f_l-\alpha_{i_1})^{X-1} \\ 1 & f_l-\alpha_{i_2} & \cdots & (f_l-\alpha_{i_2})^{X-1} \\ \vdots & \vdots & & \vdots \\ 1 & f_l-\alpha_{i_X} & \cdots & (f_l-\alpha_{i_X})^{X-1} \end{bmatrix} \begin{bmatrix} \mathbf{Z}_{l1}(k) \\ \vdots \\ \mathbf{Z}_{lX}(k) \end{bmatrix}. \tag{2.119}$$

where $\mathbf{Z}_{lx}(k)$ is the k^{th} element of the vector $\mathbf{Z}_{lx}$. Note that P is a product of a diagonal matrix which is invertible because $(f_l-\alpha_{i_j})$ are non-zero, and a Vandermonde matrix which is invertible because $(f_l-\alpha_{i_j})$ are distinct. Therefore, P is invertible, and the observations are independent of the message symbols as shown below.

$$I(W_{kl};(U_{klx})_{x\in[X]}) = I(W_{kl};W_{kl}\mathbf{1}+PZ) = I(W_{kl};W_{kl}P^{-1}\mathbf{1}+Z) = I(W_{kl};Z) = 0. \tag{2.120}$$

Furthermore, since the noise terms protecting each message symbol $W_{kl}, k\in[K], l\in[L]$, i.e., $\mathbf{Z}_{lx}(k), x\in[X]$ are independent across (k,l,x), security is preserved for all data.

The noise terms protecting each *query* also have the same structure and independence properties by design. Therefore, it follows from the same reasoning that user's privacy is protected from any T colluding servers.

Finally, note that the user is able to retrieve $L=N-X-T$ desired q-ary symbols by downloading N q-ary symbols, one from each server. The rate achieved is $L/N = 1-(X+T)/N$, which is the asymptotic capacity for this general setting. This completes the proof of Theorem 2.3. □

2.6.1 Example: $(X=1)$ Secure, $(T=1)$ Private Scheme with $N=5$ Servers

Each message consists of $L=3$ symbols from a finite field $\mathbb{F}_q$, $q\geq N+L=8$, and characteristic greater than 2. For this setting, $\Delta=(f_1-\alpha)(f_2-\alpha)(f_3-\alpha)$. The answers from all $N=5$ servers may be written explicitly as,

	Server 'n' (Replace α,Δ with α_n,Δ_n)
Storage (S_n)	$\mathbf{W}_1+(f_1-\alpha)\mathbf{Z}_1$, $\mathbf{W}_2+(f_2-\alpha)\mathbf{Z}_2$, $\mathbf{W}_3+(f_3-\alpha)\mathbf{Z}_3$
Query ($Q_n^{[\theta]}$)	$\frac{\Delta}{f_1-\alpha}\Big(\mathbf{Q}_\theta+(f_1-\alpha)\mathbf{Z}_1'\Big)$, $\frac{\Delta}{f_2-\alpha}\Big(\mathbf{Q}_\theta+(f_2-\alpha)\mathbf{Z}_2'\Big)$, $\frac{\Delta}{f_3-\alpha}\Big(\mathbf{Q}_\theta+(f_3-\alpha)\mathbf{Z}_3'\Big)$
Desired symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{(f_2-\alpha)^{-1}},\overrightarrow{(f_3-\alpha)^{-1}}\Big)$ Interference symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{1},\overrightarrow{f_1-\alpha},\overrightarrow{f_2-\alpha},\overrightarrow{f_3-\alpha}\Big)$	

Table 2.2: A summary of the XSTPIR scheme for $X=1,T=1,N=5$, showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

$$\begin{aligned}\overrightarrow{A^{[\theta]}}=&\overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)^{-1}}\mathbf{W}_1\mathbf{Q}_\theta+\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)^{-1}}\mathbf{W}_2\mathbf{Q}_\theta+\overrightarrow{\Delta}\circ\overrightarrow{(f_3-\alpha)^{-1}}\mathbf{W}_3\mathbf{Q}_\theta\\&+\overrightarrow{\Delta}\underbrace{(\mathbf{W}_1\mathbf{Z}_1'+\mathbf{W}_2\mathbf{Z}_2'+\mathbf{W}_3\mathbf{Z}_3'+\mathbf{Z}_1\mathbf{Q}_\theta+\mathbf{Z}_2\mathbf{Q}_\theta+\mathbf{Z}_3\mathbf{Q}_\theta)}_{I_{10}+I_{20}+I_{30}}\\&+\overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)}\underbrace{(\mathbf{Z}_1\mathbf{Z}_1')}_{I_{11}}+\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)}\underbrace{(\mathbf{Z}_2\mathbf{Z}_2')}_{I_{21}}+\overrightarrow{\Delta}\circ\overrightarrow{(f_3-\alpha)}\underbrace{(\mathbf{Z}_3\mathbf{Z}_3')}_{I_{31}}\end{aligned}\tag{2.121}$$

$$= \overrightarrow{\Delta} \circ \overrightarrow{(f_1-\alpha)^{-1}} W_{\theta 1} + \overrightarrow{\Delta} \circ \overrightarrow{(f_2-\alpha)^{-1}} W_{\theta 2} + \overrightarrow{\Delta} \circ \overrightarrow{(f_3-\alpha)^{-1}} W_{\theta 3}$$

$$+ \overrightarrow{\Delta}(I_{10}+I_{20}+I_{30}+f_1 I_{11}+f_2 I_{21}+f_3 I_{31}) + \overrightarrow{\Delta} \circ \overrightarrow{\alpha}(-I_{11}-I_{21}-I_{31}). \quad (2.122)$$

Privacy and security are guaranteed since $f_1-\alpha_n \neq 0, \forall n \in [5]$, the messages and queries are hidden behind the noise.

Interference terms align into the space spanned by the two vectors, $\overrightarrow{\Delta}, \overrightarrow{\Delta} \circ \overrightarrow{\alpha}$, while the 3 symbols of the desired message appear along $\overrightarrow{\Delta} \circ \overrightarrow{(f_1-\alpha)^{-1}}, \overrightarrow{\Delta} \circ \overrightarrow{(f_2-\alpha)^{-1}}, \overrightarrow{\Delta} \circ \overrightarrow{(f_3-\alpha)^{-1}}$. Independence of the 3 desired signal dimensions from the two interference dimensions is trivially verified, because the highest exponent of α along desired signal dimensions is 2, but each interference dimension has an α^3 term (contributed by Δ). Independence of the 3 desired signal dimensions among themselves is also easily verified, because for

$$c_1(f_2-\alpha)(f_3-\alpha)+c_2(f_1-\alpha)(f_3-\alpha)+c_3(f_1-\alpha)(f_2-\alpha) \quad (2.123)$$

to be the zero polynomial it must be zero everywhere, but in that case, setting $\alpha - f_i = 0$ for $i = 1,2,3$, leads us to $c_1 = c_2 = c_3 = 0$, thus proving their independence. The rate achieved is 3/5, which matches the asymptotic capacity for this setting.

2.6.2 Example: $(X = 2)$ Secure, $(T = 1)$ Private Scheme with $N = 4$ Servers

Each message consists of $L = 1$ symbol from a finite field $\mathbb{F}_q$, $q \geq N+L = 5$. $\Delta = (1-\alpha)$. The answers from all $N = 4$ servers may be written explicitly as,

$$\overrightarrow{A^{[\theta]}} = \overrightarrow{1}\mathbf{W}_1\mathbf{Q}_\theta + \overrightarrow{(f_1-\alpha)}(\mathbf{W}_1\mathbf{Z}'_1+\mathbf{Z}_{11}\mathbf{Q}_\theta) + \overrightarrow{(f_1-\alpha)^2}(\mathbf{Z}_{11}\mathbf{Z}'_1+\mathbf{Z}_{12}\mathbf{Q}_\theta) + \overrightarrow{(f_1-\alpha)^3}\mathbf{Z}_{12}\mathbf{Z}'_1 \quad (2.124)$$

	Server 'n' (Replace α,Δ with α_n,Δ_n)
Storage (S_n)	$\mathbf{W}_1+(f_1-\alpha)\mathbf{Z}_{11}+(f_1-\alpha)^2\mathbf{Z}_{12}$
Query ($Q_n^{[\theta]}$)	$\mathbf{Q}_\theta+(f_1-\alpha)\mathbf{Z}'_1$
Desired symbols appear along vector $\overrightarrow{1}$	
Interference symbols appear along vectors $\overrightarrow{\Delta}\circ\left(\overrightarrow{1},\overrightarrow{f_1-\alpha},\overrightarrow{(f_1-\alpha)^2}\right)$	

Table 2.3: A summary of the XSTPIR scheme for $X=2,T=1,N=4$, showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

$$= \overrightarrow{1}W_{\theta 1}+\overrightarrow{\Delta}\circ\left(\overrightarrow{1}\underbrace{(\mathbf{W}_1\mathbf{Z}'_1+\mathbf{Z}_{11}\mathbf{Q}_\theta)}_{I_{10}}+\overrightarrow{(f_1-\alpha)}\underbrace{(\mathbf{Z}_{11}\mathbf{Z}'_1+\mathbf{Z}_{12}\mathbf{Q}_\theta)}_{I_{11}}+\overrightarrow{(f_1-\alpha)^2}\underbrace{\mathbf{Z}_{12}\mathbf{Z}'_1}_{I_{12}}\right) \tag{2.125}$$

$$= \overrightarrow{1}W_{\theta 1}+\overrightarrow{\Delta}\underbrace{(I_{10}+f_1I_{11}+f_2I_{12})}_{I'_0}+\overrightarrow{\Delta}\circ\overrightarrow{\alpha}\underbrace{(-I_{11}-2f_1I_{12})}_{I'_1}+\overrightarrow{\Delta}\circ\overrightarrow{\alpha^2}\underbrace{(I_{12})}_{I'_2}. \tag{2.126}$$

Interference aligns in the space spanned by the three vectors, $\overrightarrow{\Delta},\overrightarrow{\Delta}\circ\overrightarrow{\alpha},\overrightarrow{\Delta}\circ\overrightarrow{\alpha^2}$, while the desired symbol appears along the vector of all ones. The independence of these directions is easily established. Privacy is guaranteed because $f_1-\alpha_n\neq 0$, $\forall n\in[4]$, so the queries are hidden behind random noise. Security is guaranteed because for any $X=2$ colluding servers, i and j, the independent noise protecting each message W_k, $k\in[K]$,

$$\underbrace{\begin{bmatrix} f_1-\alpha_i & (f_1-\alpha_i)^2 \\ f_1-\alpha_j & (f_1-\alpha_j)^2 \end{bmatrix}}_{P_{ij}}\begin{bmatrix}\mathbf{Z}_{11}(k)\\ \mathbf{Z}_{12}(k)\end{bmatrix}=\begin{bmatrix} f_1-\alpha_i & 0 \\ 0 & f_1-\alpha_j\end{bmatrix}\begin{bmatrix} 1 & (f_1-\alpha_i) \\ 1 & (f_1-\alpha_j)\end{bmatrix}\begin{bmatrix}\mathbf{Z}_{11}(k)\\ \mathbf{Z}_{12}(k)\end{bmatrix} \tag{2.127}$$

spans $X=2$ dimensions, because P_{ij} is invertible for distinct and non-zero values of $(1-\alpha_i),(1-\alpha_j)$. The rate achieved is $1/4$ which matches the asymptotic capacity for this setting.

2.6.3 Example: $(X=1)$ Secure, $(T=2)$ Private Scheme with $N=5$ Servers

Each message consists of $L=N-X-T=2$ symbols from $\mathbb{F}_q$, $q\geq 7$.

$$\Delta=(f_1-\alpha)(f_2-\alpha). \tag{2.128}$$

The answers from all $N=5$ servers may be written explicitly as,

	Server 'n' (Replace α,Δ with α_n,Δ_n)
Storage (S_n)	$\mathbf{W}_1+(f_1-\alpha)\mathbf{Z}_1,$ $\mathbf{W}_2+(f_2-\alpha)\mathbf{Z}_2$
Query $(Q_n^{[\theta]})$	$\frac{\Delta}{f_1-\alpha}\Big(\mathbf{Q}_\theta+(f_1-\alpha)\mathbf{Z}'_{11}+(f_1-\alpha)^2\mathbf{Z}'_{12}\Big),$ $\frac{\Delta}{f_2-\alpha}\Big(\mathbf{Q}_\theta+(f_2-\alpha)\mathbf{Z}'_{21}+(f_2-\alpha)^2\mathbf{Z}'_{22}\Big)$
Desired symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{(f_2-\alpha)^{-1}}\Big)$	
Interference symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{1},\overrightarrow{f_1-\alpha},\overrightarrow{(f_1-\alpha)^2},\overrightarrow{f_2-\alpha},\overrightarrow{(f_2-\alpha)^2}\Big)$	

Table 2.4: A summary of the XSTPIR scheme for $X=1,T=2,N=5$, showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

$$\begin{aligned}\overrightarrow{A^{[\theta]}}=&\ \overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)^{-1}}\mathbf{W}_1\mathbf{Q}_\theta+\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)^{-1}}\mathbf{W}_2\mathbf{Q}_\theta \\ &+\overrightarrow{\Delta}\underbrace{(\mathbf{W}_1\mathbf{Z}'_{11}+\mathbf{W}_2\mathbf{Z}'_{21}+\mathbf{Z}_1\mathbf{Q}_\theta+\mathbf{Z}_2\mathbf{Q}_\theta)}_{I_{10}+I_{20}}+\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)^2}\underbrace{\mathbf{Z}_2\mathbf{Z}'_{22}}_{I_{22}} \\ &+\overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)}\underbrace{(\mathbf{Z}_1\mathbf{Z}'_{11}+\mathbf{W}_1\mathbf{Z}'_{12})}_{I_{11}}+\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)}\underbrace{(\mathbf{Z}_2\mathbf{Z}'_{21}+\mathbf{W}_2\mathbf{Z}'_{22})}_{I_{21}} \\ &+\overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)^2}\underbrace{\mathbf{Z}_1\mathbf{Z}'_{12}}_{I_{12}} \\ =&\ \overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)^{-1}}W_{\theta 1}+\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)^{-1}}W_{\theta 2}\end{aligned} \tag{2.129}$$

$$+\overrightarrow{\Delta}(I_{10}+I_{20}+f_1I_{11}+f_2I_{21}+f_1^2I_{12}+f_2^2I_{22})$$
$$+\overrightarrow{\Delta}\circ\overrightarrow{\alpha}(-I_{11}-I_{21}-2f_1I_{12}-2f_2I_{22})+\overrightarrow{\Delta}\circ\overrightarrow{\alpha^2}(I_{12}+I_{22}). \tag{2.130}$$

Thus, interference aligns into the space spanned by the 3 vectors: $\overrightarrow{\Delta},\overrightarrow{\Delta}\circ\overrightarrow{\alpha},\overrightarrow{\Delta}\circ\overrightarrow{\alpha^2}$, while the 2 desired symbols appear along $\overrightarrow{\Delta}\circ\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{\Delta}\circ\overrightarrow{(f_2-\alpha)^{-1}}$. Note that the highest exponent of α along a desired signal dimension is 1, but every interference dimension contains α^2 (contributed by Δ), so the desired signals are independent of the interference. The independence of desired signals among themselves is also easily verified because if $c_1\frac{\Delta}{f_1-\alpha}+c_2\frac{\Delta}{f_2-\alpha}=c_1(f_2-\alpha)+c_2(f_1-\alpha)$ is the zero polynomial, then by substituting $f_i-\alpha=0$ for $i=1,2$ we find that we must have $c_1=c_2=0$. Privacy and security are guaranteed by the MDS coded independent noise terms mixed with the message and query symbols. The rate achieved is 2/5, which matches the asymptotic capacity for this setting.

2.6.4 Example: $(X=2)$ Secure, $(T=2)$ Private Scheme with $N=7$ Servers

Each message consists of $L=3$ symbols from a finite field $\mathbb{F}_q$, of size $q\geq 10$ and characteristic greater than 2.

$$\Delta=(f_1-\alpha)(f_2-\alpha)(f_3-\alpha).$$

Interference aligns into the space spanned by the 4 vectors: $\overrightarrow{\Delta},\overrightarrow{\Delta}\circ\overrightarrow{\alpha},\overrightarrow{\Delta}\circ\overrightarrow{\alpha^2},\overrightarrow{\Delta}\circ\overrightarrow{\alpha^3}$. Independence of desired signals from interference is trivially verified – highest exponent of α along any desired signal dimension is 2, but each interference dimension has an α^3 term (contributed by Δ). The desired signal dimensions are easily verified to be linearly independent among themselves because in order for

$$c_1(f_2-\alpha)(f_3-\alpha)+c_2(f_1-\alpha)(f_3-\alpha)+c_3(f_1-\alpha)(f_2-\alpha) \tag{2.131}$$

	Server ‘n’ (Replace α,Δ with α_n,Δ_n)
Storage (S_n)	$\mathbf{W}_1+(f_1-\alpha)\mathbf{Z}_{11}+(f_1-\alpha)^2\mathbf{Z}_{12}$, $\mathbf{W}_2+(f_2-\alpha)\mathbf{Z}_{21}+(f_2-\alpha)^2\mathbf{Z}_{22}$, $\mathbf{W}_3+(f_3-\alpha)\mathbf{Z}_{31}+(f_3-\alpha)^2\mathbf{Z}_{32}$
Query $(Q_n^{[\theta]})$	$\frac{\Delta}{f_1-\alpha}\Big(\mathbf{Q}_\theta+(f_1-\alpha)\mathbf{Z}'_{11}+(f_1-\alpha)^2\mathbf{Z}'_{12}\Big)$, $\frac{\Delta}{f_2-\alpha}\Big(\mathbf{Q}_\theta+(f_2-\alpha)\mathbf{Z}'_{21}+(f_2-\alpha)^2\mathbf{Z}'_{22}\Big)$, $\frac{\Delta}{f_3-\alpha}\Big(\mathbf{Q}_\theta+(f_3-\alpha)\mathbf{Z}'_{31}+(f_3-\alpha)^2\mathbf{Z}'_{32}\Big)$
Desired symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{(f_2-\alpha)^{-1}},\overrightarrow{(f_3-\alpha)^{-1}}\Big)$	
Interference symbols appear along vectors $\overrightarrow{\Delta}\circ\Big(\overrightarrow{1},\overrightarrow{f_1-\alpha},\overrightarrow{(f_1-\alpha)^2},\overrightarrow{(f_1-\alpha)^3},\overrightarrow{f_2-\alpha},\overrightarrow{(f_2-\alpha)^2},\overrightarrow{(f_2-\alpha)^3},$ $\overrightarrow{f_3-\alpha},\overrightarrow{(f_3-\alpha)^2},\overrightarrow{(f_3-\alpha)^3}\Big)$	

Table 2.5: A summary of the XSTPIR scheme for $X=2,T=2,N=7$, showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

to be the zero polynomial it must be zero everywhere, but in that case, setting $\alpha-f_i=0$ for $i=1,2,3$ leads us to $c_1=c_2=c_3=0$. Privacy and security are guaranteed by the MDS coded independent noise terms mixed with the message and query symbols. The rate achieved is 3/7, which matches the asymptotic capacity for this setting.

2.7 Proof of Theorem 2.4

In Section 2.6 we presented an XSTPIR scheme for arbitrary X,T,N,K that achieves capacity as $K\rightarrow\infty$. Since the scheme also works for any K, a natural starting point for finite K settings is to apply the same scheme. A key insight here is that the rate achieved by the scheme improves as K decreases. Let us elaborate. Note that the query $Q_n^{[\theta]}$ that is sent to each server is uniformly distributed in $\mathbb{F}_q^{LK}$. Therefore, with probability $\frac{1}{q^{LK}}$, the query vector is the all zero vector. Whenever this happens, no download is needed from the server.

Thus, the average download is reduced by the factor $(1-1/q^{LK})$ and the rate achieved is expressed as follows.

Lemma 2.6. *The asymptotically capacity achieving XSTPIR scheme of Section 2.6 achieves the rate*

$$R=\left(1-\frac{1}{q^{KL}}\right)^{-1}\left(1-\left(\frac{X+T}{N}\right)\right) \tag{2.132}$$

for arbitrary X,L,K,N values, where $N>X+T$.

Note that $N\leq X+T$ is excluded as the degenerate setting where we already know the capacity for all parameters, according to Theorem 2.2. Remarkably, the rate in Lemma 2.6 depends on the message size L and the field size q used by the scheme. As presented, the scheme uses $q\geq L+N$ and $L=N-X-T$. So the achieved rate for finite K becomes

$$R=\left(1-\frac{1}{(2N-X-T)^{K(N-X-T)}}\right)^{-1}\left(1-\left(\frac{X+T}{N}\right)\right). \tag{2.133}$$

Consider the simplest non-trivial setting of interest, i.e., the setting for Theorem 2.4, where $T=X=1$, $N=3$ and K is arbitrary. The scheme of Section 2.6 uses $L=1,q\geq 4$, so the rate achieved for arbitrary K is

$$R=\frac{1}{3}\left(1-\frac{1}{4^K}\right)^{-1}. \tag{2.134}$$

However, note that if the field size could be reduced to $q=2$, then the rate achieved by the scheme would become

$$\frac{1}{3}\left(1-\frac{1}{2^K}\right)^{-1}=\frac{2}{3}\left(\frac{1-\frac{1}{2}}{1-\frac{1}{2^K}}\right)=\frac{2}{3}\left(1+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{K-1}}\right)^{-1}$$

which matches the capacity upper bound from Theorem 2.1. Surprisingly, this can be done with some modification to the structure of the scheme, as explained below.

Suppose each message $W_k, k \in [K]$ consists of $L=1$ symbol (bit) from $\mathbb{F}_2$. Let $\mathbf{W} = (W_1, W_2, \cdots, W_K)$ be a random row vector in $\mathbb{F}_2^{1\times K}$, containing all messages. Let $\mathbf{Z}$ and $\mathbf{Z}'$ be uniformly random noise vectors from $\mathbb{F}_2^{1\times K}$ and $\mathbb{F}_2^{K\times 1}$, that are used to guarantee data security and user privacy, respectively. The noise vectors are independent of each other and of the message vector and θ, i.e., $H(\mathbf{W},\mathbf{Z},\mathbf{Z}',\theta) = H(\mathbf{W}) + H(\mathbf{Z}) + H(\mathbf{Z}') + H(\theta)$. Let $\mathbf{Q}_\theta$ represent the θ^{th} column of $\mathbf{I}_K$ (the $K \times K$ identity matrix). Note that $\mathbf{W}\mathbf{Q}_\theta = W_\theta$ is the message desired by the user. The storage at the servers, the queries and the answers are listed below.

	Server 1	Server 2	Server 3
Storage S_n	$\mathbf{W}+\mathbf{Z}$	$\mathbf{W}+\mathbf{Z}\mathbf{B}$	$\mathbf{Z}$
Query $Q_n^{[\theta]}$	$\mathbf{Z}'$	$\mathbf{Q}_\theta+\mathbf{Z}'$	$(\mathbf{I}_K+\mathbf{B})\mathbf{Z}'+\mathbf{B}\mathbf{Q}_\theta$
Answer $A_n^{[\theta]}$	$\mathbf{W}\mathbf{Z}'+\mathbf{Z}\mathbf{Z}'$	$\mathbf{W}\mathbf{Q}_\theta+\mathbf{W}\mathbf{Z}'+\mathbf{Z}\mathbf{B}\mathbf{Z}'+\mathbf{Z}\mathbf{B}\mathbf{Q}_\theta$	$\mathbf{Z}\mathbf{Z}'+\mathbf{Z}\mathbf{B}\mathbf{Z}'+\mathbf{Z}\mathbf{B}\mathbf{Q}_\theta$

where $\mathbf{B}$ is a $K \times K$ deterministic binary matrix such that $\mathbf{B}$ and $\mathbf{I}_K+\mathbf{B}$ are both full rank. Any such choice of $\mathbf{B}$ will work for our scheme. The existence of such $\mathbf{B}$ is established in the following lemma whose proof appears in Appendix A.1.

Lemma 2.7. *For all $K \geq 2$, there exists a matrix $\mathbf{B} \in \mathbb{F}_2^{K\times K}$ such that $\mathbf{B}$ and $\mathbf{I}_K+\mathbf{B}$ are both invertible.*

Now, let us check the correctness, security and privacy of this scheme. The scheme is obviously correct because by adding the three answers shown in the table above, the user recovers W_θ. It is obviously secure because $\mathbf{B}$ is invertible, so $\mathbf{Z}\mathbf{B} \sim \mathbf{Z}$, is still uniform noise independent of $\mathbf{W}$. And similarly, it is also obviously private, because $\mathbf{I}_K+\mathbf{B}$ is also invertible, so $(\mathbf{I}_K+\mathbf{B})\mathbf{Z}' \sim \mathbf{Z}'$ is still uniform noise independent of $\mathbf{B}\mathbf{Q}_\theta$. Thus, surprisingly, we have achieved the capacity of XSTPIR for arbitrary K, when $X=T=1$ and $N=3$, completing the proof of Theorem 2.4. □

2.8 Discussion

The XSTPIR problem is timely due to the growing importance of privacy and security concerns in modern information storage and retrieval systems. It is a conceptually rich topic that reveals new insights into alignment of noise terms, dependence of coding and query structures, cost of symmetric security, significance of field size for the rate of information retrieval, etc.

Chapter 3

X-Secure T-Private Information Retrieval with Graph Based Replicated Storage

The problem of private information retrieval with graph-based replicated storage was recently introduced by Raviv, Tamo and Yaakobi. Its capacity remains open in almost all cases. In this chapter the asymptotic (large number of messages) capacity of this problem is studied along with its generalizations to include arbitrary T-privacy and X-security constraints, where the privacy of the user must be protected against any set of up to T colluding servers and the security of the stored data must be protected against any set of up to X colluding servers. A general achievable scheme for arbitrary storage patterns is presented that achieves the rate $(\rho_{\min} - X - T)/N$, where N is the total number of servers, and each message is replicated at least $\rho_{\min}$ times. Notably, the scheme makes use of a special structure inspired by dual Generalized Reed Solomon (GRS) codes. A general converse is also presented. The two bounds are shown to match for many settings, including symmetric storage patterns. Finally, the asymptotic capacity is fully characterized for the case without security constraints ($X=0$) for arbitrary storage patterns provided that each message is replicated no more than $T+2$ times. As an example of this result, consider PIR with arbitrary graph based storage ($T=1, X=0$) where every message is replicated at exactly 3 servers. For this 3-replicated storage setting, the asymptotic capacity is equal to $2/\nu_2(G)$ where $\nu_2(G)$ is the maximum size of a 2-matching in a storage graph $G[V,E]$. In this undirected graph, the vertices V correspond to the set of servers, and there is an edge $uv \in E$ between vertices u,v only if a subset of messages is replicated at both servers u and v.

3.1 Introduction

Most relevant to this chapter is the characterization in Chapter 2 of the asymptotic ($K \to \infty$) capacity of XSTPIR as $C_{\text{XSTPIR}} = 1-(X+T)/N$. Note that the XSTPIR setting includes as special case the TPIR setting, obtained by setting $X=0$, as well as the original PIR setting, obtained by setting $X=0$ and $T=1$. It is limited, however, by its assumption of fully replicated storage, i.e., all messages are stored by all servers, which can be burdensome for large data sets. Motivated by the preference for simple storage, Raviv, Tamo and Yaakobi in [89] introduced a graph based replicated storage model. Instead of full replication where every message is replicated at every server, graph based replication assumes that each message is replicated only among a subset of servers. This allows a graph representation where the vertices are the N servers and each message is represented by a hyperedge comprised of vertices (servers) where this message is replicated. Reference [89] primarily focuses on GTPIR, i.e., PIR with graph based replicated storage and T-privacy. An achievable scheme is proposed that achieves the rate $1/N$ as long as T is smaller than the replication factor of each message (the number of servers where the message is replicated), and is shown to be within a factor of 2 from optimality for some special cases. Reference [13] presents capacity achieving schemes for several cases of GPIR, i.e., GTPIR with 1-privacy where each server stores 2 messages. However, optimal GTPIR schemes remain unknown in almost all settings. Understanding the key ideas that constitute optimal PIR schemes under graph based replicated storage is our goal in this chapter.

The main contributions of this chapter are as follows. We study the asymptotic capacity of T-private and X-secure PIR with graph-based replicated storage, in short GXSTPIR. Recall that asymptotic capacity is quite meaningful for PIR because the number of messages is typically large, and the convergence of capacity to its asymptotic value tends to take place quite rapidly, see Chapter 2. GXSTPIR includes as special cases the settings of GTPIR [89], XSTPIR [57], TPIR [106] and basic PIR [102], and as such it presents a unified view of these settings. Our first result is an achievable scheme for GXSTPIR that achieves the rate $(\rho_{\min} - X - T)/N$ for arbitrary storage patterns provided every message is replicated at least $\rho_{\min}$ times. In addition to ideas like cross-subspace alignment, Reed-Solomon (RS) coded storage and RS coded queries that were previously used for XSTPIR in Chapter 2, a key novelty of our achievable scheme for GXSTPIR is how it creates and takes advantage of a structure inspired by dual Generalized Reed Solomon (GRS) codes. This is explained intuitively in Section 3.3.2. Our second contribution is a general converse bound for asymptotic capacity of GXSTPIR with arbitrary storage patterns. While the asymptotic capacity of GXSTPIR remains open in general, it is remarkable that our converse bound is tight in all settings where we are able to settle the capacity. In particular, the general achievable scheme matches the converse bound when the storage is symmetric, settling the asymptotic capacity for those settings [1]. For several examples with asymmetric storage, it turns out that the achievable scheme can be improved to match the converse bound by applying it only after eliminating[2] certain redundant servers. Thus, the asymptotic capacity for such cases is settled as well. In general however, with arbitrary graph based storage, more sophisticated achievable schemes may be obtained by combining our achievable scheme with ideas from private computation

[1] We refer the reader to Section 3.3 for the definition of symmetric storage.

[2] By “eliminating a server” for an achievable scheme, we mean using an achievable scheme that does not send any query to that server.

[107]. To illustrate this, we consider the GTPIR problem ($X=0$) where every message is replicated no more than $T+2$ times. As our final result, for this problem we fully settle the asymptotic capacity for arbitrary storage patterns. The asymptotic capacity depends strongly on the storage graph structure, and requires a private computation scheme on top of our general achievable scheme. As an example of this result, consider GPIR, i.e., PIR with arbitrary graph based storage ($T=1, X=0$) where every message is replicated at exactly 3 servers. For this 3-replicated storage setting, the asymptotic capacity is exactly equal to $2/\nu_2(G)$ where $\nu_2(G)$ is the maximum size of a 2-matching in a storage graph $G[V,E]$. In this storage graph, the vertices V correspond to the set of servers, and there is an edge $uv \in E$ between vertices u,v only if a subset of messages is replicated at both servers u and v. This is consistent with the intuition that storage graph properties must be essential to the asymptotic capacity of graph-based storage.

3.2 Problem Statement

We begin with a description of messages and storage structure. Based on the storage structure we will partition the set of messages into M subsets so that the messages in the same subset have the same storage structure. Define $\mathcal{W}=(\mathcal{W}_1,\mathcal{W}_2,\cdots,\mathcal{W}_M)$ where $\mathcal{W}_m, m \in [M]$, are disjoint message sets, each comprised of K_m messages,

$$\mathcal{W}_m=(W_{m,1},W_{m,2},\cdots,W_{m,K_m}). \tag{3.1}$$

Messages are independent, and each message is composed of L i.i.d. uniform symbols from $\mathbb{F}_q$, i.e.,

$$H(W_{m,k})=H(W_{m,k}(1),W_{m,k}(2),\cdots,W_{m,k}(L))=L, \qquad \forall m \in [M], k \in [K_m] \tag{3.2}$$

$$H(W_{1,1},\cdots,W_{M,K_M})=\sum_{m=1}^{M}K_mL, \tag{3.3}$$

in q-ary units. There are a total of N servers. Corresponding to $\mathcal{W}=(\mathcal{W}_1,\cdots,\mathcal{W}_M)$, let us define

$$\mathcal{R}=(\mathcal{R}_1,\cdots,\mathcal{R}_M), \tag{3.4}$$

$$\mathcal{R}_m=(\mathcal{R}_m(1),\cdots,\mathcal{R}_m(\rho_m)),\forall m\in[M], \tag{3.5}$$

$$\mathcal{R}_m(r)\in[N],\forall r\in[\rho_m], \tag{3.6}$$

where $\mathcal{R}_m,m\in[M]$ contains the servers, $\mathcal{R}_m(r)\in[N]$ that store the m^{th} set of messages $\mathcal{W}_m$. Without loss of generality we will assume that the servers are listed in increasing order in each tuple $\mathcal{R}_m$. The cardinality of $\mathcal{R}_m$ is $|\mathcal{R}_m|=\rho_m$, which will be referred to as the replication factor for the messages in $\mathcal{W}_m$. The minimum replication factor is defined as

$$\rho_{\min}\triangleq\min_{m\in[M]}\rho_m. \tag{3.7}$$

It is important to note that the messages may not be directly replicated at the servers. Because of security constraints, each message $W_{m,k}\in\mathcal{W}_m$, is represented by a total of ρ_m *shares* (the nomenclature comes from secret-sharing), denoted $\overline{W}_{m,k}=\left(W_{m,k}^{(n)},n\in\mathcal{R}_m\right)$, such that the share $W_{m,k}^{(n)}$ is stored at Server n, for all $n\in\mathcal{R}_m$. Messages are independently secured and must be recoverable from their shares, as specified by the following constraints.

$$H\left(\overline{W}_{1,1},\cdots,\overline{W}_{M,K_m}\right)=\sum_{m\in[M],k\in[K_M]}H\left(\overline{W}_{m,k}\right), \tag{3.8}$$

$$H\left(W_{m,k}\mid\overline{W}_{m,k}\right)=0. \tag{3.9}$$

Let us define the index set of $\mathcal{W}_m$ that are stored at Server n, as

$$\mathcal{M}_n = \{m \in [M] \Big| \mathcal{R}_m \ni n\}. \tag{3.10}$$

The information stored at Server n is defined as

$$S_n = \left\{W_{m,k}^{(n)}, m \in \mathcal{M}_n, k \in [K_m]\right\}. \tag{3.11}$$

For example, suppose we have $M = 4$ message sets (each comprised of $K_m = 2$ messages), stored at $N = 4$ servers as shown.

Server 1	**Server** 2	**Server** 3	**Server** 4
$\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3$	$\mathcal{W}_1, \mathcal{W}_2$	$\mathcal{W}_2, \mathcal{W}_4$	$\mathcal{W}_1, \mathcal{W}_3, \mathcal{W}_4$

Then for this example,[3] we have,

$$\mathcal{M}_1 = \{1,2,3\}, \quad S_1 = \{W_{1,1}^{(1)}, W_{1,2}^{(1)}, W_{2,1}^{(1)}, W_{2,2}^{(1)}, W_{3,1}^{(1)}, W_{3,2}^{(1)}\}, \quad \mathcal{R}_1 = (1,2,4), \quad \rho_1 = 3, \tag{3.12}$$

$$\mathcal{M}_2 = \{1,2\}, \quad S_2 = \{W_{1,1}^{(2)}, W_{1,2}^{(2)}, W_{2,1}^{(2)}, W_{2,2}^{(2)}\}, \quad \mathcal{R}_2 = (1,2,3), \quad \rho_2 = 3, \tag{3.13}$$

$$\mathcal{M}_3 = \{2,4\}, \quad S_3 = \{W_{2,1}^{(3)}, W_{2,2}^{(3)}, W_{4,1}^{(3)}, W_{4,2}^{(3)}\}, \quad \mathcal{R}_3 = (1,4), \quad \rho_3 = 2, \tag{3.14}$$

$$\mathcal{M}_4 = \{1,3,4\}, \quad S_4 = \{W_{1,1}^{(4)}, W_{1,2}^{(4)}, W_{3,1}^{(4)}, W_{3,2}^{(4)}, W_{4,1}^{(4)}, W_{4,2}^{(4)}\}, \quad \mathcal{R}_4 = (3,4), \quad \rho_4 = 2, \tag{3.15}$$

and $\rho_{\min} = 2$.

[3] Incidentally, our results will show that as $K_m \to \infty$, for this example $C_\infty = 1/3$, and Server 2 is redundant.

The X-secure constraint, $0 \leq X \leq N$, requires that any X (or fewer) colluding servers learn nothing about the messages.

$$[X\text{-Security}] \qquad I(S_{\mathcal{X}};\mathcal{W})=0, \qquad \forall \mathcal{X} \subset [N], |\mathcal{X}| \leq X. \tag{3.16}$$

$X=0$ represents the setting without security constraints. If $X=0$, then no secret sharing is needed, so each share of a message is the message itself,

$$X=0 \Longrightarrow W_{m,k}^{(n)}=W_{m,k}, \qquad \forall n \in \mathcal{R}_m. \tag{3.17}$$

This completes the description of the messages and the storage at the N servers. Next, let us describe the private information retrieval aspect.

The user desires the message $W_{\mu,\kappa}$, where the indices μ and κ are chosen privately and uniformly by the user from $\mu \in [M], \kappa \in [K_\mu]$, respectively. In order to retrieve his desired message, the user generates N queries, $Q_1^{[\mu,\kappa]}, Q_2^{[\mu,\kappa]}, \ldots, Q_N^{[\mu,\kappa]}$, and sends the n^{th} query, $Q_n^{[\mu,\kappa]}$ to the n-th server. The user has no prior knowledge of the message realizations,

$$I\left(S_{[N]}\ ;\ \mu,\kappa,Q_{[N]}^{[1,1]},\cdots,Q_{[N]}^{[M,K_M]}\right)=0. \tag{3.18}$$

A T-private scheme, $1 \leq T \leq N$, requires that any T (or fewer) colluding servers learn nothing about (μ,κ).

$$[T\text{-Privacy}] \qquad I\left(Q_{\mathcal{T}}^{[\mu,\kappa]}\ ;\ \mu,\kappa\right)=0, \qquad \forall \mathcal{T} \subset [N], |\mathcal{T}| \leq T. \tag{3.19}$$

Upon receiving the query $Q_n^{[\mu,\kappa]}$, the n-th server generates an answer string $A_n^{[\mu,\kappa]}$, which is a function of the query $Q_n^{[\mu,\kappa]}$ and its stored information S_n.

$$H\left(A_n^{[m,k]} \mid Q_n^{[m,k]}, S_n\right)=0, \qquad \forall m \in [M], k \in [K_m]. \tag{3.20}$$

The correctness constraint guarantees that from all the answers, the user is able to decode the desired message $W_{\mu,\kappa}$,

$$\text{[Correctness]} \qquad H\left(W_{\mu,\kappa} \mid A_{[N]}^{[\mu,\kappa]}, Q_{[N]}^{[\mu,\kappa]}, \mu, \kappa\right) = 0. \tag{3.21}$$

The rate of a GXSTPIR scheme is defined by the number of q-ary symbols of desired message that are retrieved per downloaded q-ary symbol,

$$R = \frac{H(W_{\mu,\kappa})}{\sum_{n\in[N]} H\left(A_n^{[\mu,\kappa]}\right)} = \frac{L}{D}, \tag{3.22}$$

where $D = \sum_{n\in[N]} H\left(A_n^{[\mu,\kappa]}\right)$ is the expected[4] total number of q-ary symbols downloaded by the user from all servers. The capacity of GXSTPIR, denoted as $C(N,X,T,\mathcal{W},\mathcal{S})$, is the supremum of R across all feasible schemes. In this chapter we are interested in the setting where each subset of messages is comprised of a large number of messages. Specifically, we wish to characterize the asymptotic capacity, as $K_m \to \infty$ for all $m \in [M]$. In order to have K_m approach infinity together for all $m \in [M]$, let us define,

$$K_m = \lceil \chi_m K \rceil, \tag{3.23}$$

so that $\chi_m, m \in [M]$ are fixed constants, while K approaches infinity. Then the asymptotic capacity is defined as

$$C_\infty = \lim_{K\to\infty} C(N,X,T,\mathcal{W},\mathcal{S}). \tag{3.24}$$

Note that the number of message sets, M, and the storage pattern $\mathcal{R}$ remain unchanged, while K_m, i.e., the number of messages in each $\mathcal{W}_m$ approaches infinity.

[4] While the achievable schemes used in this chapter only download a deterministic number of bits from each server, note that our capacity formulation allows schemes for which the number of bits downloaded from each server may be random. This means that our capacity results cannot be improved upon by schemes that download a random number of bits from each server.

3.3 Results

Our first result is a general achievability argument that provides us a lower bound on the asymptotic capacity of GXSTPIR.

Theorem 3.1. *The asymptotic capacity of GXSTPIR is bounded below as follows,*

$$C_\infty \geq \frac{\rho_{\min} - X - T}{N}. \tag{3.25}$$

The proof of Theorem 3.1 appears in Section 3.4. From a practical standpoint, it is worth noting that while the achievable rate in (3.25) does not depend on the storage structure beyond just the minimum replication factor $\rho_{\min}$, the achievable scheme does require that the user be aware of the storage structure for the construction of queries that are sent to the servers. From a technical standpoint, the most interesting aspect of the proof is the use of a structure inspired by dual GRS codes, that is intuitively explained in Section 3.3.2. Another interesting aspect of Theorem 3.1 is that applying it to a subset of servers (by eliminating the rest) may produce a higher achievable rate than if all servers were used. Therefore, in order to find the best achievable rate guaranteed by Theorem 3.1 we must choose the best subset of servers. Example 4 in Section 3.3.1 illustrates this idea. As a final remark, let us reiterate that the focus of this chapter is on settings with large number of messages. Indeed for smaller values of K and M the schemes presented in [89, 13] can achieve better rates than the scheme presented in Section 3.4.

Our next result is a converse argument that holds for arbitrary storage patterns. Recall that $D_n = H(A_n^{[\mu,\kappa]})/L$ is the normalized download from Server n.

Theorem 3.2. *The asymptotic capacity of GXSTPIR is bounded above as follows,*

$$C_\infty \leq \begin{cases} 0, & \rho_{\min} \leq X+T \\ \max_{(D_1,\cdots,D_N)\in\mathcal{D}} \quad (D_1+D_2+\cdots+D_N)^{-1}, & \rho_{\min} > X+T \end{cases} \tag{3.26}$$

and $\mathcal{D}$ is defined as

$$\mathcal{D} \triangleq \left\{ (D_1,\cdots,D_N) \in \mathbb{R}_+^N \Big| \sum_{n\in\mathcal{R}'_m} D_n \geq 1, \forall m \in [M], \forall \mathcal{R}'_m \subset \mathcal{R}_m, |\mathcal{R}'_m| = |\mathcal{R}_m| - X - T \right\}. \tag{3.27}$$

The proof of Theorem 3.2 appears in Section 3.5. Since the asymptotic capacity is zero for $\rho_{\min} \leq X+T$, in the remainder of this section we will assume that $\rho_{\min} > X+T$. Note that since our focus is on settings with asymptotically large number of messages, our coding schemes achieve rate zero for cases where the asymptotic capacity is zero.

Remark 3.1. *Note that* (3.27) *implies that the total normalized download from any $\rho_m - X - T$ servers in $\mathcal{R}_m$ must be at least* 1*. A simple averaging argument implies that the total normalized download from all ρ_m servers in any $\mathcal{R}_m$ must be at least $\rho_m/(\rho_m - X - T)$.*

The general lower bound in Theorem 3.1 is in closed form and the general upper bound in Theorem 3.2 is essentially a linear program, so for arbitrary settings it is possible to evaluate both to check if they match (provided the parameter values are not too large to be computationally feasible). Conceptually, the condition for them to match may be understood as follows. Consider a hypergraph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with the set of vertices $\mathcal{V} = [N]$ representing the N servers, and the set of hyperedges $\mathcal{E}$ such that $e \in \mathcal{E}$ if and only if $\exists m \in [M]$ such that $e \subset \mathcal{R}_m$ and $|e| = |\mathcal{R}_m| - X - T$. For this graph, hyperedges $e \in \mathcal{E}$, with corresponding weights $x_e \in \mathbb{R}_+$, are said to form a fractional matching if for every vertex $v \in \mathcal{V}$ the total weight of the edges that include v is less than or equal to 1. The largest possible total weight of a fractional matching is called the fractional matching number of $\mathcal{G}$ [93]. The relationship between the optimal converse bound from Theorem 3.2 on the total normalized download, i.e., $\min_{\mathcal{D}}(D_1+\cdots+D_N)$ and the fractional matching number of $\mathcal{G}[\mathcal{V},\mathcal{E}]$ is characterized in the following lemma.

Lemma 3.1. *The optimal value of total normalized download,* $\min_{\mathcal{D}}(D_1+D_2+\cdots+D_N)$*, in Theorem 3.2 is equal to the fractional matching number of* $\mathcal{G}[\mathcal{V},\mathcal{E}]$*.*

The proof of Lemma 3.1 is presented in Appendix B.1. From Lemma 3.1, the following corollary immediately follows.

Corollary 3.1. *The lower bound of Theorem 3.1 matches the upper bound of Theorem 3.2 if and only if the fractional matching number of* $\mathcal{G}(\mathcal{V},\mathcal{E})$ *is equal to* $\frac{N}{\rho_{\min}-X-T}$*. For all such cases, the asymptotic capacity* $C_\infty=(\rho_{\min}-X-T)/N$*.*

Next let us identify some interesting special cases of Corollary 3.1.

Let $\mathcal{R}_{\mathcal{M}'}$ be a collection of the sets $\mathcal{R}_m, m\in\mathcal{M}'\subset[M]$. We define $\mathcal{R}_{\mathcal{M}'}$ to be an exact b-cover of $[N]$ if $\rho_m=\rho_{\min}$ for all $m\in\mathcal{M}'$, and every element of $[N]$ is contained in exactly b sets in $\mathcal{R}_{\mathcal{M}'}$. It follows that the asymptotic capacity $C_\infty=(\rho_{\min}-X-T)/N$ if there exists an exact b-cover for some $b\in\mathbb{Z}_+$. This is easily seen because for each $\mathcal{R}_m$ in $\mathcal{R}_{\mathcal{M}'}$ we have the bound $\sum_{n\in\mathcal{R}_m}D_n\geq\rho_{\min}/(\rho_{\min}-X-T)$ according to (3.27). Adding all these bounds we obtain the desired converse bound $b\sum_{n\in[N]}D_n\geq(bN/\rho_{\min})(\rho_{\min}/(\rho_{\min}-X-T))$, i.e., $\sum_{n\in[N]}D_n\geq N/(\rho_{\min}-X-T)$, which is achievable according to Theorem 3.1.

(**Symmetric Storage)** As a special case that is of particular interest, define a symmetric storage setting as one where (after some permutation of message and server indices) for all $m\in[M]$, $\mathcal{R}_m=(\rho m+1,\rho m+2,\cdots,\rho m+\rho_{\min})$. Here, $\rho\leq\rho_{\min}$ and server indices are interpreted modulo N, e.g., Server $N+1$ is the same as Server 1. Furthermore, $b=M\rho_{\min}/N$ is an integer value. Then any symmetric storage setting thus defined has asymptotic capacity

$C_\infty = (\rho_{\min} - X - T)/N$ because the storage sets form an exact b-cover.

Based on these observations, here are some examples of storage patterns where the asymptotic capacity is $C_\infty = (\rho_{\min} - X - T)/N$.

1. $\mathcal{R} = ((1,2),(2,3),(3,1))$ which is a symmetric storage setting (forms an exact 2 cover).
2. $\mathcal{R} = ((1,2,3),(3,4,5),(5,1,2),(2,3,4),(4,5,1))$ which is a symmetric storage setting (forms an exact 3-cover).
3. $\mathcal{R} = ((1,2),(2,3),(3,1),(4,5),(5,6),(6,4))$ because it forms an exact 2 cover.
4. $\mathcal{R} = ((1,2,3),(4,5,6),(i,j,k),(a,b,c,d))$ for arbitrary $\{i,j,k\},\{a,b,c,d\} \subset [N] = [6]$ because it contains an exact 1-cover, $\mathcal{R}_{\mathcal{M}'} = \{(1,2,3),(4,5,6)\}$.
5. $\mathcal{R} = ((1,2,3),(3,4,1),(2,5,6),(4,5,6),(1,3,6),(1,2,5,6))$ because it contains an exact 2-cover of $[N] = [6]$ in $\mathcal{R}_{\mathcal{M}'} = \{(1,2,3),(3,4,1),(2,5,6),(4,5,6)\}$.

While the existence of an exact b-cover for some positive integer b is *sufficient* to guarantee that the asymptotic capacity is $C_\infty = (\rho_{\min} - X - T)/N$, it is not a *necessary* condition. Examples 1 and 2 in Section 3.3.1 show such settings.

On the other hand, it is also easy to see that the lower bound of Theorem 3.1 and the upper bound of Theorem 3.2 do not always match. Remarkably, in all such cases that we have been able to settle so far, it is the upper bound that is tight, and the achievability that needs to be improved. In many cases, such as Example 4 in Section 3.3.1, an improved achievability result is found easily by eliminating a redundant server before applying Theorem 3.1. However, more sophisticated achievable schemes may be required in general.

Our final result emphasizes this point by settling the asymptotic capacity of GTPIR, i.e., T-private information retrieval with arbitrary graph based storage and no security constraints

$(X=0)$, provided each message is replicated no more than $(T+2)$ times. Because this result deals with arbitrary storage patterns, for its precise statement we will need the following definitions that follow the convention of Schrijver [93].

Definition 3.1. *Define $G=(V,E)$ as a simple undirected graph with vertices $V=[N]$ corresponding to the N servers, and with edges $uv \in E$ if and only if $\{u,v\} \subset \mathcal{R}_m$ for some $m \in [M]$.*

Definition 3.2. *For a set $U \subset V$, we define $G[U]$ as the induced subgraph of G whose vertex set is U and whose edge set, denoted $E[U]$ consists of all edges $uv \in E$ such that $u,v \in U$.*

Definition 3.3. *A set $U \subset V$ is called a stable set (also called independent set) if there are no edges between any two members of U.*

Definition 3.4. *For $U \subset [N]$, define $\mathcal{N}(U)$ as the set of vertices in $V \backslash U$ that are neighbors of vertices in U.*

Definition 3.5. *Define $\delta(n)$ as the set of edges incident with vertex n.*

Definition 3.6. *A function $x: E \to \mathbb{Z}_+$ is denoted as a vector $x \in \mathbb{Z}_+^E$. A function $y: V \to \mathbb{Z}_+$ is similarly denoted as a vector $y \in \mathbb{Z}_+^V$. The size of a vector is defined as the sum of its entries.*

Definition 3.7. *For any $x \in \mathbb{Z}_+^E$, and $F \subset E$, define $x(F) = \sum_{f \in F} x(f)$.*

Definition 3.8. *For a positive integer b, a b-matching in G is defined as a vector $x \in \mathbb{Z}_+^E$ satisfying $x(\delta(v)) \leq b$ for each vertex $v \in V$. The maximum size of a b-matching in G is defined as $\nu_b(G)$.*

Definition 3.9. *Define $\mathcal{N}_r$ as the set of servers that do not store any messages that are replicated fewer than r times.*

$$\mathcal{N}_r \triangleq \{n \in [N] \Big| m \in \mathcal{M}_n \implies \rho_m \geq r\}. \tag{3.28}$$

It is worthwhile to recall that from basic results in graph theory (see Chapter 30, Section 30.1 of Schrijver [93]), it is known that

$$\nu_2(G)=\min\{|V\backslash U|+|\mathcal{N}(U)| \;\Big|\; U\subset V, \text{ and } U \text{ is a stable set}\}. \tag{3.29}$$

With this we are ready to state our final result.

Theorem 3.3. *The asymptotic capacity of GTPIR with $\rho_m\leq T+2$ for all $m\in[M]$, i.e., when each message set is replicated no more than $(T+2)$ times, is*

$$C_\infty=\begin{cases}0, & \rho_{\min}\leq T\\ \frac{2}{\nu_2(G[\mathcal{N}_{T+2}])+2|\mathcal{N}_{T+1}|}, & \rho_{\min}>T\end{cases}. \tag{3.30}$$

The proof of Theorem 3.3 appears in Section 3.6. While the converse bound for Theorem 3.3 follows directly from the general converse bound in Theorem 3.2, the achievability goes beyond the scheme of Theorem 3.1, to involve a limited generalization to private computation that is presented in Section 3.4.3. As an interesting special case of Theorem 3.3, note that if all messages are $T+2$ replicated, i.e., $\mathcal{N}_{T+1}$ is an empty set, then the asymptotic capacity is exactly $2/\nu_2(G)$.

Remark 3.2. *In general, for arbitrary positive integer b, any bounds on b-matching will result in a corresponding lower bound of $\sum_{n\in[N]}D_n$. However, since the converse bounds for 2-matchings are found to be tight, there is no need to pursue $b\neq 2$, because no more bounds are needed. For larger ρ_m, perhaps similar results are possible in the hypergraph version, but we have not been able to find meaningful generalizations along these lines.*

3.3.1 Examples

Let us consider a few more examples to illustrate our results. For these examples we set $X=0,T=1$ for simplicity, but similar examples are easily constructed for $X>0,T>1$ as well.

1. Consider $M=3$ message sets, stored at $N=4$ servers according to the replication pattern $\mathcal{R}_1=(1,2,4)$, $\mathcal{R}_2=(1,2,3)$, $\mathcal{R}_3=(1,3,4)$. Since every message is 3-replicated, according to Theorem 3.1 we have $C_\infty \geq 2/4=1/2$. For the converse we note that $\mathcal{R}_1 \implies D_1+D_2\geq 1$, $\mathcal{R}_2 \implies D_2+D_3\geq 1$, $\mathcal{R}_3 \implies D_3+D_4\geq 1, D_4+D_1\geq 1$, and adding these bounds gives us $D_1+D_2+D_3+D_4\geq 2$. Thus we have $C_\infty=1/2$ for this example. Note that this example does not contain an exact b-cover for any positive integer b, but the asymptotic capacity for this example is still $C_\infty=(\rho_{\min}-X-T)/N$.

2. Consider $M=3$ message sets stored at $N=5$ servers according to the replication pattern $\mathcal{R}_1=(1,3,4),\mathcal{R}_2=(3,4,5),\mathcal{R}_3=(2,3,5)$, so that every message is 3-replicated, but the storage is not symmetric, nor does it contain an exact b-cover. For the converse we note that $\mathcal{R}_1 \implies D_4+D_1\geq 1, D_1+D_3\geq 1$; $\mathcal{R}_3 \implies D_3+D_2\geq 1, D_2+D_5\geq 1$; $\mathcal{R}_2 \implies D_5+D_4\geq 1$; and combining these bounds gives us the converse bound as $C_\infty \leq \max_{\mathcal{D}} 1/(\sum_{n\in[5]} D_n)\leq 2/5$. Since $\rho_{\min}=3$, Theorem 3.1 shows that the rate $(\rho_{\min}-X-T)/N=2/5$ is achievable, so that $C_\infty=2/5$ for this example.

3. Consider $M=3$ message sets stored at $N=5$ servers according to the replication pattern $\mathcal{R}_1=(1,3,4),\mathcal{R}_2=(1,3,4,5),\mathcal{R}_3=(2,3,5)$, so that messages in $\mathcal{W}_2$ are 4-replicated while those in $\mathcal{W}_1,\mathcal{W}_3$ are only 3-replicated. For the converse we note that $\mathcal{R}_1 \implies D_1+D_3\geq 1, D_3+D_4\geq 1, D_4+D_1\geq 1$; while $\mathcal{R}_3 \implies 2D_2+2D_5\geq 2$. Adding them up we have the bound $D_1+D_2+D_3+D_4+D_5\geq 5/2$, which gives us the converse bound $C_\infty\leq 2/5$. Since $\rho_{\min}=3$, the lower bound from Theorem 3.1 is also $2/5$, so that $C_\infty=2/5$ for this example. Note that we could eliminate any one element from $\mathcal{R}_2$ so

that messages in $\mathcal{W}_2$ are also only 3-replicated, but that would not change the asymptotic capacity. Or we could add one more element to $\mathcal{R}_2$ so that messages in $\mathcal{W}_2$ are replicated at every server, and that would also not change the capacity. Thus, this example illustrates redundant storage.

4. Consider $M=2$ message sets stored at $N=5$ servers according to the replication pattern $\mathcal{R}_1=(1,2,3,4)$, $\mathcal{R}_2=(2,3,4,5)$, so that each message is 4-replicated. The converse from Theorem 3.2 says $C_\infty \leq 2/3$, which corresponds to $D_1=D_5=0, D_2=D_3=D_4=1/2$, but since $\rho_{\min}=4$, Theorem 3.1 applied directly only proves the achievability of rate $(\rho_{\min}-X-T)/N=3/5$ which does not match the converse bound. However, note that if we eliminate Server 1 and Server 5, then we are left with the same[5] $M=2$ message sets stored at $N'=3$ servers according to the replication pattern $\mathcal{R}'_1=(2,3,4), \mathcal{R}'_2=(2,3,4)$, for which $\rho'_{\min}=3$, and Theorem 3.1 shows that the rate $(\rho'_{\min}-X-T)/N'=2/3$ is achievable, which indeed matches the converse bound. Thus, the asymptotic capacity for this example is $C_\infty=2/3$. The example shows that achievable rates may be improved by eliminating redundant servers.

5. Consider $M=4$ message sets stored at $N=5$ servers according to the storage pattern $\mathcal{R}_1=(1,2,3), \mathcal{R}_2=(2,3,4), \mathcal{R}_3=(1,3,5), \mathcal{R}_4=(2,4)$, so that messages in $\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3$ are 3-replicated, while messages in $\mathcal{R}_4$ are 2-replicated, and $\rho_{\min}=2$. The achievable scheme from Theorem 3.1 achieves a rate 1/5, however Theorem 3.3 builds upon that scheme to achieve the rate 2/7 which also matches the converse. Thus, for this setting, the capacity is settled by Theorem 3.3 as $C_\infty=2/7$.

6. Consider $M=5$ message sets stored at $N=8$ servers according to the storage pattern $\mathcal{R}_1=(1,2,3), \mathcal{R}_2=(1,3,4), \mathcal{R}_3=(4,5,7), \mathcal{R}_4=(4,6,7), \mathcal{R}_5=(7,8)$. The capacity for this case is settled by Theorem 3.3 as 2/9. To explicitly see the converse bound, note that in (3.27) $\mathcal{R}_1 \implies D_1+D_2+D_3 \geq 3/2$; $\mathcal{R}_5 \implies D_7 \geq 1, D_8 \geq 1$; and $\mathcal{R}_3 \implies D_4+D_5 \geq 1$.

[5] Note that while some servers may be eliminated (i.e., not used) by an achievable scheme, the message sets cannot be reduced because the achievable scheme must still work for all messages.

Adding these bounds we have $D_1+D_2+D_3+D_4+D_5+D_7+D_8 \geq 9/2$, which implies that asymptotically the total normalized download $D \geq 9/2$ and the converse bound follows. The graph representation for this setting, $G(V,E)$ is shown in Figure 3.1. Vertices in $\mathcal{N}_3 = \{1,2,3,4,5,6\}$ are shown with a red border, while vertices in $\mathcal{N}_2 = \{7,8\}$ are shown with a black border. The maximum size of a 2-matching on $G[\mathcal{N}_3]$ is 5, corresponding to the 5 edges shown in red. Alternatively, it corresponds to the choice of $U = \{5,6\} \subset \mathcal{N}_3$ in (3.29). Note that while U has 2 neighbors in G, i.e., $\mathcal{N}(U) = \{4,7\}$, it has only 1 neighbor in $\mathcal{N}_3$, i.e., $\mathcal{N}(U) \cap \mathcal{N}_3 = \{4\}$. Therefore, $\nu_2(G[\mathcal{N}_3]) + 2|\mathcal{N}_2| = |\mathcal{N}_3 \setminus U| + |\mathcal{N}(U) \cap \mathcal{N}_3| + 2|\mathcal{N}_2| = 4+1+2(2) = 9$. Achievability follows by the scheme presented in the proof of Theorem 3.3, downloading a symbol from each of $[N] \setminus U = \{1,2,3,4,7,8\}$, and downloading another symbol from each of $\mathcal{N}(U) \cup \mathcal{N}_2 = \{4,7,8\}$ according to a private computation scheme described in Section 3.4.3, for a total download of 9 symbols from which 2 desired symbols are retrieved.

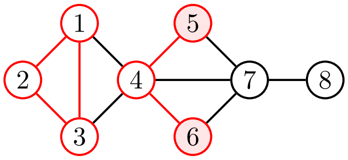


Figure 3.1: The graph $G[V,E]$ for Example 6.

3.3.2 Solution Structure inspired by Dual GRS Codes

The most interesting aspect of the achievable scheme in Theorem 3.1 is a generalized query and storage structure that is inspired by dual GRS codes. Since the storage and query structure for XSTPIR in [57] was based on RS codes, the generalization to GRS code structure for GXSTPIR is somewhat serendipitous (note that the G in GRS codes is not automatically associated with the G in GXSTPIR which stands for Graph based replicated storage). It is also surprisingly effective, as explained intuitively in this section.

Before discussing how GRS codes are a part of the solution, let us illustrate the nature of the problem with a simple example. Let us consider a very basic setting, where we have $M=4$ subsets of messages, $N=4$ servers, and $\forall m\in[M]$, we have $\mathcal{R}_m=[N]\setminus\{m\}$, i.e., messages in $\mathcal{W}_m$ are stored at all servers except Server m. Let $V_m, m\in[M]$ be four vectors in $\mathbb{F}$, each of size $N\times 1$, such that the vector V_m has a zero in its m^{th} coordinate (reflecting the fact that messages in $\mathcal{W}_m$ are not stored at Server m) and all other coordinates are non-zero. Then, as we will explain shortly, the rank of the matrix $[\mathbf{V}_1,\mathbf{V}_2,\mathbf{V}_3,\mathbf{V}_4]$ reflects the number of dimensions occupied by interference, i.e., downloaded symbols that are undesired. For example, suppose we are operating in $\mathbb{F}_5$ and we choose,

$$\mathbf{V}=[V_1,V_2,V_3,V_4]=\begin{bmatrix}0\ 1\ 1\ 1\\ 1\ 0\ 3\ 2\\ 1\ 2\ 0\ 4\\ 1\ 3\ 1\ 0\end{bmatrix} \tag{3.31}$$

which has rank 2. Then this choice corresponds to a scheme where interference occupies $\text{rank}(\mathbf{V})=2$ out of the $N=4$ dimensions, leaving the remaining 2 dimensions available for retrieving desired message symbols. To see this explicitly, suppose each message is comprised of $L=2$ symbols, $W_{m,k}=(W_{m,k}(1),W_{m,k}(2))$ in $\mathbb{F}_5$, and the user desires the message $W_{\mu,\kappa}\in\mathcal{W}_\mu$. The download from the n^{th} server is the n^{th} row of the following $N\times 1$ vector.

$$\mathbf{V}=\left(\sum_{k\in[K_1],\ell\in[L]}W_{1,k}(\ell)Z_{1,k,(\ell)}\right)\mathbf{V}_1+\left(\sum_{k\in[K_2],\ell\in[L]}W_{2,k}(\ell)Z_{2,k,(\ell)}\right)\mathbf{V}_2$$

$$+\left(\sum_{k\in[K_3],\ell\in[L]}W_{3,k}(\ell)Z_{3,k,(\ell)}\right)\mathbf{V}_3+\left(\sum_{k\in[K_4],\ell\in[L]}W_{4,k}(\ell)Z_{4,k,(\ell)}\right)\mathbf{V}_4 \tag{3.32}$$

$$+W_{\mu,\kappa}(1)\mathbf{F}^{[\mu,\kappa]}_{(1)}+W_{\mu,\kappa}(2)\mathbf{F}^{[\mu,\kappa]}_{(2)} \tag{3.33}$$

The vectors $\mathbf{F}^{[\mu,\kappa]}_{(1)},\mathbf{F}^{[\mu,\kappa]}_{(2)}$ are two 4×1 vectors, called demand vectors that help retrieve the desired message symbols. Due to storage constraints, the demand vectors $\mathbf{F}_{(1)}^{[\mu,\kappa]}$,$\mathbf{F}_{(2)}^{[\mu,\kappa]}$ must also have zeros in the coordinates where $\mathbf{V}_\mu$ has zeros. The $Z_{k,m,(\ell)}$ random variables are i.i.d. uniform noise terms added to hide the demand vectors contained in the query sent to each server, thus ensuring privacy of user's demand. The demand vectors, which carry the 2 desired message symbols must be linearly independent of $\mathbf{V}_1,\mathbf{V}_2,\mathbf{V}_3,\mathbf{V}_4$ which carry only interference. To retrieve his desired message, the user projects V into the 2 dimensional null space of $\mathbf{V}_1,\mathbf{V}_2,\mathbf{V}_3,\mathbf{V}_4$, where all interference disappears and only the two desired signal dimensions remain, from which the 2 desired symbols are retrieved. The rate achieved by this scheme is $2/4=1/2$ which is also the asymptotic capacity for this setting (converse follows from Theorem 3.2).

From this example, it is clear that the problem is related to min-rank of the V matrix subject to constraints on which terms take zero or non-zero values. These constraints are affected not only by the given storage structure, but also from the possibility of redundant servers[6] as well as privacy and correctness constraints, e.g., because demand vectors must share the same structure to ensure privacy. Evidently, PIR with graph based storage is connected to other problems such as index coding, where also min-rank is important [15]. For arbitrary storage patterns such min-rank problems can be difficult to solve in general. However, now let us consider what happens if every message is replicated the same number of times, $|\mathcal{R}_m|=\rho_m=\rho_{\min}$ for all $m\in[M]$. As will be shown in the proof of Theorem 3.3, even if replication factors vary across messages, schemes for such settings may use the constant-replication-factor schemes as their essential building blocks. Thus, the constant-replication-factor setting is of fundamental significance. It is also the setting where we exploit the structure of dual GRS codes.

For simplicity we will only consider a setting with $X=0$ and $T=1$. Consider such a setting with an arbitrary number of message sets M, with $N=5$ servers, constant-replication-factor

[6] As illustrated by examples in Section 3.3.1 the solution may be further optimized on storage structure by ignoring redundant storage.

$\rho_{min}=3$, and an example of an arbitrary storage pattern that satisfies these constraints (5 servers, every message replicated at 3 servers) reflected in the structure of the following $\mathbf{V}$ matrix.

$$\mathbf{V}=\begin{array}{r|ccccc|} & m{=}1 & m{=}2 & m{=}3 & \cdots & m{=}M \\ \text{Server 1} & v_{1,1} & 0 & v_{3,1} & \cdots & v_{M,1} \\ \text{Server 2} & 0 & v_{2,2} & v_{3,2} & \cdots & 0 \\ \text{Server 3} & v_{1,3} & v_{2,3} & 0 & \cdots & v_{M,3} \\ \text{Server 4} & v_{1,4} & 0 & v_{3,4} & \cdots & 0 \\ \text{Server 5} & 0 & v_{2,5} & 0 & \cdots & v_{M,5} \end{array} \tag{3.34}$$

Note that the m^{th} column has exactly $\rho_m=3$ non-zero entries corresponding to the 3 servers that store the messages in $\mathcal{W}_m$. The structure of each column is arbitrary, fixed by the given storage pattern, but each column must have exactly 3 non-zero entries. For this setting, it turns out that regardless of the value of M, it is possible to choose non-zero values for $v_{m,n}$ such that the rank of this matrix is not more than 3, i.e., all interference can be limited to 3 dimensions. This is done as follows. Let β_n be distinct non-zero constants for all $n\in[N]$. Furthermore, let us define,

$$v_{m,n}=\left(\prod_{n'\in\mathcal{R}_m\setminus\{n\}}(\beta_n-\beta_{n'})\right)^{-1} \tag{3.35}$$

Based on dual GRS codes (see Lemma B.2), it turns out that this choice of $v_{m,n}$ ensures that

$$\sum_{n\in\mathcal{R}_m} v_{m,n}\beta_n^j=0 \tag{3.36}$$

for all $j\in\{0,1,\cdots,\rho_{\min}-2\}$. For this example, since $\rho_{\min}=3$, it means that $\forall m\in[M]$,

$\sum_{n\in\mathcal{R}_m} v_{m,n} = 0$, and $\sum_{n\in\mathcal{R}_m} v_{m,n}\beta_n = 0$. Writing this out explicitly, we have

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 \end{bmatrix} \begin{bmatrix} \frac{1}{(\beta_1-\beta_3)(\beta_1-\beta_4)} & 0 & \frac{1}{(\beta_1-\beta_2)(\beta_1-\beta_4)} & \cdots & \frac{1}{(\beta_1-\beta_3)(\beta_1-\beta_5)} \\ 0 & \frac{1}{(\beta_2-\beta_3)(\beta_2-\beta_5)} & \frac{1}{(\beta_2-\beta_1)(\beta_2-\beta_4)} & \cdots & 0 \\ \frac{1}{(\beta_3-\beta_1)(\beta_3-\beta_4)} & \frac{1}{(\beta_3-\beta_2)(\beta_3-\beta_5)} & 0 & \cdots & \frac{1}{(\beta_3-\beta_1)(\beta_3-\beta_5)} \\ \frac{1}{(\beta_4-\beta_1)(\beta_4-\beta_3)} & 0 & \frac{1}{(\beta_4-\beta_1)(\beta_4-\beta_2)} & \cdots & 0 \\ 0 & \frac{1}{(\beta_5-\beta_2)(\beta_5-\beta_3)} & 0 & \cdots & \frac{1}{(\beta_5-\beta_1)(\beta_5-\beta_3)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{3.37}$$

which is easily verified because for any $n_1, n_2, n_3 \in [N]$,

$$v_{m,n_1} + v_{m,n_2} + v_{m,n_3} = \frac{(\beta_{n_2}-\beta_{n_3}) + (\beta_{n_3}-\beta_{n_1}) + (\beta_{n_1}-\beta_{n_2})}{(\beta_{n_1}-\beta_{n_2})(\beta_{n_1}-\beta_{n_3})(\beta_{n_2}-\beta_{n_3})} = 0, \tag{3.38}$$

$$v_{m,n_1}\beta_{n_1} + v_{m,n_2}\beta_{n_2} + v_{m,n_3}\beta_{n_3} = \frac{(\beta_{n_2}-\beta_{n_3})\beta_{n_1} + (\beta_{n_3}-\beta_{n_1})\beta_{n_2} + (\beta_{n_1}-\beta_{n_2})\beta_{n_3}}{(\beta_{n_1}-\beta_{n_2})(\beta_{n_1}-\beta_{n_3})(\beta_{n_2}-\beta_{n_3})} = 0. \tag{3.39}$$

Thus, there are $\rho_{\min} - 1 = 2$ vectors along which $\mathbf{V}$ has null projection, corresponding to $j = 0$ and $j = 1$ in (3.36). These two interference free dimensions allow us to retrieve 2 desired symbols, achieving a rate of 2/5 for this example.

As another example, consider a setting with an arbitrary number of messages M and an arbitrary number of servers N, where each message is replicated 4 times, i.e., $\rho_m = \rho_{\min} = 4$ for all $m \in [M]$. Given an arbitrary 4-replicated storage structure, choosing $v_{m,n}$ according to (3.35) allows us to find $\rho_{\min} - 1 = 3$ dimensions along which interference is nulled, corresponding to

$j=0, j=1$, and $j=2$ in (3.36). This is illustrated below.

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_N \\ \beta_1^2 & \beta_2^2 & \cdots & \beta_N^2 \end{bmatrix} \begin{array}{r|ccc|} & & \text{Column } m & \\ & \vdots & \mathbf{0} & \vdots \\ \text{row } n_1 & \cdots & v_{m,n_1} & \cdots \\ & \vdots & \mathbf{0} & \vdots \\ \text{row } n_2 & \cdots & v_{m,n_2} & \cdots \\ & \vdots & \mathbf{0} & \vdots \\ \text{row } n_3 & \cdots & v_{m,n_3} & \cdots \\ & \vdots & \mathbf{0} & \vdots \\ \text{row } n_4 & \cdots & v_{m,n_4} & \cdots \\ & \vdots & \mathbf{0} & \vdots \end{array} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{3.40}$$

Column m corresponds to an arbitrary message set $\mathcal{W}_m$ that is replicated at the 4 servers n_1, n_2, n_3, n_4, and it is easily verified that if $v_{m,n}$ are chosen according to (3.35) then

$$v_{m,n_1} + v_{m,n_2} + v_{m,n_3} + v_{m,n_4} = 0, \tag{3.41}$$

$$\beta_{n_1} v_{m,n_1} + \beta_{n_2} v_{m,n_2} + \beta_{n_3} v_{m,n_3} + \beta_{n_4} v_{m,n_4} = 0, \tag{3.42}$$

$$\beta_{n_1}^2 v_{m,n_1} + \beta_{n_2}^2 v_{m,n_2} + \beta_{n_3}^2 v_{m,n_3} + \beta_{n_4}^2 v_{m,n_4} = 0. \tag{3.43}$$

Thus, there are 3 interference-free dimensions which allow us to retrieve 3 desired symbols for a rate of $3/N$.

In general, if the $\mathbf{V}$ matrix has $\rho_{\min}$ non-zero entries in each column, then by assigning $v_{m,n}$ according to (3.35) there are $\rho_{\min} - 1$ dimensions that are interference free, corresponding to $j \in \{0, 1, \cdots, \rho_{\min} - 2\}$ in (3.36), along which $\rho_{\min} - 1$ desired symbols can be retrieved to achieve the rate $(\rho_{\min} - 1)/N$, which matches $(\rho_{\min} - X - T)/N$ for $X = 0, T = 1$. When $T > 1$ and/or $X > 0$, then additional interference terms enter into the picture due to the additional noise terms needed to protect the messages (X-security) and the queries (T-privacy). Following the construction previously introduced for XSTPIR, these additional interference dimensions are restricted by using cross-subspace alignment [57]. Fortunately, since the storage and query structure used for XSTPIR in [57] is also based on Reed Solomon Codes, it turns out to be compatible with the additional structure imposed by the choice of $v_{m,n}$ in (3.35) according to dual Generalized Reed Solomon Codes. Combining both ideas, it turns out that the number of interference free dimensions that remain available for desired message symbols is equal to $\rho_{\min}-X-T$, which allows us to achieve a rate of $(\rho_{\min}-X-T)/N$. The details are left to the proof of Theorem 3.1.

3.4 Proof of Theorem 3.1

3.4.1 A Simple Example

To make the proof more accessible, let us start with a simple example, which is essentially derived from (3.34). Consider the setting where $N=5$, $T=1$ and $X=0$. There are $M=4$ message sets, $\mathcal{W}_m=(W_{m,1},W_{m,2},\cdots,W_{m,K_m}),m\in[4]$, that are stored in the 5 servers according to the replication pattern $\mathcal{R}=((1,3,4),(2,3,5),(1,2,4),(1,3,5))$, where $K_m,m\in[M]$ are positive integers representing the number of messages in the message set $\mathcal{W}_m$. Note that this replication pattern $\mathcal{R}$ corresponds to the four columns shown in the V matrix in (3.34). The scheme operates over a block where each message is comprised of $L=2$ symbols from $\mathbb{F}_q$ where $q\geq N+L=7$. Let $\beta_1,\beta_2,\cdots,\beta_5,f_1,f_2$ be 7 distinct non-zero elements in $\mathbb{F}_q$. Now let us specify the storage at each server. Server n stores all the $L=2$ symbols of messages from the message sets $\mathcal{W}_m$, for all $m\in\mathcal{M}_n$, i.e.,

$$S_n=\{\mathbf{W}_{m,(1)},\mathbf{W}_{m,(2)},\forall m\in\mathcal{M}_n\} \tag{3.44}$$

$$\mathbf{W}_{m,(\ell)} = [W_{m,1}(\ell), W_{m,2}(\ell), \cdots, W_{m,K_m}(\ell)], \qquad \forall \ell \in [2]. \tag{3.45}$$

For example, consider the first server, we have $\mathcal{M}_1 = \{1,3,4\}$, therefore,

$$S_1 = \{\mathbf{W}_{1,(1)}, \mathbf{W}_{1,(2)}, \mathbf{W}_{3,(1)}, \mathbf{W}_{3,(2)}, \mathbf{W}_{4,(1)}, \mathbf{W}_{4,(2)}\}. \tag{3.46}$$

Notably, for all $m \in [4]$, the $1 \times K_m$ row vector $\mathbf{W}_{m,(\ell)}$ contains the ℓ^{th} symbol from every message in $\mathcal{W}_m$. Suppose the user wishes to retrieve the message $W_{\mu,\kappa} = (W_{\mu,\kappa}(1), W_{\mu,\kappa}(2))$. The query sent to Server n is

$$Q_n^{[\mu,\kappa]} = \{\mathbf{Q}_{m,n,(1)}^{[\mu,\kappa]}, \mathbf{Q}_{m,n,(2)}^{[\mu,\kappa]}, \forall m \in \mathcal{M}_n\} \tag{3.47}$$

where,

$$\begin{aligned} \mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]} &= \frac{v_{m,n}}{f_\ell - \beta_n}\left(\mathbf{F}_m^{[\mu,\kappa]} + (f_\ell - \beta_n)\mathbf{Z}'_{m,1,(\ell)}\right) && (3.48) \\ &= \frac{v_{m,n}}{f_\ell - \beta_n}\mathbf{F}_m^{[\mu,\kappa]} + v_{m,n}\mathbf{Z}'_{m,1,(\ell)}, && (3.49) \end{aligned}$$

the constant values $v_{m,n}$ are defined as

$$v_{m,n} \triangleq \left(\prod_{n' \in \mathcal{R}_m \setminus \{n\}} (\beta_n - \beta_{n'})\right)^{-1}, \tag{3.50}$$

$\mathbf{F}_m^{[\mu,\kappa]}$ are demand vectors defined as

$$\mathbf{F}_m^{[\mu,\kappa]} = \begin{cases} \mathbf{e}_\kappa, \text{ if } m = \mu, \\ \mathbf{0}, \text{ otherwise.} \end{cases} \tag{3.51}$$

where $\mathbf{e}_\kappa$ is the κ^{th} column of the $K_m \times K_m$ identity matrix. The values of $\mathbf{F}_m^{[\mu,\kappa]}$ are kept private from any server, by the $K_m \times 1$ column vectors $\mathbf{Z}'_{m,1,(\ell)}$ comprised of i.i.d uniform noise symbols, for all $m\in[4],\ell\in[2]$. Thus, the scheme is $T=1$-private. To further clarify the construction of queries, consider the first server, we have

$$Q_1^{[\mu,\kappa]}=\{\mathbf{Q}_{1,1,(1)}^{[\mu,\kappa]},\mathbf{Q}_{1,1,(2)}^{[\mu,\kappa]},\mathbf{Q}_{3,1,(1)}^{[\mu,\kappa]},\mathbf{Q}_{3,1,(2)}^{[\mu,\kappa]},\mathbf{Q}_{4,1,(1)}^{[\mu,\kappa]},\mathbf{Q}_{4,1,(2)}^{[\mu,\kappa]}\}. \tag{3.52}$$

The answer returned by Server n is

$$\begin{aligned} A_n^{[\mu,\kappa]} &= \sum_{\ell\in[2]}\sum_{m\in\mathcal{M}_n}\mathbf{W}_{m,(\ell)}\mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]} & (3.53)\\ &= \underbrace{\sum_{\ell\in[2]}\sum_{m\in\mathcal{M}_n}\frac{v_{m,n}}{f_\ell-\beta_n}\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}}_{A_n^{'[\mu,\kappa]}}+\sum_{\ell\in[2]}\sum_{m\in\mathcal{M}_n}v_{m,n}\mathbf{W}_{m,(\ell)}\mathbf{Z}'_{m,1,(\ell)} & (3.54)\end{aligned}$$

Upon receiving all $N=5$ answers, the user evaluates the $L=2$ values Y_1,Y_2, as follows.

$$\begin{bmatrix} Y_1\\ Y_2\end{bmatrix}=\begin{bmatrix} 1 & 1 & \cdots & 1\\ \beta_1 & \beta_2 & \cdots & \beta_5\end{bmatrix}\begin{bmatrix} A_1^{[\mu,\kappa]}\\ A_2^{[\mu,\kappa]}\\ \vdots\\ A_5^{[\mu,\kappa]}\end{bmatrix}. \tag{3.55}$$

Note that by the construction of $A_1^{[\mu,\kappa]},A_2^{[\mu,\kappa]},\cdots,A_5^{[\mu,\kappa]}$, the column vector on the RHS of (3.55) can be written in the following matrix form.

$$\begin{bmatrix} A_1^{[\mu,\kappa]}\\ A_2^{[\mu,\kappa]}\\ \vdots\\ A_5^{[\mu,\kappa]}\end{bmatrix}=\begin{bmatrix} A_1^{'[\mu,\kappa]}\\ A_2^{'[\mu,\kappa]}\\ \vdots\\ A_5^{'[\mu,\kappa]}\end{bmatrix}+\begin{bmatrix} v_{1,1} & 0 & v_{3,1} & v_{4,1}\\ 0 & v_{2,2} & v_{3,2} & 0\\ v_{1,3} & v_{2,3} & 0 & v_{4,3}\\ v_{1,4} & 0 & v_{3,4} & 0\\ 0 & v_{2,5} & 0 & v_{4,5}\end{bmatrix}\begin{bmatrix} \mathbf{W}_{1,(1)}\mathbf{Z}'_{1,1,(1)}+\mathbf{W}_{1,(2)}\mathbf{Z}'_{1,1,(2)}\\ \mathbf{W}_{2,(1)}\mathbf{Z}'_{2,1,(1)}+\mathbf{W}_{2,(2)}\mathbf{Z}'_{2,1,(2)}\\ \mathbf{W}_{3,(1)}\mathbf{Z}'_{3,1,(1)}+\mathbf{W}_{3,(2)}\mathbf{Z}'_{3,1,(2)}\\ \mathbf{W}_{4,(1)}\mathbf{Z}'_{4,1,(1)}+\mathbf{W}_{4,(2)}\mathbf{Z}'_{4,1,(2)}\end{bmatrix}. \tag{3.56}$$

As previously shown in (3.37), guaranteed by Lemma B.2, the choice of constant values $v_{m,n}$ satisfy the property that

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_5 \end{bmatrix} \begin{bmatrix} v_{1,1} & 0 & v_{3,1} & v_{4,1} \\ 0 & v_{2,2} & v_{3,2} & 0 \\ v_{1,3} & v_{2,3} & 0 & v_{4,3} \\ v_{1,4} & 0 & v_{3,4} & 0 \\ 0 & v_{2,5} & 0 & v_{4,5} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{3.57}$$

Besides, we note that by the definition of demand vectors $\mathbf{F}_m^{[\mu,\kappa]}$, we have

$$\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\mu,\kappa]} = \begin{cases} W_{\mu,\kappa}(\ell), \text{ if } m=\mu, \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{3.58}$$

Therefore, we have

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \beta_{\mathcal{R}_\mu(1)} & \beta_{\mathcal{R}_\mu(2)} & \beta_{\mathcal{R}_\mu(3)} \end{bmatrix} \begin{bmatrix} \frac{v_{\mu,\mathcal{R}_\mu(1)}}{f_1-\beta_{\mathcal{R}_\mu(1)}} & \frac{v_{\mu,\mathcal{R}_\mu(1)}}{f_2-\beta_{\mathcal{R}_\mu(1)}} \\ \frac{v_{\mu,\mathcal{R}_\mu(2)}}{f_1-\beta_{\mathcal{R}_\mu(2)}} & \frac{v_{\mu,\mathcal{R}_\mu(2)}}{f_2-\beta_{\mathcal{R}_\mu(2)}} \\ \frac{v_{\mu,\mathcal{R}_\mu(3)}}{f_1-\beta_{\mathcal{R}_\mu(3)}} & \frac{v_{\mu,\mathcal{R}_\mu(3)}}{f_2-\beta_{\mathcal{R}_\mu(3)}} \end{bmatrix} \begin{bmatrix} W_{\mu,\kappa}(1) \\ W_{\mu,\kappa}(2) \end{bmatrix}. \tag{3.59}$$

Guaranteed by Lemma B.3, the product of the first two matrices on the RHS is an invertible 2×2 matrix. Thus the user is able to obtain the desired message $W_{\mu,\kappa}=(W_{\mu,\kappa}(1),W_{\mu,\kappa}(2))$ by inverting the matrix. Note that the 2 symbols of the desired message are obtained from a total of 5 downloaded symbols, the rate achieved by the scheme is $R=2/5$, which achieves the desired rate.

3.4.2 A General Scheme

Now let us we present the achievable scheme for GXSTPIR for arbitrary N,T,X,M,K_m,ρ_m values that allows private retrieval of any desired message at a rate $R=\frac{\rho_{\min}-X-T}{N}$. Without loss of generality we will assume that $\rho_m=\rho_{\min}$ for all $m\in[M]$. For any message that is replicated more than $\rho_{\min}$ times, the scheme can be applied by arbitrarily choosing any $\rho_{\min}$ replications of that message and ignoring the rest. In order to achieve the rate $R=\frac{\rho_{\min}-X-T}{N}$, the scheme will retrieve $\rho_{\min}-X-T$ desired symbols by downloading one symbol from each server.

The scheme operates over a block where each message is comprised of L symbols and we have

$$L=\rho_{\min}-X-T. \tag{3.60}$$

All symbols are in $\mathbb{F}_q$ and without loss of generality we will assume that $q\geq N+L$. Let $\beta_{[N]},f_{[N]}$ be distinct non-zero values in $\mathbb{F}_q$. Server n stores,

$$S_n=\{\mathbf{W}_{m,(1)}^{(n)},\mathbf{W}_{m,(2)}^{(n)},\cdots,\mathbf{W}_{m,(L)}^{(n)},\forall m\in\mathcal{M}_n\} \tag{3.61}$$

$$\mathbf{W}_{m,(\ell)}^{(n)}=\mathbf{W}_{m,(\ell)}+\sum_{x\in[X]}(f_\ell-\beta_n)^x\mathbf{Z}_{m,x,(\ell)} \tag{3.62}$$

$$\mathbf{W}_{m,(\ell)}=[W_{m,1}(\ell),W_{m,2}(\ell),\cdots,W_{m,K_m}(\ell)], \qquad \forall \ell\in[L]. \tag{3.63}$$

Thus, for all $m\in[M]$, the $1\times K_m$ row vector $\mathbf{W}_{m,(\ell)}$ contains the ℓ^{th} symbol from every message in $\mathcal{W}_m$. For all $m\in[M],x\in[X],\ell\in[L]$, the $1\times K_m$ row vectors $\mathbf{Z}_{m,x,(\ell)}$ are comprised of i.i.d. uniform noise symbols. Any message symbol $W_{m,k}(\ell)$ that is secret-shared among servers $\mathcal{R}_m$, is protected by the X noise symbols $\mathbf{Z}_{m,1,(\ell)}(k),\mathbf{Z}_{m,2,(\ell)}(k),\cdots,\mathbf{Z}_{m,X,(\ell)}(k)$ that are i.i.d. uniform and coded according to an MDS(X,$\rho_{\min}$) code, so that the shares accessible to any set of up to X colluding servers are independent of $W_{m,k}(\ell)$. Thus the scheme is

X-secure.

The query sent to Server n is

$$Q_n^{[\mu,\kappa]} = \{\mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]}, \forall m \in \mathcal{M}_n, \ell \in [L]\} \tag{3.64}$$

where,

$$\mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]} = \frac{v_{m,n}}{f_\ell - \beta_n}\left(\mathbf{F}_m^{[\mu,\kappa]} + \sum_{t\in[T]} (f_\ell - \beta_n)^t \mathbf{Z}'_{m,t,(\ell)}\right) \tag{3.65}$$

$\mathbf{F}_m^{[\mu,\kappa]}$ are demand vectors defined as

$$\mathbf{F}_m^{[\mu,\kappa]} = \begin{cases} \mathbf{e}_\kappa, \text{ if } m = \mu, \\ \mathbf{0}, \text{ otherwise.} \end{cases} \tag{3.66}$$

where $\mathbf{e}_\kappa$ is the κ^{th} column of the $K_m \times K_m$ identity matrix. The values of $\mathbf{F}_m^{[\mu,\kappa]}$ are kept private from any set of up to T colluding servers, by the $K_m \times 1$ column vectors $\mathbf{Z}'_{m,t,(\ell)}$ comprised of i.i.d uniform noise symbols, for all $m \in [M], t \in [T], \ell \in [L]$. Note that the noise vectors that protect $\mathbf{F}_m^{[\mu,\kappa]}$ are coded according to an MDS$(T, \rho_{\min})$ code spread across the queries sent to servers in $\mathcal{R}_m$, i.e., all queries that contain $\mathbf{F}_m^{[\mu,\kappa]}$, so that the queries accessible to any set of up to T servers reveal no information about the demand vectors. Thus, the scheme is T-private.

The constant values $v_{m,n}$ in (3.65) are defined as

$$v_{m,n} \triangleq \left(\prod_{n' \in \mathcal{R}_m \setminus \{n\}} (\beta_n - \beta_{n'})\right)^{-1} \tag{3.67}$$

As shown in Lemma B.2 in Appendix B.1 using the properties of dual GRS codes, this choice of $v_{m,n}$ satisfies the crucial property that

$$\sum_{n\in\mathcal{R}_m} v_{m,n}\beta_n^j = 0 \tag{3.68}$$

for all $m\in[M]$ and for all $j\in\{0,1,\cdots,\rho_{\min}-2\}$.

The answer returned by Server n is

$$A_n^{[\mu,\kappa]} = \sum_{\ell\in[L]}\sum_{m\in\mathcal{M}_n} \mathbf{W}_{m,(\ell)}^{(n)}\mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]} \tag{3.69}$$

Upon receiving all N answers, the user evaluates the L values $Y_1,Y_2,\cdots,Y_L$, as follows.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_L \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_N \\ \vdots & \vdots & \cdots & \vdots \\ \beta_1^{L-1} & \beta_2^{L-1} & \cdots & \beta_N^{L-1} \end{bmatrix} \begin{bmatrix} A_1^{[\mu,\kappa]} \\ A_2^{[\mu,\kappa]} \\ \vdots \\ A_N^{[\mu,\kappa]} \end{bmatrix} \tag{3.70}$$

so that for all $i\in[L]$,

$$\begin{aligned}
Y_i &= \sum_{n\in[N]} \beta_n^{i-1} A_n^{[\mu,\kappa]} && (3.71)\\
&= \sum_{n\in[N]} \beta_n^{i-1} \sum_{l\in[L]}\sum_{m\in\mathcal{M}_n} \mathbf{W}_{m,(\ell)}^{(n)}\mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]} && (3.72)\\
&= \sum_{\ell\in[L]}\sum_{m\in[M]}\sum_{n\in\mathcal{R}_m} \beta_n^{i-1}\mathbf{W}_{m,(\ell)}^{(n)}\mathbf{Q}_{m,n,(\ell)}^{[\mu,\kappa]} && (3.73)\\
&= \sum_{\ell\in[L]}\sum_{m\in[M]}\sum_{n\in\mathcal{R}_m} \frac{v_{m,n}\beta_n^{i-1}}{f_\ell-\beta_n}\left(\mathbf{W}_{m,(\ell)} + \sum_{x\in[X]}(f_\ell-\beta_n)^x\mathbf{Z}_{m,x,(\ell)}\right) \\
&\qquad \left(\mathbf{F}_m^{[\mu,\kappa]} + \sum_{t\in[T]}(f_\ell-\beta_n)^t\mathbf{Z}'_{m,t,(\ell)}\right) && (3.74)
\end{aligned}$$

$$
\begin{aligned}
=& \sum_{\ell\in[L]}\sum_{m\in[M]}\sum_{n\in\mathcal{R}_m}\Bigg(\frac{v_{m,n}\beta_n^{i-1}}{f_\ell-\beta_n}\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}+\sum_{t\in[T]}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{t-1}\mathbf{W}_{m,(\ell)}\mathbf{Z}'_{m,t,(\ell)}\\
&\qquad\qquad +\sum_{x\in[X]}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x-1}\mathbf{Z}_{m,x,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}\\
&\qquad\qquad +\sum_{x\in[X]}\sum_{t\in[T]}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x+t-1}\mathbf{Z}_{m,x,(\ell)}\mathbf{Z}'_{m,t,(\ell)}\Bigg) \qquad (3.75)
\end{aligned}
$$

$$
\begin{aligned}
=& \sum_{\ell\in[L]}\sum_{m\in[M]}\sum_{n\in\mathcal{R}_m}\left(\frac{v_{m,n}\beta_n^{i-1}}{f_\ell-\beta_n}\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}\right)\\
&+\sum_{\ell\in[L]}\sum_{m\in[M]}\left(\sum_{t\in[T]}\mathbf{W}_{m,(\ell)}\mathbf{Z}'_{m,t,(\ell)}\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{t-1}\right)\right)\\
&+\sum_{\ell\in[L]}\sum_{m\in[M]}\left(\sum_{x\in[X]}\mathbf{Z}_{m,x,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x-1}\right)\right)\\
&+\sum_{\ell\in[L]}\sum_{m\in[M]}\left(\sum_{x\in[X]}\sum_{t\in[T]}\mathbf{Z}_{m,x,(\ell)}\mathbf{Z}'_{m,t,(\ell)}\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x+t-1}\right)\right) \qquad (3.76)
\end{aligned}
$$

The terms $\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{t-1}\right)$, $\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x-1}\right)$ and $\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x+t-1}\right)$ are equal to zero because of (3.68). This is because all of these can be expanded into weighted sums of terms of the form $\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^j$ for j taking values in $\{0,1,\cdots,\rho_{\min}-2\}$. Let us show this explicitly for $\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{t-1}$ as follows,

$$
\begin{aligned}
\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{t-1} &= \sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i-1}\left(\sum_{\tau\in\{0,1,\cdots,t-1\}}\binom{t-1}{\tau}\beta_n^{\tau}\ell^{t-1-\tau}\right) && (3.77)\\
&= \sum_{\tau\in\{0,1,\cdots,t-1\}}\binom{t-1}{\tau}f_\ell^{t-1-\tau}\left(\sum_{n\in\mathcal{R}_m}v_{m,n}\beta_n^{i+\tau-1}\right) && (3.78)\\
&= 0 && (3.79)
\end{aligned}
$$

because $0\le i+\tau-1\le L+(T-1)-1=\rho_{\min}-X-2\le\rho_{\min}-2$. It can be similarly shown that $\left(\sum_{n\in\mathcal{R}_m} v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x-1}\right)=0$ and $\left(\sum_{n\in\mathcal{R}_m} v_{m,n}\beta_n^{i-1}(f_\ell-\beta_n)^{x+t-1}\right)=0$. Thus, we have,

$$
\begin{aligned}
Y_i &= \sum_{\ell\in[L]}\sum_{m\in[M]}\sum_{n\in\mathcal{R}_m}\left(\frac{v_{m,n}\beta_n^{i-1}}{f_\ell-\beta_n}\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}\right) && (3.80)\\
&= \sum_{\ell\in[L]}\sum_{m\in[M]}\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\mu,\kappa]}\left(\sum_{n\in\mathcal{R}_m}\frac{v_{m,n}\beta_n^{i-1}}{f_\ell-\beta_n}\right) && (3.81)\\
&= \sum_{\ell\in[L]}\mathbf{W}_{\mu,(\ell)}\mathbf{e}_\kappa\left(\sum_{n\in\mathcal{R}_\mu}\frac{v_{\mu,n}\beta_n^{i-1}}{f_\ell-\beta_n}\right) && (3.82)\\
&= \sum_{\ell\in[L]}\sum_{n\in\mathcal{R}_\mu}W_{\mu,\kappa}(\ell)\frac{v_{\mu,n}\beta_n^{i-1}}{f_\ell-\beta_n} && (3.83)
\end{aligned}
$$

Note that we used (3.66) to obtain (3.82). In matrix notation, we have,

$$
\begin{bmatrix} Y_1\\ Y_2\\ \vdots\\ Y_L\end{bmatrix}
=
\underbrace{\begin{bmatrix} 1 & \cdots & 1\\ \beta_{\mathcal{R}_\mu(1)} & \cdots & \beta_{\mathcal{R}_\mu(\rho_m)}\\ \vdots & \vdots & \vdots\\ \beta_{\mathcal{R}_\mu(1)}^{L-1} & \cdots & \beta_{\mathcal{R}_\mu(\rho_m)}^{L-1}\end{bmatrix}}_{\mathbf{A}}
\underbrace{\begin{bmatrix} \frac{v_{\mu,\mathcal{R}_\mu(1)}}{f_1-\beta_{\mathcal{R}_\mu(1)}} & \cdots & \frac{v_{\mu,\mathcal{R}_\mu(1)}}{f_L-\beta_{\mathcal{R}_\mu(1)}}\\ \vdots & \vdots & \vdots\\ \frac{v_{\mu,\mathcal{R}_\mu(\rho_m)}}{f_1-\beta_{\mathcal{R}_\mu(\rho_m)}} & \cdots & \frac{v_{\mu,\mathcal{R}_\mu(\rho_m)}}{f_L-\beta_{\mathcal{R}_\mu(\rho_m)}}\end{bmatrix}}_{\mathbf{B}}
\begin{bmatrix} W_{\mu,\kappa}(1)\\ W_{\mu,\kappa}(2)\\ \vdots\\ W_{\mu,\kappa}(L)\end{bmatrix}. \qquad (3.84)
$$

Guaranteed by Lemma B.3 and the the definitions of $v_{m,n}$ and β_n, $\forall m\in[M], n\in[N]$, the $L\times L$ matrix $\mathbf{AB}$ is invertible, and the desired message is retrievable by inverting the matrix. Thus the scheme is correct. This completes the proof of Theorem 3.1. □

3.4.3 A Private Computation Scheme for $X=0$, $\rho_{\min}=T+1$.

From the description of the scheme, it is evident that the demand vectors are protected by the uniform noise, regardless of how they are chosen. Modifying the choice of demand vectors would allow the user to privately retrieve various forms of desired information, generalizing the scheme to broader applications. Here we present a simple example that will also be useful for the proof of Theorem 3.3.

Suppose there are no security constraints ($X=0$) and every message is replicated $T+1$ times ($\rho_{\min}=T+1$), so that our scheme operates over blocks comprised of $L=\rho_{\min}-X-T=1$ symbol per message. Recall that our scheme allows the user to retrieve an arbitrary message $W_{\mu,\kappa}$ at the rate $R=(\rho_{\min}-X-T)/N=1/N$ in this setting. Now, suppose instead of an arbitrary message, the user wants to retrieve an arbitrary linear combination of all messages,

$$\lambda(\mathcal{W})\triangleq\sum_{m\in[M]}\sum_{k\in K_m}\lambda_{m,k}W_{m,k}(1)=\sum_{m\in[M]}\mathbf{W}_{m,(1)}\boldsymbol{\lambda}_m, \qquad \forall\ell\in[L] \tag{3.85}$$

where

$$\boldsymbol{\lambda}_m=[\lambda_{m,1},\lambda_{m,2},\cdots,\lambda_{m,K_m}]^T\in\mathbb{F}_q^{K_m\times 1}, \qquad \forall m\in[M], \tag{3.86}$$

are the combining coefficients to be kept private from any set of up to T colluding servers. This is a form of the private linear computation problem studied in [107] applied here to graph based replicated storage. To apply our scheme to this setting, replace the demand vectors $\mathbf{F}_m^{[\mu,\kappa]}$ with $\mathbf{F}_m^{[\lambda]}$ defined as follows.

$$\mathbf{F}_m^{[\lambda]}=\left(\sum_{n\in\mathcal{R}_m}\frac{v_{m,n}}{f_1-\beta_n}\right)^{-1}\boldsymbol{\lambda}_m \tag{3.87}$$

so that continuing from (3.81) we have

$$Y_i=\sum_{\ell\in[L]}\sum_{m\in[M]}\mathbf{W}_{m,(\ell)}\mathbf{F}_m^{[\lambda]}\left(\sum_{n\in\mathcal{R}_m}\frac{v_{m,n}\beta_n^{i-1}}{f_\ell-\beta_n}\right),\quad i\in[L]=\{1\} \tag{3.88}$$

$$\tag{3.89}$$

$$\Rightarrow Y_1=\sum_{m\in[M]}\mathbf{W}_{m,(1)}\boldsymbol{\lambda}_m\left(\sum_{n\in\mathcal{R}_m}\frac{v_{m,n}}{f_1-\beta_n}\right)^{-1}\left(\sum_{n\in\mathcal{R}_m}\frac{v_{m,n}}{f_1-\beta_n}\right) \tag{3.90}$$

$$= \sum_{m\in[M]} \mathbf{W}_{m,(1)}\boldsymbol{\lambda}_m = \lambda(\mathcal{W}) \tag{3.91}$$

Thus, a private computation scheme is readily obtained for the case where all messages are replicated at least $T+1$ times. The rate of this scheme is $(\rho_{\min}-T)/N = 1/N$. Just as in [107], there is no rate loss relative to the case where the user wants to retrieve only one message $W_{\mu,\kappa}$.

3.5 Proof of Theorem 3.2

Let $\mathcal{T}$ be a subset of $\mathcal{R}_m$, such that $|\mathcal{T}| = \max(|\mathcal{R}_m|,T)$. Let $\mathcal{X}$ be a subset of $\mathcal{R}_m \setminus \mathcal{T}$, such that $|\mathcal{X}| = \max(|\mathcal{R}_m|-|\mathcal{T}|,X)$. Note that it follows from the definition that $\mathcal{T}\cap\mathcal{X}=\emptyset$. From the decodability of message $W_{m,k}$ we have,

$$\begin{aligned}
L &= I\left(W_{m,k} \ ; \ A_{[N]}^{[m,k]} \mid Q_{[N]}^{[m,k]}\right) && (3.92)\\
&\le I\left(W_{m,k} \ ; \ A_{\mathcal{R}_m\setminus\mathcal{X}}^{[m,k]}, S_{[N]\setminus\mathcal{R}_m}, S_{\mathcal{X}} \mid Q_{[N]}^{[m,k]}\right) && (3.93)\\
&= I\left(W_{m,k} \ ; \ S_{[N]\setminus\mathcal{R}_m}, S_{\mathcal{X}} \mid Q_{[N]}^{[m,k]}\right) + I\left(W_{m,k} \ ; \ A_{\mathcal{R}_m\setminus\mathcal{X}}^{[m,k]} \mid S_{[N]\setminus\mathcal{R}_m}, S_{\mathcal{X}}, Q_{[N]}^{[m,k]}\right) && (3.94)\\
&= I\left(W_{m,k} \ ; \ A_{\mathcal{R}_m\setminus\mathcal{X}}^{[m,k]} \mid S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{[N]}^{[m,k]}\right) && (3.95)\\
&= I\left(W_{m,k} \ ; \ A_{\mathcal{T}}^{[m,k]}, A_{(\mathcal{R}_m\setminus\mathcal{X})\setminus\mathcal{T}}^{[m,k]} \mid S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{[N]}^{[m,k]}\right) && (3.96)\\
&= I\left(W_{m,k}; A_{\mathcal{T}}^{[m,k]} \mid S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{[N]}^{[m,k]}\right)\\
&\qquad + I\left(W_{m,k}; A_{(\mathcal{R}_m\setminus\mathcal{X})\setminus\mathcal{T}}^{[m,k]} \mid A_{\mathcal{T}}^{[m,k]}, S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{[N]}^{[m,k]}\right) && (3.97)\\
&\le I(W_{m,k}; A_{\mathcal{T}}^{[m,k]} \mid S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{[N]}^{[m,k]}) + \sum_{n\in(\mathcal{R}_m\setminus\mathcal{X})\setminus\mathcal{T}} H(A_n^{[m,k]}) && (3.98)\\
&\le I(W_{m,k}; A_{\mathcal{T}}^{[m,k]} \mid S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{\mathcal{T}}^{[m,k]}) + \sum_{n\in(\mathcal{R}_m\setminus\mathcal{X})\setminus\mathcal{T}} H(A_n^{[m,k]}) && (3.99)\\
&\le I(W_{m,k}; A_{\mathcal{T}}^{[m,k']} \mid S_{\mathcal{X}}, S_{[N]\setminus\mathcal{R}_m}, Q_{\mathcal{T}}^{[m,k']}) + \sum_{n\in(\mathcal{R}_m\setminus\mathcal{X})\setminus\mathcal{T}} H(A_n^{[m,k']}) && (3.100)
\end{aligned}$$

In (3.93) we used the fact that $A^{[m,k]}_{[N]\backslash(\mathcal{R}_m\backslash\mathcal{X})}$ is a function of $\left(S_{[N]\backslash\mathcal{R}_m}, S_{\mathcal{X}}, Q^{[m,k]}_{[N]}\right)$, and $I(A;f(B,C)\mid C) \leq I(A;f(B,C),B\mid C) = I(A;B\mid C) + I(A;f(B,C)\mid B,C) = I(A;B\mid C)$ where $f(B,C)$ is some function of B,C. The chain rule of mutual information is used for (3.94). For (3.95) we used the fact that $(S_{[N]\backslash\mathcal{R}_m}, S_{\mathcal{X}})$ is independent of $\left(W_{m,k}, Q^{[m,k]}_{[N]}\right)$ according to Lemma B.4. The next step, (3.96) simply re-writes the same expression in different notation, while (3.97) follows from chain rule of mutual information. For (3.98) we used the fact that $I(A;B\mid C) = H(B\mid C) - H(B\mid A,C) \leq H(B)$ because entropy is non-negative and conditioning reduces entropy. (3.99) follows from Lemma B.5. (3.100) follows because $I(Q^{[m,\kappa]}_{\mathcal{T}}, A^{[m,\kappa]}_{\mathcal{T}}, S_{[N]}; \kappa) = 0$ according to Lemma B.6. Equivalently,

$$\left(Q^{[m,k]}_{\mathcal{T}}, A^{[m,k]}_{\mathcal{T}}, S_{[N]}\right) \sim \left(Q^{[m,k']}_{\mathcal{T}}, A^{[m,k']}_{\mathcal{T}}, S_{[N]}\right) \tag{3.101}$$

for all $m \in [M]$ and $k,k' \in [K_m]$, which in turn implies (3.100).

Summing (3.100) over all $k \in [K_m]$ we have

$$K_m L \leq \left(\sum_{k\in[K_m]} I(W_{m,k}; A^{[m,k']}_{\mathcal{T}} \mid S_{\mathcal{X}}, S_{[N]\backslash\mathcal{R}_m}, Q^{[m,k']}_{\mathcal{T}})\right) + K_m \sum_{n\in(\mathcal{R}_m\backslash\mathcal{X})\backslash\mathcal{T}} H(A^{[m,k']}_n) \tag{3.102}$$

$$\leq I(W_{m,1}, \cdots, W_{m,K_m}; A^{[m,k']}_{\mathcal{T}} \mid S_{\mathcal{X}}, S_{[N]\backslash\mathcal{R}_m}, Q^{[m,k']}_{\mathcal{T}}) + K_m \sum_{n\in(\mathcal{R}_m\backslash\mathcal{X})\backslash\mathcal{T}} H(A^{[m,k']}_n) \tag{3.103}$$

$$\leq H(A^{[m,k']}_{\mathcal{T}}) + K_m \sum_{n\in(\mathcal{R}_m\backslash\mathcal{X})\backslash\mathcal{T}} H(A^{[m,k']}_n) \tag{3.104}$$

(3.103) follows from the chain rule of mutual information and repeated use of the property that $I(A;C\mid D) + I(B;C\mid D) \leq I(A;C\mid D) + I(B;C\mid A,D) = I(A,B;C\mid D)$ when A,B are independent conditioned on D, i.e., $I(A;B\mid D) = 0$. This conditional independence property for (3.103) is proved in Lemma B.7. (3.104) follows from the facts that entropy is non-negative and conditioning reduces entropy, i.e., $I(A;B\mid C) = H(A\mid C) - H(A\mid B,C) \leq H(A\mid$

$C) \leq H(A)$.

From (3.104) we note that if $|\mathcal{R}_m| \leq X+T$ then $\mathcal{R}_m \setminus \mathcal{X} \setminus \mathcal{T} = \emptyset$, which means that as $K_m \rightarrow \infty$, we must have $H(A_{\mathcal{T}}^{[m,k']}) \rightarrow \infty$, and since the download approaches infinity, the asymptotic capacity is zero. This is the degenerate case in Theorem 3.2.

Having dealt with the degenerate setting, henceforth, let us assume that $|\mathcal{R}_m| > X+T$ for all $m \in [M]$. Since the capacity for this case is not zero (follows from achievability), there is no loss of generality in assuming that the asymptotic value of download cost is bounded, i.e., $H(A_n^{[m,k']})/K_m = o(1)$ as a function of K_m for all $n \in [N]$. Recall that $f(x) = o(1)$ is equivalent to the condition that $\lim_{x \rightarrow \infty} f(x) = 0$. In this case we have

$$\sum_{n \in (\mathcal{R}_m \setminus \mathcal{X}) \setminus \mathcal{T}} \frac{H(A_n^{[m,k']})}{L} + o(1) \geq 1 \tag{3.105}$$

$$\Rightarrow \sum_{n \in (\mathcal{R}_m \setminus \mathcal{X}) \setminus \mathcal{T}} D_n + o(1) \geq 1. \tag{3.106}$$

where $D_n = \frac{H(A_n^{[m,k']})}{L}$ is defined as the value of download from server n, normalized by L. As $K \rightarrow \infty$ all $o(1)$ terms approach 0 and we obtain the set of conditions that define $\mathcal{D}$ in (3.27). The capacity bound in Theorem 3.2 for the non-degenerate setting follows from the definition of capacity as the supremum of $L/D = (D_1 + \cdots + D_N)^{-1}$. □

3.6 Proof of Theorem 3.3

3.6.1 Proof of Converse for Theorem 3.3

It already follows from Theorem 3.2 that if $\rho_{\min} \leq T$ then the capacity is zero. So let us assume that $\rho_{\min} > T$. Theorem 3.3 also limits $\rho_m \leq T+2$ for all $m \in [M]$, therefore we must have $\rho_m \in \{T+1, T+2\}$ for all $m \in [M]$, i.e., every message is either $(T+1)$-replicated or

$(T+2)$-replicated. Recall that $\mathcal{N}_{T+2}$ is the set of servers that do not store any messages that are $(T+1)$-replicated. The remaining servers are in $\mathcal{N}_{T+1}$.

According to the general converse bound in Theorem 3.2, the asymptotic capacity C_∞ is bounded above by the maximum value of $(D_1+\cdots+D_N)^{-1}$ subject to the constraints,

$$D_u+D_v \geq 1, \qquad \forall uv \in E[\mathcal{N}_{T+2}] \tag{3.107}$$

$$D_t \geq 1, \qquad \forall t \in [\mathcal{N}_{T+1}] \tag{3.108}$$

We use the notation $G[\mathcal{N}_{T+2}]$ to represent the induced subgraph of $G[V,E]$ whose vertex set is $\mathcal{N}_{T+2}$ and whose edge set, denoted $E[\mathcal{N}_{T+2}]$ consists of all edges $uv \in E$ such that $u,v \in \mathcal{N}_{T+2}$. Recall that a 2-matching in $G[\mathcal{N}_{T+2}]$ is a vector x that assigns to each edge $uv \in E[\mathcal{N}_{T+2}]$, a value from $\{0,1,2\}$ such that the sum of values assigned to all edges in $E[\mathcal{N}_{T+2}]$ that are incident with any vertex $n \in \mathcal{N}_{T+2}$ is not more than 2. Let x be the vector that produces the maximum size 2-matching in $G[\mathcal{N}_{T+2}]$, i.e., the size of x is

$$\sum_{uv \in E[\mathcal{N}_{T+2}]} x(uv) = \nu_2(G[\mathcal{N}_{T+2}]). \tag{3.109}$$

Multiplying both sides of (3.107) by $x(uv)$, summing up over all $uv \in E[\mathcal{N}_{T+2}]$, and adding $2 \times (3.108)$, we have

$$\sum_{uv \in E[\mathcal{N}_{T+2}]} (D_u+D_v)x(uv) + 2\sum_{t \in [\mathcal{N}_{T+1}]} (D_t) \geq \sum_{uv \in E[\mathcal{N}_{T+2}]} x(uv) + 2|\mathcal{N}_{T+1}| \tag{3.110}$$

$$\Rightarrow \sum_{u \in \mathcal{N}_{T+2}} x(\delta(u) \cap E[\mathcal{N}_{T+2}])(D_u) + 2\sum_{t \in [\mathcal{N}_{T+1}]} (D_t) \geq \nu_2(G[\mathcal{N}_{T+2}]) + 2|\mathcal{N}_{T+1}| \tag{3.111}$$

$$\Rightarrow 2\sum_{u \in \mathcal{N}_{T+2}} (D_u) + 2\sum_{t \in [\mathcal{N}_{T+1}]} (D_t) \geq \nu_2(G[\mathcal{N}_{T+2}]) + 2|\mathcal{N}_{T+1}| \tag{3.112}$$

$$\Rightarrow 2\sum_{u \in [N]} (D_u) \geq \nu_2(G[\mathcal{N}_{T+2}]) + 2|\mathcal{N}_{T+1}| \tag{3.113}$$

$$\Rightarrow (D_1+D_2+\cdots+D_n) \geq \frac{\nu_2(G[\mathcal{N}_{T+2}]) + 2|\mathcal{N}_{T+1}|}{2} \tag{3.114}$$

In (3.112) we used the fact that the sum of values assigned by x to all edges in $E[\mathcal{N}_{T+2}]$ that are incident with the vertex u is not more than 2. Combining (3.114) with the result of Theorem 3.2, we obtain the desired converse bound

$$C_\infty \leq \frac{2}{\nu_2(G[\mathcal{N}_{T+2}]) + 2|\mathcal{N}_{T+1}|}. \tag{3.115}$$

Thus, the proof of converse for Theorem 3.3 is complete. □

3.6.2 Proof of Achievability for Theorem 3.3

Let us define $\widetilde{\mathcal{W}}_{T+1}$ as the set of messages that are replicated $T+1$ times. Let $U \subset \mathcal{N}_{T+2}$ be a stable set. We will show that it is possible to retrieve $L=2$ desired symbols with a total normalized download,

$$D_1 + \cdots + D_N = \frac{|[1:N] \backslash U| + |\mathcal{N}(U) \cup \mathcal{N}_{T+1}|}{2} \tag{3.116}$$

The achievable scheme does not use the servers in U. Let $\widetilde{\mathcal{W}}_U$ denote the set of messages that are stored at any of the servers in U. Note that none of these messages is in $\widetilde{\mathcal{W}}_{T+1}$ because $U \subset \mathcal{N}_{T+2}$. Also note that no message is replicated more than once in U because U is a stable set. After the servers in U are eliminated, the messages $\widetilde{\mathcal{W}}^* = \widetilde{\mathcal{W}}_U \cup \widetilde{\mathcal{W}}_{T+1}$ are now replicated exactly $(T+1)$ times in the remaining servers. All other messages are replicated $(T+2)$ times. As a thought experiment, suppose we add a genie server that stores $\widetilde{\mathcal{W}}^*$. Now we have a storage system where all messages are replicated $(T+2)$ times, so that the scheme presented in the proof of Theorem 3.1 can be used to retrieve $L=2$ desired symbols while downloading $|[N] \backslash U| + 1$ symbols, which includes one genie symbol, say $\lambda(\widetilde{\mathcal{W}}^*)$. In order to obtain $\lambda(\widetilde{\mathcal{W}}^*)$ without a genie, we will use the servers in the set $\mathcal{N}(U) \cup \mathcal{N}_{T+1}$. Note that $\mathcal{N}(U)$ and $\mathcal{N}_{T+1}$ may have some servers in common. More importantly, note that $\widetilde{\mathcal{W}}^*$ is replicated $(T+1)$ times within this set. Therefore, we can privately retrieve $\lambda(\widetilde{\mathcal{W}}^*)$ by downloading one symbol from each of these servers, with the scheme described in Section 3.4.3. Thus, we have a private and correct scheme that retrieves $L=2$ desired symbols with a total download of $|[N]\backslash U|+|\mathcal{N}(U)\cup\mathcal{N}_{T+1}|$. Next, we note the following identity,

$$\underbrace{|[N]\backslash U|}_{t_1}+\underbrace{|\mathcal{N}(U)\cup\mathcal{N}_{T+1}|}_{t_2}=\underbrace{|\mathcal{N}_{T+2}\backslash U|}_{t_3}+\underbrace{|\mathcal{N}(U)\cap\mathcal{N}_{T+2}|}_{t_4}+2\underbrace{|\mathcal{N}_{T+1}|}_{t_5} \tag{3.117}$$

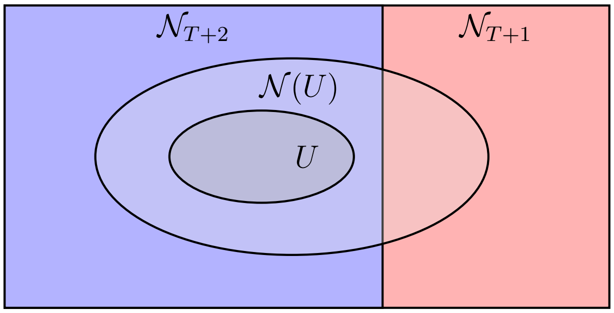


Figure 3.2: General setting of $U\subset\mathcal{N}_{T+2}$ which may have neighbors $\mathcal{N}(U)$ both in $\mathcal{N}_{T+1}$ and $\mathcal{N}_{T+2}$. Note that $\mathcal{N}(U)$ does not include U.

Let us verify that the identity holds as follows. First consider the servers in $\mathcal{N}_{T+1}$. On the LHS all these servers are included in t_1 as well as t_2, i.e., they are counted twice. On the RHS these servers are included only in t_5 which is scaled by a factor of 2, so both sides match. Now consider servers that are in $\mathcal{N}_{T+2}$ and are neighbors of servers in U. On the LHS these servers are included in t_1 as well as t_2, i.e., they are counted twice. On the RHS, these servers are included in t_3 as well as t_4, so again they are counted twice and the two sides match. Finally, consider the servers that are in $\mathcal{N}_{T+2}$ but are neither in U nor among the neighbors of the servers in U. On the LHS all these servers are included in t_1, while on the RHS they are included in t_3. Thus on both sides these servers are included once, and the two sides match. Finally, note that the servers in U are not included in any term on either the LHS or the RHS. Thus, we have verified that (3.117) holds.

Now, let us recall that according to (3.29),

$$\nu_2(G[\mathcal{N}_{T+2}]) = \min\{|\mathcal{N}_{T+2}\backslash U| + |\mathcal{N}(U) \cap \mathcal{N}_{T+2}| \Big| \text{such that } U \subset \mathcal{N}_{T+2}, U \text{ is a stable set}\}. \tag{3.118}$$

Therefore, minimizing over $U \in \mathcal{N}_{T+2}$, the scheme achieves the normalized download,

$$D_1 + \cdots + D_N = \frac{\nu_2(G[\mathcal{N}_{T+2}])}{2} + |\mathcal{N}_{T+1}|, \tag{3.119}$$

and therefore we have a lower bound on capacity,

$$C_\infty \geq \frac{2}{\nu_2(G[\mathcal{N}_{T+2}]) + 2|\mathcal{N}_{T+1}|}. \tag{3.120}$$

Because the achievable scheme works for any number of messages, it is notable that this lower bound holds not only for asymptotic capacity, but also for capacity with arbitrary number of messages K_m. This completes the proof of achievability for Theorem 3.3. □

3.7 Discussion

The asymptotic capacity of GXSTPIR studied in this chapter reveals important insights into the structure of optimal schemes for graph-based replicated storage. In particular the special structure inspired by dual GRS codes emerges as a powerful idea for GXSTPIR.

Chapter 4

X-secure T-private Information Retrieval from MDS Coded Storage with Byzantine and Unresponsive Servers

The problem of X-secure T-private information retrieval from MDS coded storage is studied in this chapter, where the user wishes to privately retrieve one out of K independent messages that are distributed over N servers according to an MDS code. It is guaranteed that any group of up to X colluding servers learn nothing about the messages and that any group of up to T colluding servers learn nothing about the identity of desired message. A lower bound of achievable rates is proved by presenting a novel scheme based on *cross-subspace alignment* and a successive decoding with interference cancellation strategy. For large number of messages ($K \to \infty$) the achieved rate, which we conjecture to be optimal, improves upon the best known rates previously reported in the literature by Raviv and Karpuk, and generalizes an achievable rate for MDS-TPIR previously found by Freij-Hollanti et al. that is also conjectured to be asymptotically optimal. The setting is then expanded to allow unresponsive and Byzantine servers. Finally, the scheme is applied to find a new lower convex hull of (download, upload) pairs of secure and private distributed matrix multiplication that generalizes, and in certain asymptotic settings strictly improves upon the best known previous results.

4.1 Introduction

The study of PIR is important from an information theoretic perspective not only because privacy is important, but also because optimal PIR schemes often reveal novel coding structures, thereby advancing our understanding of structured codes, a cornerstone of network information theory. The fundamental significance of these coding structures is emphasized by the connections between PIR and a number of other important problems such as locally decodable codes [62, 136], locally repairable codes [43], batch codes [46], oblivious transfer [87, 40], instance hiding [2, 23], secret sharing [95], blind interference alignment [47, 101], and secure computation [133], including recent works on secure distributed matrix multiplication [18, 32, 60, 3, 51, 17]. As the literature on information theoretic PIR continues to grow, it is also valuable to find unified perspectives that combine our understanding of various aspects of PIR and allow generalizations beyond PIR. Against this background, the contribution of this chapter is summarized in Figure 4.1.

An important concern in systems that rely on distributed servers is that some of these servers may turn out to be unresponsive or Byzantine (return erroneous responses)[68, 114, 19, 98]. In this chapter we study the problem of U-B-MDS-XSTPIR, i.e., PIR with X-secure data, T-private queries, (N,K_c) MDS coded storage, where U servers are unresponsive and up to B servers are Byzantine. In particular we show that a rate of $R^{\infty}_{\text{U-B-MDS-XSTPIR}} = 1-\left(\frac{K_c+X+T+2B-1}{N-U}\right)$ is achievable for any number of messages K. This rate strictly improves upon the previous best known rate $R=\left(1-\left(\frac{K_c+X+T+2B-1}{N-U}\right)\right)\left(\frac{K_c}{K_c+X}\right)$ for U-B-MDS-

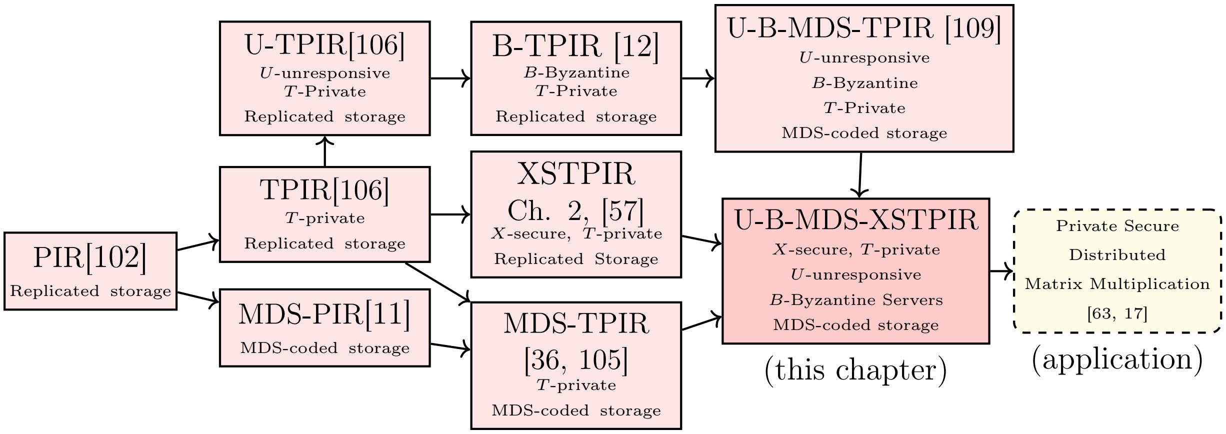


Figure 4.1: The U-B-MDS-XSTPIR setting studied in this chapter generalizes previously studied settings of PIR [102], TPIR [106], MDS-PIR [11], MDS-TPIR [36, 105], XSTPIR [57], U-TPIR [106], B-TPIR [12], and U-B-MDS-TPIR [109] as shown, and finds application beyond PIR in the context of Private Secure Distributed Matrix Multiplication (PSDMM).

XSTPIR, found[1] in [88]. In fact, for MDS-XSTPIR, i.e., with $U=B=0$, we conjecture that our rate of $R^{\infty}_{\text{MDS-XSTPIR}}=1-\left(\frac{K_c+X+T-1}{N}\right)$ is asymptotically optimal as $K\to\infty$, thus generalizing a previous conjecture for MDS-TPIR in [36] that can be obtained by further setting $X=0$. Remarkably, U-B-MDS-XSTPIR is a generalization of PIR, TPIR, MDS-PIR, XSTPIR, U-TPIR, B-TPIR, and U-B-MDS-TPIR and the asymptotically optimal (or the best known) structured coding schemes for all of these problems can be obtained as a special case of the unified scheme for U-B-MDS-XSTPIR that we present in this chapter. The basis for this unified view, and the central technical contribution of this chapter, is a scheme that combines the cross-subspace alignment idea in Chapter 2 with a layered structure that allows successive decoding and interference cancellation to retrieve multiple layers of symbols from the desired message. The scheme is also shown to be applicable to the problem of secure and private distributed matrix multiplication (PSDMM) that was recently introduced in [63, 17]. Remarkably, the new scheme is able to generalize, and in certain asymptotic settings strictly improve upon the previously best known rates for PSDMM.

[1] Reference [88] considers the problem of private polynomial computation with Lagrange encoding, which reduces to U-B-MDS-XSTPIR in the special case where the functions to be computed are all distinct coordinate projections.

4.2 Problem Statement: U-B-MDS-XSTPIR

Consider K independent messages, $W_1, W_2, \ldots, W_K$. Each message is represented by ℓ uniformly random symbols from the finite field $\mathbb{F}_q$.

$$H(W_1) = H(W_2) = \cdots = H(W_K) = \ell, \tag{4.1}$$

$$H(W_{[K]}) = K\ell, \tag{4.2}$$

in q-ary units. Note that as is typical in information theory, the message sizes are unbounded, and the coding scheme may freely choose the block size ℓ. The information stored at the n^{th} server is denoted by S_n, $n \in [N]$. Messages are stored among N servers according to an MDS$(N, X+K_c)$ code which codes each message separately. From any $X+K_c$ servers, it must be possible to recover all messages.

$$H(W_{[K]} | S_{\mathcal{M}}) = 0, \quad \forall \mathcal{M} \subset [N], |\mathcal{M}| = X + K_c. \tag{4.3}$$

The storage requirement at each server is $K\ell/K_c$, i.e.,

$$H(S_n) = \frac{K\ell}{K_c}, \quad \forall n \in [N]. \tag{4.4}$$

Thus, compared to replicated storage, the storage requirement is reduced by a factor of $1/K_c$. X-secure storage, $0 \leq X \leq N$, guarantees that any X (or fewer) colluding servers learn nothing about the messages.

$$I(S_{\mathcal{X}}; W_{[K]}) = 0, \quad \forall \mathcal{X} \subset [N], |\mathcal{X}| = X. \tag{4.5}$$

The user privately and uniformly generates the index of his desired message $\theta \in [K]$. To retrieve the desired message privately, the user generates N queries, $Q_{[N]}^{\theta}$. The n^{th} query Q_n^{θ}

is sent to the n^{th} server. The user has no prior knowledge of the information stored at the servers, i.e.,

$$I(S_{[N]};\theta,Q^{\theta}_{[N]})=0. \tag{4.6}$$

T-privacy, $0\leq T\leq N$, guarantees that any T (or fewer) colluding servers learn nothing about the desired message index θ.

$$I(Q^{\theta}_{\mathcal{T}},S_{\mathcal{T}};\theta)=0, \quad \forall\mathcal{T}\subset[N],|\mathcal{T}|=T. \tag{4.7}$$

Upon receiving the user's query Q^{θ}_{n}, the n^{th} server responds with the answer A^{θ}_{n}.

There exists a set of servers $\mathcal{B}$, $\mathcal{B}\subset[N],|\mathcal{B}|\leq B$, known as Byzantine servers, and another (disjoint) set of servers $\mathcal{U}$, $\mathcal{U}\subset[N],|\mathcal{U}|=U$, known as unresponsive servers. The user knows U,B but the realizations of the sets $\mathcal{U},\mathcal{B}$, are not known to the user *apriori*. The Byzantine servers respond to the user arbitrarily, possibly introducing errors. The unresponsive servers do not respond at all. However, the remaining servers, i.e., servers in $[N]\setminus(\mathcal{B}\cup\mathcal{U})$, respond to the user truthfully with a function of the query and their stored information.

$$H(A^{\theta}_{n}|Q^{\theta}_{n},S_{n})=0, \quad \forall n\in[N]\setminus(\mathcal{B}\cup\mathcal{U}). \tag{4.8}$$

The user must be able to recover the desired message W_θ from the responses that he receives.

$$H(W_\theta \mid A^{\theta}_{[N]\setminus\mathcal{U}},Q^{\theta}_{[N]},\theta)=0 \quad \forall\mathcal{U},\mathcal{B}\subset[N],\mathcal{U}=U,\mathcal{B}=B,\mathcal{U}\cap\mathcal{B}=\emptyset. \tag{4.9}$$

The rate of a U-B-MDS-XSTPIR scheme is defined by the number of bits of desired message that are retrieved per total bit of download from all servers on average,

$$R_{\text{U-B-MDS-XSTPIR}}=\frac{H(W_\theta)}{\sum_{n\in[N]\setminus\mathcal{U}}A^{\theta}_{n}}=\frac{\ell}{D}. \tag{4.10}$$

$D=\sum_{n\in[N]\setminus\mathcal{U}} A_n^\theta$ is the expected number of downloaded bits from all servers. When $B=0,U=0$, i.e., there are no Byzantine servers and no unresponsive servers, then we refer to the problem simply as MDS-XSTPIR.

4.3 Result: An Achievable Rate for U-B-MDS-XSTPIR

Theorem 4.1. *The following rate is achievable for U-B-MDS-XSTPIR,*

$$R_{\textit{U-B-MDS-XSTPIR}}(N,K_c,X,T,U,B,K)=1-\left(\frac{K_c+X+T+2B-1}{N-U}\right). \tag{4.11}$$

The achievability of this rate, proved in Section 4.4, is the central contribution of this chapter. It is based on a coding scheme that uses cross-subspace alignment along with a layered structure that allows successive decoding with interference cancellation. Note that previously the best known achievable result for U-B-MDS-XSTPIR for large number of messages ($K\rightarrow\infty$) was $R=\left(1-\left(\frac{K_c+X+T+2B-1}{N-U}\right)\right)\left(\frac{K_c}{K_c+X}\right)$, found in [88]. Evidently our scheme achieves a strictly higher rate. While we conjecture that the rate in Theorem 4.1 for MDS-XSTPIR ($U=0,B=0$) is also the asymptotic capacity of MDS-XSTPIR, a converse proof to this effect remains beyond reach. This is to be expected, because the converse proof has also been unavailable for MDS-TPIR, which is a special case of MDS-XSTPIR. Our final result appears in Section 4.5 where the result of Theorem 4.1 is applied to the problem of Private Secure Distributed Matrix Multiplication.

4.4 Proof of Theorem 4.1

First we provide the proof of achievability for $U=0,B=0$, i.e., with no unresponsive or Byzantine servers. Throughout the scheme, let us define

$$L=N-(K_c+X+T-1). \tag{4.12}$$

and let us set

$$\ell=LK_c. \tag{4.13}$$

Let us start with an illustrative example.

4.4.1 $X=1,T=1,K_c=2,N=4$

Here we have $L=1$ and $\ell=2$. So let each message consist of $\ell=2$ symbols from a finite field $\mathbb{F}_q$, where $q\geq L+N=5$. Let $\mathbf{W}_{11}$ and $\mathbf{W}_{12}$ be two $1\times K$ row vectors containing the first and second symbol from every message, respectively. Let $\mathbf{Z}_{11}$ be a uniformly distributed random noise vector from $\mathbb{F}_q^{1\times K}$, that will be used to provide $X=1$ security for the stored data. Let $\mathbf{Z}'^{1}_{11}$, $\mathbf{Z}'^{2}_{11}$ be independent, uniformly distributed random noise vectors from $\mathbb{F}_q^{K\times 1}$ that will be used to provide $T=1$ privacy for the queries. Let $\mathbf{Q}_\theta$ be the θ-th column of the $K\times K$ identity matrix, where θ is the index of desired message. The independence between messages, noise vectors, and desired message index θ is formalized as follows.

$$H(\mathbf{W}_{11},\mathbf{W}_{12},\mathbf{Z}_{11},\mathbf{Z}'^{1}_{11},\mathbf{Z}'^{2}_{11},\theta)=H(\mathbf{W}_{11})+H(\mathbf{W}_{12})+H(\mathbf{Z}_{11})+H(\mathbf{Z}'^{1}_{11})+H(\mathbf{Z}'^{2}_{11})+H(\theta). \tag{4.14}$$

Note that by the definition of $\mathbf{W}_{11}$, $\mathbf{W}_{12}$ and $\mathbf{Q}_\theta$, the inner products $\mathbf{W}_{11}\mathbf{Q}_\theta$ and $\mathbf{W}_{12}\mathbf{Q}_\theta$ are precisely the two symbols of the desired message, that the user wishes to retrieve. Let $f_1,\alpha_1,\alpha_2,\cdots,\alpha_N$, represent $N+1$ distinct elements of $\mathbb{F}_q$. The storage at the n-th server is constructed as follows.

$$S_n=\left(\frac{1}{(f_1-\alpha_n)^2}\mathbf{W}_{11}+\frac{1}{f_1-\alpha_n}\mathbf{W}_{12}+\mathbf{Z}_{11}\right), \tag{4.15}$$

Thus, the data is coded along with the noise according to an MDS(N,K_c+X), i.e., MDS(4,3) code. The presence of noise guarantees that the data is $(X=1)$ secure. The query sent by the user to the n-th server to privately retrieve the θ-th message, consists of $K_c=2$ rounds, which are denoted as $Q_n^{\theta,1}$ and $Q_n^{\theta,2}$ respectively.

$$Q_n^{\theta,1}=(f_1-\alpha_n)\mathbf{Q}_\theta+(f_1-\alpha_n)^2\mathbf{Z}_{11}'^1, \tag{4.16}$$

$$Q_n^{\theta,2}=\mathbf{Q}_\theta+(f_1-\alpha_n)^2\mathbf{Z}_{11}'^2. \tag{4.17}$$

Upon receiving the query from user, the answer returned by the n-th server is

$$A_n^\theta=(S_nQ_n^{\theta,1},S_nQ_n^{\theta,2}). \tag{4.18}$$

Now let us see why correctness is guaranteed. We rewrite $S_nQ_n^{\theta,1}$ as

$$S_nQ_n^{\theta,1}=\left(\frac{1}{(f_1-\alpha_n)^2}\mathbf{W}_{11}+\frac{1}{f_1-\alpha_n}\mathbf{W}_{12}+\mathbf{Z}_{11}\right)\left((f_1-\alpha_n)\mathbf{Q}_\theta+(f_1-\alpha_n)^2\mathbf{Z}_{11}'^1\right) \tag{4.19}$$

$$=\frac{1}{f_1-\alpha_n}\mathbf{W}_{11}\mathbf{Q}_\theta+\underbrace{\left(\mathbf{W}_{11}\mathbf{Z}_{11}'^1+\mathbf{W}_{12}\mathbf{Q}_\theta\right)}_{I_1}$$

$$+(f_1-\alpha_n)\underbrace{\left(\mathbf{W}_{12}\mathbf{Z}_{11}'^1+\mathbf{Z}_{11}\mathbf{Q}_\theta\right)}_{I_2}+(f_1-\alpha_n)^2\underbrace{\mathbf{Z}_{11}\mathbf{Z}_{11}'^1}_{I_3}. \tag{4.20}$$

Now, note that the terms $1,(f_1-\alpha_n),(f_1-\alpha_n)^2$, can each be expanded into weighted sums of the terms $1,\alpha_n,\alpha_n^2$. Re-grouping terms according to this expansion, and collecting $S_nQ_n^{\theta,1}$

terms from the answers received from all $N=4$ servers, we obtain

$$\begin{bmatrix} S_1 Q_1^{\theta,1} \\ S_2 Q_2^{\theta,1} \\ S_3 Q_3^{\theta,1} \\ S_4 Q_4^{\theta,1} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_1-\alpha_1} & 1 & \alpha_1 & \alpha_1^2 \\ \frac{1}{f_1-\alpha_2} & 1 & \alpha_2 & \alpha_2^2 \\ \frac{1}{f_1-\alpha_3} & 1 & \alpha_3 & \alpha_3^2 \\ \frac{1}{f_1-\alpha_4} & 1 & \alpha_4 & \alpha_4^2 \end{bmatrix}}_{\mathbf{M}_{1,4}} \begin{bmatrix} \mathbf{W}_{11}\mathbf{Q}_\theta \\ I_1 + f_1 I_2 + f_1^2 I_3 \\ -I_2 - f_1 I_3 - f_1 I_3 \\ I_3 \end{bmatrix} \tag{4.21}$$

Since the 4×4 matrix $\mathbf{M}_{1,4}$ is invertible according to Lemma 2.5, the user is able to retrieve his first desired symbol, $\mathbf{W}_{11}\mathbf{Q}_\theta$. Now, in order to retrieve his second desired symbol, $\mathbf{W}_{11}\mathbf{Q}_\theta$, the user will use successive decoding along with cancellation of interference from the previously retrieved desired symbol. Consider the second part of the answer received from each server, $S_n Q_n^{\theta,2}$, which can be written as follows.

$$S_n Q_n^{\theta,2} = \left(\frac{1}{(f_1-\alpha_n)^2}\mathbf{W}_{11} + \frac{1}{f_1-\alpha_n}\mathbf{W}_{12} + \mathbf{Z}_{11}\right)\left(\mathbf{Q}_\theta + (f_1-\alpha_n)^2 \mathbf{Z}_{11}'^{2}\right) \tag{4.22}$$

$$= \frac{1}{(f_1-\alpha_n)^2}\underbrace{\mathbf{W}_{11}\mathbf{Q}_\theta}_{I_0'} + \frac{1}{f_1-\alpha_n}\mathbf{W}_{12}\mathbf{Q}_\theta$$

$$+ \underbrace{(\mathbf{W}_{11}\mathbf{Z}_{11}'^{2} + \mathbf{Z}_{11}\mathbf{Q}_\theta)}_{I_1'} + (f_1-\alpha_n)\underbrace{\mathbf{W}_{12}\mathbf{Z}_{11}'^{2}}_{I_2'} + (f_1-\alpha_n)^2 \underbrace{\mathbf{Z}_{11}\mathbf{Z}_{11}'^{2}}_{I_3'} \tag{4.23}$$

Aside from the desired symbol $\mathbf{W}_{12}\mathbf{Q}_\theta$, there are four *interference* terms I_0', I_1', I_2', I_3'. Now, since the user has already retrieved $\mathbf{W}_{11}\mathbf{Q}_\theta$, he can subtract I_0' from $S_n Q_n^{\theta,2}$. Furthermore, like before, the remaining interference terms can be expanded along α_n^t, $t\in\{0,1,2\}$. Thus the user is able to obtain

$$\begin{bmatrix} S_1 Q_1^{\theta,2} - \frac{I_0'}{(f_1-\alpha_1)^2} \\ S_2 Q_2^{\theta,2} - \frac{I_0'}{(f_1-\alpha_2)^2} \\ S_3 Q_3^{\theta,2} - \frac{I_0'}{(f_1-\alpha_3)^2} \\ S_4 Q_4^{\theta,2} - \frac{I_0'}{(f_1-\alpha_4)^2} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_1-\alpha_1} & 1 & \alpha_1 & \alpha_1^2 \\ \frac{1}{f_1-\alpha_2} & 1 & \alpha_2 & \alpha_2^2 \\ \frac{1}{f_1-\alpha_3} & 1 & \alpha_3 & \alpha_3^2 \\ \frac{1}{f_1-\alpha_4} & 1 & \alpha_4 & \alpha_4^2 \end{bmatrix}}_{\mathbf{M}_{1,4}} \begin{bmatrix} \mathbf{W}_{12}\mathbf{Q}_\theta \\ I_1' + f_1 I_2' + f_1^2 I_3' \\ -I_2' - f_1 I_3' - f_1 I_3' \\ I_3' \end{bmatrix} \tag{4.24}$$

	Server 'n' (Replace α with α_n)
Storage(S_n)	$\frac{1}{(f_1-\alpha)^2}\mathbf{W}_{11}+\frac{1}{f_1-\alpha}\mathbf{W}_{12}+\mathbf{Z}_{11}$
Query ($Q_n^{[\theta]}$)	$(f_1-\alpha)\mathbf{Q}_\theta+(f_1-\alpha)^2\mathbf{Z}'^1_{11}$ - $\mathbf{Q}_\theta+(f_1-\alpha)^2\mathbf{Z}'^2_{11}$
Desired symbols appear along vectors $\overrightarrow{(f_1-\alpha)^{-1}}$ - $\boxed{\overrightarrow{(f_1-\alpha)^{-2}}},\overrightarrow{(f_1-\alpha)^{-1}}$	
Interference appears along vectors $\overrightarrow{1},\overrightarrow{(f_1-\alpha)},\overrightarrow{(f_1-\alpha)^2}\equiv\overrightarrow{1},\overrightarrow{\alpha},\overrightarrow{\alpha^2}$	

Table 4.1: A summary of the MDS-XSTPIR scheme for $X=1,T=1,K_c=2,N=4,U=0,B=0$, showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

from which, by inverting the matrix $\mathbf{M}_{1,4}$, the user is able to retrieve his second desired symbol, $\mathbf{W}_{12}\mathbf{Q}_\theta$. This completes the proof of correctness.

For ease of reference, a compact summary of the storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server, is provided in Table 4.1. Queries and answers of each round are partitioned with dashed lines. Recovered desired symbols from previous rounds that can be canceled appear along vectors that are wrapped with rounded-corner boxes.

$T=1$-privacy and $X=1$-security follows from the fact that queries and storage are protected by the i.i.d. uniformly distributed noise vectors $\mathbf{Z}_{11}$ and $\mathbf{Z}'^1_{11}$, $\mathbf{Z}'^2_{11}$ respectively. Finally, let us calculate the rate achieved by the scheme. From 8 downloaded q-ary symbols, the user retrieves 2 desired q-ary symbols, so the rate achieved is $R=2/8=1/4=1-3/4$. This completes the proof of achievability for the setting $U=B=0,X=1,T=1,K_c=2,N=4$.

$X=1,T=1,K_c=2,N=5$

Here we have $L=N-(X+T+K_c-1)=2$ and $\ell=LK_c=4$. So let each message consist of $\ell=4$ symbols from a finite field $\mathbb{F}_q$, where $q\geq L+N=7$. Let $\mathbf{W}_{11},\mathbf{W}_{21},\mathbf{W}_{12},\mathbf{W}_{22}$ be four $1\times K$ row vectors containing the four symbols from every message, respectively. Let $\mathbf{Z}_{11},\mathbf{Z}_{21}$ be two independent, uniformly distributed random noise vectors from $\mathbb{F}_q^{1\times K}$ that will be used to guarantee $X=1$ security. Similarly, let $\mathbf{Z}_{11}^{'1},\mathbf{Z}_{21}^{'1}$, $\mathbf{Z}_{11}^{'2},\mathbf{Z}_{21}^{'2}$ be independent, uniformly distributed random noise vectors from $\mathbb{F}_q^{K\times 1}$ that will be used to guarantee $T=1$ privacy. As before, let $\mathbf{Q}_\theta$ be the θ-th column of the $K\times K$ identity matrix, where θ is the index of desired message. The desired message W_θ can be represented as

$$\begin{aligned}W_\theta &= (\mathbf{W}_{lk}\mathbf{Q}_\theta)_{l\in[2],k\in[2]} &\quad (4.25)\\ &= (\mathbf{W}_{11}\mathbf{Q}_\theta,\mathbf{W}_{12}\mathbf{Q}_\theta,\mathbf{W}_{21}\mathbf{Q}_\theta,\mathbf{W}_{22}\mathbf{Q}_\theta). &\quad (4.26)\end{aligned}$$

The independence between messages, noise vectors, and desired message index θ is specified as follows.

$$\begin{aligned}&H(\mathbf{W}_{11},\mathbf{W}_{21},\mathbf{W}_{12},\mathbf{W}_{22},\mathbf{Z}_{11},\mathbf{Z}_{21},\mathbf{Z}_{11}^{'1},\mathbf{Z}_{21}^{'1},\mathbf{Z}_{11}^{'2},\mathbf{Z}_{21}^{'2},\theta)\\ &=\sum_{l\in[2],k\in[2]} H(\mathbf{W}_{lk})+H(\mathbf{Z}_{11})+H(\mathbf{Z}_{21})+H(\mathbf{Z}_{11}^{'1})+H(\mathbf{Z}_{21}^{'1})+H(\mathbf{Z}_{11}^{'2})+H(\mathbf{Z}_{21}^{'2})+H(\theta).\end{aligned} \quad (4.27)$$

Let $f_1,f_2,\alpha_1,\alpha_2,\cdots,\alpha_5$ be $L+N=2+5=7$ distinct elements of $\mathbb{F}_q$, $q\geq 7$. The storage at the n-th server is constructed as follows.

$$S_n=(S_{n1},S_{n2}), \quad (4.28)$$

where

$$S_{n1} = \frac{1}{(f_1-\alpha_n)^2}\mathbf{W}_{11} + \frac{1}{f_1-\alpha_n}\mathbf{W}_{12} + \mathbf{Z}_{11}, \tag{4.29}$$

$$S_{n2} = \frac{1}{(f_2-\alpha_n)^2}\mathbf{W}_{21} + \frac{1}{f_2-\alpha_n}\mathbf{W}_{22} + \mathbf{Z}_{21} \tag{4.30}$$

so that each of (4.29) and (4.30) codes noise with message symbols across N servers according to an MDS(N, K_c+X) code, guaranteeing $X=1$ security on top of MDS coded storage. The query sent to the n-th server to retrieve the θ^{th} message consists of $K_c=2$ rounds, $Q_n^{\theta,1}$ and $Q_n^{\theta,2}$. Furthermore, we will set

$$Q_n^{\theta,1} = (Q_{n1}^{\theta,1}, Q_{n2}^{\theta,1}) \tag{4.31}$$

$$Q_n^{\theta,2} = (Q_{n1}^{\theta,2}, Q_{n2}^{\theta,2}) \tag{4.32}$$

where

$$Q_{n1}^{\theta,1} = (f_1-\alpha_n)\mathbf{Q}_\theta + (f_1-\alpha_n)^2\mathbf{Z}_{11}'^{1}, \tag{4.33}$$

$$Q_{n2}^{\theta,1} = (f_2-\alpha_n)\mathbf{Q}_\theta + (f_2-\alpha_n)^2\mathbf{Z}_{21}'^{1}, \tag{4.34}$$

$$Q_{n1}^{\theta,2} = \mathbf{Q}_\theta + (f_1-\alpha_n)^2\mathbf{Z}_{11}'^{2}, \tag{4.35}$$

$$Q_{n2}^{\theta,2} = \mathbf{Q}_\theta + (f_2-\alpha_n)^2\mathbf{Z}_{21}'^{2}. \tag{4.36}$$

Upon receiving the query from user, the answer returned by the n-th server is comprised of two symbols,

$$A_n^\theta = (A_{n1}^\theta, A_{n2}^\theta) \tag{4.37}$$

$$= (S_{n1}Q_{n1}^{\theta,1} + S_{n2}Q_{n2}^{\theta,1}, \quad S_{n1}Q_{n1}^{\theta,2} + S_{n2}Q_{n2}^{\theta,2}). \tag{4.38}$$

Now let us see why correctness is guaranteed. Consider the first symbol, A^{θ}_{n1}.

$$\begin{aligned}
A^{\theta}_{n1} &= S_{n1}Q^{\theta,1}_{n1} + S_{n2}Q^{\theta,1}_{n2} \\
&= \frac{1}{f_1-\alpha_n}\mathbf{W}_{11}\mathbf{Q}_{\theta} + \frac{1}{f_2-\alpha_n}\mathbf{W}_{21}\mathbf{Q}_{\theta} + (\mathbf{W}_{11}\mathbf{Z}'^{1}_{11} + \mathbf{W}_{21}\mathbf{Z}'^{1}_{21} + \mathbf{W}_{12}\mathbf{Q}_{\theta} + \mathbf{W}_{22}\mathbf{Q}_{\theta}) \\
&\quad + (f_1-\alpha_n)(\mathbf{W}_{12}\mathbf{Z}'^{1}_{11} + \mathbf{Z}_{11}\mathbf{Q}_{\theta}) + (f_2-\alpha_n)(\mathbf{W}_{22}\mathbf{Z}'^{1}_{21} + \mathbf{Z}_{21}\mathbf{Q}_{\theta}) \\
&\quad + (f_1-\alpha_n)^2\mathbf{Z}_{11}\mathbf{Z}'^{1}_{11} + (f_1-\alpha_n)^2\mathbf{Z}_{21}\mathbf{Z}'^{1}_{21}. \qquad (4.39)
\end{aligned}$$

The first two terms in (4.39) are desired message symbols. Each of the remaining 5 terms can be expanded into weighted sums of terms of the form α_n^t, $t \in \{0,1,2\}$, allowing the user to represent the symbols A^{θ}_{n1} downloaded from all $n \in [N]$ servers, as

$$\begin{bmatrix} A^{\theta}_{11} \\ A^{\theta}_{21} \\ A^{\theta}_{31} \\ A^{\theta}_{41} \\ A^{\theta}_{51} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_1-\alpha_1} & \frac{1}{f_2-\alpha_1} & 1 & \alpha_1 & \alpha_1^2 \\ \frac{1}{f_1-\alpha_2} & \frac{1}{f_2-\alpha_2} & 1 & \alpha_2 & \alpha_2^2 \\ \frac{1}{f_1-\alpha_3} & \frac{1}{f_2-\alpha_3} & 1 & \alpha_3 & \alpha_3^2 \\ \frac{1}{f_1-\alpha_4} & \frac{1}{f_2-\alpha_4} & 1 & \alpha_4 & \alpha_4^2 \\ \frac{1}{f_1-\alpha_5} & \frac{1}{f_2-\alpha_5} & 1 & \alpha_5 & \alpha_5^2 \end{bmatrix}}_{\mathbf{M}_{2,5}} \begin{bmatrix} \mathbf{W}_{11}\mathbf{Q}_{\theta} \\ \mathbf{W}_{21}\mathbf{Q}_{\theta} \\ * \\ * \\ * \end{bmatrix} \qquad (4.40)$$

where we have used $*$ to represent various combinations of interference symbols that can be found explicitly by expanding (4.39), since those forms are not important. What matters is that the 5×5 square matrix in (4.40) is $\mathbf{M}_{2,5}$ which is invertible according to Lemma 2.5, so the user can retrieve the two desired symbols, $\mathbf{W}_{11}\mathbf{Q}_{\theta}$, $\mathbf{W}_{21}\mathbf{Q}_{\theta}$ by inverting the matrix. Next, the user needs to retrieve the remaining two desired symbols $\mathbf{W}_{12}\mathbf{Q}_{\theta}$, $\mathbf{W}_{22}\mathbf{Q}_{\theta}$, for which we will use successive decoding with interference cancellation. Consider the downloaded symbol A^{θ}_{n2}.

$$\begin{aligned}
A^{\theta}_{n2} &= S_{n1}Q^{\theta,2}_{n1} + S_{n2}Q^{\theta,2}_{n2} \\
&= \frac{1}{(f_1-\alpha_n)^2}\mathbf{W}_{11}\mathbf{Q}_{\theta} + \frac{1}{(f_2-\alpha_n)^2}\mathbf{W}_{21}\mathbf{Q}_{\theta} + \frac{1}{f_1-\alpha_n}\mathbf{W}_{12}\mathbf{Q}_{\theta} + \frac{1}{f_2-\alpha_n}\mathbf{W}_{22}\mathbf{Q}_{\theta}
\end{aligned}$$

$$
\begin{aligned}
&+(\mathbf{W}_{11}\mathbf{Z}_{11}'^{2}+\mathbf{W}_{21}\mathbf{Z}_{21}'^{2}+\mathbf{Z}_{11}\mathbf{Q}_{\theta}+\mathbf{Z}_{21}\mathbf{Q}_{\theta})+(f_1-\alpha_n)\mathbf{W}_{12}\mathbf{Z}_{11}'^{2}+(f_2-\alpha_n)\mathbf{W}_{22}\mathbf{Z}_{21}'^{2}\\
&+(f_1-\alpha_n)^2\mathbf{Z}_{11}\mathbf{Z}_{11}'^{2}+(f_2-\alpha_n)^2\mathbf{Z}_{21}\mathbf{Z}_{21}'^{2}.
\end{aligned}
\tag{4.41}
$$

The first two symbols in (4.41) are desired symbols that have already been decoded. So these terms can be subtracted out, leaving the user with the following downloaded information from all $N=5$ servers.

$$
\begin{bmatrix}
A_{12}^{\theta}-\frac{1}{(f_1-\alpha_1)^2}\mathbf{W}_{11}\mathbf{Q}_{\theta}-\frac{1}{(f_2-\alpha_1)^2}\mathbf{W}_{21}\mathbf{Q}_{\theta}\\
A_{22}^{\theta}-\frac{1}{(f_1-\alpha_2)^2}\mathbf{W}_{11}\mathbf{Q}_{\theta}-\frac{1}{(f_2-\alpha_2)^2}\mathbf{W}_{21}\mathbf{Q}_{\theta}\\
A_{32}^{\theta}-\frac{1}{(f_1-\alpha_3)^2}\mathbf{W}_{11}\mathbf{Q}_{\theta}-\frac{1}{(f_2-\alpha_3)^2}\mathbf{W}_{21}\mathbf{Q}_{\theta}\\
A_{42}^{\theta}-\frac{1}{(f_1-\alpha_4)^2}\mathbf{W}_{11}\mathbf{Q}_{\theta}-\frac{1}{(f_2-\alpha_4)^2}\mathbf{W}_{21}\mathbf{Q}_{\theta}\\
A_{52}^{\theta}-\frac{1}{(f_1-\alpha_5)^2}\mathbf{W}_{11}\mathbf{Q}_{\theta}-\frac{1}{(f_2-\alpha_5)^2}\mathbf{W}_{21}\mathbf{Q}_{\theta}
\end{bmatrix}
=
\underbrace{\begin{bmatrix}
\frac{1}{f_1-\alpha_1} & \frac{1}{f_2-\alpha_1} & 1 & \alpha_1 & \alpha_1^2\\
\frac{1}{f_1-\alpha_2} & \frac{1}{f_2-\alpha_2} & 1 & \alpha_2 & \alpha_2^2\\
\frac{1}{f_1-\alpha_3} & \frac{1}{f_2-\alpha_3} & 1 & \alpha_3 & \alpha_3^2\\
\frac{1}{f_1-\alpha_4} & \frac{1}{f_2-\alpha_4} & 1 & \alpha_4 & \alpha_4^2\\
\frac{1}{f_1-\alpha_5} & \frac{1}{f_2-\alpha_5} & 1 & \alpha_5 & \alpha_5^2
\end{bmatrix}}_{\mathbf{M}_{2,5}}
\begin{bmatrix}
\mathbf{W}_{12}\mathbf{Q}_{\theta}\\
\mathbf{W}_{22}\mathbf{Q}_{\theta}\\
*\\
*\\
*
\end{bmatrix}
\tag{4.42}
$$

Once again, the 5×5 square matrix in (4.42) is $\mathbf{M}_{2,5}$ which is invertible according to Lemma 2.5, so the user can retrieve his remaining two desired symbols, $\mathbf{W}_{12}\mathbf{Q}_{\theta}$, $\mathbf{W}_{22}\mathbf{Q}_{\theta}$ by inverting the matrix. This completes the proof of correctness. Let us summarize the storage at each server, the queries, and the partitioning of signal and interference dimensions contained in the answers from each server in Table 4.2. $T=1$-privacy and $X=1$-security follows from the fact that queries and storage are protected by the i.i.d. uniformly distributed noise vectors. Now consider the rate achieved by the scheme. Since the user downloads 2 symbols from each of 5 servers, we note that from a total of 10 downloaded q-ary symbols, the user is able to recover 4 desired q-ary symbols, so the rate achieved is $R=4/10=2/5=1-3/5$. This completes the construction of the scheme for the setting $U=B=0, X=1, T=1, K_c=2, N=5$. We now specify the scheme for $U=B=0$ and arbitrary X,T,K_c,N parameters.

$U=B=0$, **arbitrary** X,T,K_c,N

Let each message consist of $\ell=LK_c$ symbols from a finite field $\mathbb{F}_q$ where $L=N-(X+T+K_c-1)$ and $q\geq L+N$. Let $\mathbf{W}_{lk},l\in[L],k\in[K_c]$ be $1\times K$ row vectors. For each value of $\in[L],k\in[K_c]$, the $1\times K$ row vector $\mathbf{W}_{lk}$ contains the $(L(k-1)+l)^{th}$ symbol from every message. Let $(\mathbf{Z}_{lx})_{l\in[L],x\in[X]}$ be independent, uniformly distributed random noise vectors from $\mathbb{F}_q^{1\times K}$ that will be used to guarantee X-security. Let $(\mathbf{Z}_{lt}'^{\kappa})_{l\in[L],t\in[T],\kappa\in[K_c]}$ be independent, uniformly distributed random noise vectors from $\mathbb{F}_q^{K\times 1}$ that will be used to guarantee that the queries are T-private. As before, let $\mathbf{Q}_\theta$ be the θ-th column of the $K\times K$ identity matrix, where θ is the index of desired message. The desired message W_θ can be represented as,

$$W_\theta=(\mathbf{W}_{lk}\mathbf{Q}_\theta)_{l\in[L],k\in[K_c]} \tag{4.43}$$

	Server 'n' (Replace α with α_n)
Storage (S_n)	$\frac{1}{(f_1-\alpha)^2}\mathbf{W}_{11}+\frac{1}{f_1-\alpha}\mathbf{W}_{12}+\mathbf{Z}_{11}$ $\frac{1}{(f_2-\alpha)^2}\mathbf{W}_{21}+\frac{1}{f_2-\alpha}\mathbf{W}_{22}+\mathbf{Z}_{21}$
Query ($Q_n^{[\theta]}$)	$(f_1-\alpha)\mathbf{Q}_\theta+(f_1-\alpha)^2\mathbf{Z}_{11}'^{1}$ $(f_2-\alpha)\mathbf{Q}_\theta+(f_2-\alpha)^2\mathbf{Z}_{21}'^{1}$ $\mathbf{Q}_\theta+(f_1-\alpha)^2\mathbf{Z}_{11}'^{2}$ $\mathbf{Q}_\theta+(f_2-\alpha)^2\mathbf{Z}_{21}'^{2}$
Desired symbols appear along vectors $\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{(f_2-\alpha)^{-1}}$ $\boxed{\overrightarrow{(f_1-\alpha)^{-2}}},\boxed{\overrightarrow{(f_2-\alpha)^{-2}}},\overrightarrow{(f_1-\alpha)^{-1}},\overrightarrow{(f_2-\alpha)^{-1}}$ Interference appears along vectors $\overrightarrow{1},\overrightarrow{(f_1-\alpha)},\overrightarrow{(f_1-\alpha)^2},\overrightarrow{(f_2-\alpha)},\overrightarrow{(f_2-\alpha)^2}\equiv\overrightarrow{1},\overrightarrow{\alpha},\overrightarrow{\alpha^2}$	

Table 4.2: A summary of the MDS-XSTPIR scheme for $X=1,T=1,K_c=2,N=5,U=0,B=0$, showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

$$= \begin{pmatrix} \mathbf{W}_{11}\mathbf{Q}_\theta, & \mathbf{W}_{12}\mathbf{Q}_\theta, & \cdots, & \mathbf{W}_{1K_c}\mathbf{Q}_\theta \\ \mathbf{W}_{21}\mathbf{Q}_\theta, & \mathbf{W}_{22}\mathbf{Q}_\theta, & \cdots, & \mathbf{W}_{2K_c}\mathbf{Q}_\theta \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W}_{L1}\mathbf{Q}_\theta, & \mathbf{W}_{L2}\mathbf{Q}_\theta, & \cdots, & \mathbf{W}_{LK_c}\mathbf{Q}_\theta \end{pmatrix}. \tag{4.44}$$

The independence between messages, noise vectors and θ is formalized as follows.

$$\begin{aligned} & H((\mathbf{W}_{lk})_{l\in[L],k\in[K_c]},(\mathbf{Z}_{lx})_{l\in[L],x\in[X]},(\mathbf{Z}'^{\kappa}_{lt})_{l\in[L],t\in[T],\kappa\in[K_c]},\theta) \\ & = \sum_{l\in[L],k\in[K_c]} H(\mathbf{W}_{lk}) + \sum_{l\in[L],x\in[X]} H(\mathbf{Z}_{lx}) + \sum_{l\in[L],t\in[T],\kappa\in[K_c]} H(\mathbf{Z}'^{\kappa}_{lt}) + H(\theta). \end{aligned} \tag{4.45}$$

Let $f_1,f_2,\cdots,f_L,\alpha_1,\alpha_2,\cdots,\alpha_N$ be $L+N$ distinct elements of $\mathbb{F}_q$. Since $q\geq N+L$, these constants must exist. The storage at the n^{th} server is comprised of L symbols $(S_{nl})_{l\in[L]}$, i.e.,

$$S_n = (S_{n1},S_{n,2},\ldots,S_{nL}). \tag{4.46}$$

For all $l\in[L]$, S_{nl} is constructed as

$$S_{nl} = \frac{1}{(f_l-\alpha_n)^{K_c}}\mathbf{W}_{l1} + \frac{1}{(f_l-\alpha_n)^{K_c-1}}\mathbf{W}_{l2} + \cdots + \frac{1}{f_l-\alpha_n}\mathbf{W}_{lK_c} + \sum_{x\in[X]}(f_l-\alpha_n)^{x-1}\mathbf{Z}_{lx} \tag{4.47}$$

$$= \sum_{k\in[K_c]} \frac{1}{(f_l-\alpha_n)^{K_c-k+1}}\mathbf{W}_{lk} + \sum_{x\in[X]}(f_l-\alpha_n)^{x-1}\mathbf{Z}_{lx}. \tag{4.48}$$

Thus, for each $l\in[L]$, the values S_{nl} stored across all N servers comprise an MDS(N,K_c+X) code which includes X noise symbols for X-security. The query sent by the user to the n-th server, in order to retrieve the θ^{th} desired message, is comprised of K_c rounds, $(Q_n^{\theta,\kappa})_{\kappa\in[K_c]}$. For each $\kappa\in[K_c]$, the query is constructed as follows.

$$Q_n^{\theta,\kappa} = (Q_{n1}^{\theta,\kappa},Q_{n2}^{\theta,\kappa},\ldots,Q_{nL}^{\theta,\kappa}), \tag{4.49}$$

where $\forall l \in [L]$, let us set

$$Q_{nl}^{\theta,\kappa} = (f_l - \alpha_n)^{K_c-\kappa}\mathbf{Q}_\theta + \sum_{t\in[T]}(f_l-\alpha_n)^{K_c+t-1}\mathbf{Z}_{lt}^{\prime\kappa}. \tag{4.50}$$

Upon receiving the query from the user, the n-th server responds with the following K_c symbols.

$$A_n^\theta = (A_{n1}^\theta, A_{n2}^\theta, \cdots, A_{nK_c}^\theta) \tag{4.51}$$

where for all $\kappa \in [K_c]$,

$$A_{n\kappa}^\theta = (S_{n1}Q_{n1}^{\theta,\kappa} + S_{n2}Q_{n2}^{\theta,\kappa} + \cdots + S_{nL}Q_{nL}^{\theta,\kappa}). \tag{4.52}$$

To show that the scheme is correct, for any $\kappa \in [K_c]$, let us rewrite the symbol $A_{n\kappa}^\theta$ as,

$$\begin{aligned}
A_{n\kappa}^\theta &= \sum_{l\in[L]} S_{nl}Q_{nl}^{\theta,\kappa} && (4.53)\\
&= \sum_{l\in[L]}\left(\sum_{k\in[K_c]}\frac{1}{(f_l-\alpha_n)^{K_c-k+1}}\mathbf{W}_{lk} + \sum_{x\in[X]}(f_l-\alpha_n)^{x-1}\mathbf{Z}_{lx}\right)\\
&\qquad\left((f_l-\alpha_n)^{K_c-\kappa}\mathbf{Q}_\theta + \sum_{t\in[T]}(f_l-\alpha_n)^{K_c+t-1}\mathbf{Z}_{lt}^{\prime\kappa}\right) && (4.54)\\
&= \sum_{l\in[L]}\sum_{k\in[\kappa]}\frac{1}{(f_l-\alpha_n)^{\kappa-k+1}}\mathbf{W}_{lk}\mathbf{Q}_\theta + \sum_{l\in[L]}\sum_{k=\kappa+1}^{K_c}(f_l-\alpha_n)^{k-\kappa-1}\mathbf{W}_{lk}\mathbf{Q}_\theta\\
&\quad + \sum_{l\in[L]}\sum_{x\in[X]}(f_l-\alpha_n)^{K_c-\kappa+x-1}\mathbf{Z}_{lx}\mathbf{Q}_\theta + \sum_{l\in[L]}\sum_{k\in[K_c]}\sum_{t\in[T]}(f_l-\alpha_n)^{k+t-2}\mathbf{W}_{lk}\mathbf{Z}_{lt}^{\prime\kappa}\\
&\quad + \sum_{l\in[L]}\sum_{x\in[X]}\sum_{t\in[T]}(f_l-\alpha_n)^{K_c+t+x-2}\mathbf{Z}_{lx}\mathbf{Z}_{lt}^{\prime\kappa}. && (4.55)
\end{aligned}$$

Now we will see why it is possible to recover all desired symbols $(\mathbf{W}_{lk}\mathbf{Q}_\theta)_{l\in[L],k\in[K_c]}$. Consider $\kappa=1$.

$$A_{n1}^\theta=\sum_{l\in[L]}S_{nl}Q_{nl}^{\theta,1} \tag{4.56}$$

$$\begin{aligned}&=\sum_{l\in[L]}\frac{1}{f_l-\alpha_n}\mathbf{W}_{l1}\mathbf{Q}_\theta+\sum_{l\in[L]}\sum_{k=2}^{K_c}(f_l-\alpha_n)^{k-2}\mathbf{W}_{lk}\mathbf{Q}_\theta+\sum_{l\in[L]}\sum_{x\in[X]}(f_l-\alpha_n)^{K_c+x-2}\mathbf{Z}_{lx}\mathbf{Q}_\theta\\&\quad+\sum_{l\in[L]}\sum_{k\in[K_c]}\sum_{t\in[T]}(f_l-\alpha_n)^{t+k-2}\mathbf{W}_{lk}\mathbf{Z}_{lt}'^1+\sum_{l\in[L]}\sum_{x\in[X]}\sum_{t\in[T]}(f_l-\alpha_n)^{K_c+t+x-2}\mathbf{Z}_{lx}\mathbf{Z}_{lt}'^1\end{aligned} \tag{4.57}$$

The first term contains the L desired symbols $(\mathbf{W}_{11}\mathbf{Q}_\theta,\ldots,\mathbf{W}_{L1}\mathbf{Q}_\theta)$ that are to be retrieved in the first round, i.e., for $\kappa=1$. Each of the remaining four terms constitute interference which can be expanded into weighted sums of terms of the form α_n^t, $t\in\{0,1,\ldots,K_c+X+T-2\}$. Therefore, collecting the A_{n1}^θ symbols from all N servers, the user obtains

$$\begin{bmatrix}A_{11}^\theta\\A_{21}^\theta\\\vdots\\A_{N1}^\theta\end{bmatrix}=\underbrace{\begin{bmatrix}\frac{1}{f_1-\alpha_1}&\cdots&\frac{1}{f_L-\alpha_1}&1&\alpha_1&\cdots&\alpha_1^{K_c+X+T-2}\\\frac{1}{f_1-\alpha_2}&\cdots&\frac{1}{f_L-\alpha_2}&1&\alpha_2&\cdots&\alpha_2^{K_c+X+T-2}\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\\frac{1}{f_1-\alpha_N}&\cdots&\frac{1}{f_L-\alpha_N}&1&\alpha_N&\cdots&\alpha_N^{K_c+X+T-2}\end{bmatrix}}_{\mathbf{M}_{L,N}}\begin{bmatrix}\mathbf{W}_{11}\mathbf{Q}_\theta\\\mathbf{W}_{21}\mathbf{Q}_\theta\\\vdots\\\mathbf{W}_{L1}\mathbf{Q}_\theta\\*\\\vdots\\*\end{bmatrix} \tag{4.58}$$

where $*$ represents various combinations of interference terms, whose precise forms are inconsequential. What matters is that the $N\times N$ matrix in (4.58) $\mathbf{M}_{L,N}$ is invertible according to Lemma 2.5, so that the user is able to retrieve the desired symbols $(\mathbf{W}_{11}\mathbf{Q}_\theta,\ldots,\mathbf{W}_{L1}\mathbf{Q}_\theta)$ by inverting the matrix.

The scheme proceeds similarly to retrieve desired symbols $(\mathbf{W}_{1\kappa}\mathbf{Q}_\theta,\ldots,\mathbf{W}_{L\kappa}\mathbf{Q}_\theta)$ with the κ^{th}

round of queries. To prove this by induction, let us consider any κ, such that $2 \le \kappa \le K_c$, and assume that the desired symbols $(\mathbf{W}_{lk}\mathbf{Q}_\theta)_{l\in[L],k\in[\kappa-1]}$ have already been retrieved. Now we wish to show that the desired symbols $(\mathbf{W}_{l\kappa}\mathbf{Q}_\theta)_{l\in[L]}$ can be retrieved.

$$\begin{aligned}
A_{n\kappa}^{\theta} &= \sum_{l\in[L]} S_{nl} Q_{nl}^{\theta,\kappa} && (4.59)\\
&= \sum_{l\in[L]}\sum_{k\in[\kappa-1]} \frac{1}{(f_l-\alpha_n)^{\kappa-k+1}} \mathbf{W}_{lk}\mathbf{Q}_\theta + \sum_{l\in[L]} \frac{1}{f_l-\alpha_n}\mathbf{W}_{l\kappa}\mathbf{Q}_\theta \\
&\quad + \sum_{l\in[L]}\sum_{k=\kappa+1}^{K_c} (f_l-\alpha_n)^{k-\kappa-1}\mathbf{W}_{lk}\mathbf{Q}_\theta + \sum_{l\in[L]}\sum_{x\in[X]} (f_l-\alpha_n)^{K_c-\kappa+x-1}\mathbf{Z}_{lx}\mathbf{Q}_\theta && (4.60)\\
&\quad + \sum_{l\in[L]}\sum_{k\in[K_c]}\sum_{t\in[T]} (f_l-\alpha_n)^{t+k-2}\mathbf{W}_{lk}\mathbf{Z}_{lt}^{\prime\kappa} + \sum_{l\in[L]}\sum_{x\in[X]}\sum_{t\in[T]} (f_l-\alpha_n)^{K_c+t+x-2}\mathbf{Z}_{lx}\mathbf{Z}_{lt}^{\prime\kappa}. && (4.61)
\end{aligned}$$

The first term contains symbols that have already been retrieved, so the user can subtract this term from $A_{n\kappa}^{\theta}$.

$$A_{n\kappa}^{\theta'} = A_{n\kappa}^{\theta} - \sum_{l\in[L]}\sum_{k\in[\kappa-1]} \frac{1}{(f_l-\alpha_n)^{\kappa-k+1}} \mathbf{W}_{lk}\mathbf{Q}_\theta. \quad (4.62)$$

The next term is comprised of the L symbols $(\mathbf{W}_{l\kappa}\mathbf{Q}_\theta)_{l\in[L]}$ that the user wishes to retrieve. The remaining 4 terms constitute interference which can be expanded as before into weighted sums of terms of the form $\alpha_n^t, t \in \{0,1,\ldots,K_c+X+T-2\}$. Therefore, collecting the $A_{n\kappa}^{\theta'}$

symbols from all N servers, the user obtains,

$$\begin{bmatrix} A_{11}^{\theta'} \\ A_{21}^{\theta'} \\ \vdots \\ A_{N1}^{\theta'} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_1-\alpha_1} & \cdots & \frac{1}{f_L-\alpha_1} & 1 & \alpha_1 & \cdots & \alpha_1^{K_c+X+T-2} \\ \frac{1}{f_1-\alpha_2} & \cdots & \frac{1}{f_L-\alpha_2} & 1 & \alpha_2 & \cdots & \alpha_2^{K_c+X+T-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \cdots & \frac{1}{f_L-\alpha_N} & 1 & \alpha_N & \cdots & \alpha_N^{K_c+X+T-2} \end{bmatrix}}_{\mathbf{M}_{L,N}} \begin{bmatrix} \mathbf{W}_{1\kappa}\mathbf{Q}_\theta \\ \mathbf{W}_{2\kappa}\mathbf{Q}_\theta \\ \vdots \\ \mathbf{W}_{L\kappa}\mathbf{Q}_\theta \\ * \\ \vdots \\ * \end{bmatrix} \tag{4.63}$$

The desired symbols $(\mathbf{W}_{l\kappa}\mathbf{Q}_\theta)_{l\in[L]}$ can be retrieved by inverting the $N\times N$ square matrix in (4.63), which is guaranteed to be invertible according to Lemma 2.5. Thus, the induction argument shows that all $\ell = LK_c$ desired symbols are retrieved successfully. A summary of the storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server is provided in Table 4.3.

T-privacy is guaranteed because $\mathbf{Q}_\theta$ is protected by the noise vectors $(\mathbf{Z}_{lt}'^{\kappa})_{l\in[L],t\in[T],\kappa\in[K_c]}$ that are i.i.d. uniform and coded according to an MDS(N,T) code. Similarly, X-security is guaranteed because for each $l\in[L]$, the messages $(\mathbf{W}_{lk})_{k\in[K_c]}$ are protected by the noise vectors $(\mathbf{Z}_{lx})_{x\in[X]}$ that are i.i.d. uniform and coded according to an MDS(N,X) code. Now let us consider the rate achieved by the scheme. From a total of NK_c downloaded q-ary symbols, the user is able to retrieve his $\ell = LK_c$ desired symbols, so the rate achieved is

$$R = \frac{LK_c}{NK_c} = \frac{L}{N} = 1 - \left(\frac{K_c+X+T-1}{N}\right), \tag{4.64}$$

which matches the result in Theorem 4.1.

Server ‘n’ (Replace α with α_n)	
Storage (S_n)	$\frac{1}{(f_1-\alpha)^{K_c}}\mathbf{W}_{11}+\cdots+\frac{1}{f_1-\alpha}\mathbf{W}_{1K_c}+\mathbf{Z}_{11}+\cdots+(f_1-\alpha)^{X-1}\mathbf{Z}_{1X}$ $\frac{1}{(f_2-\alpha)^{K_c}}\mathbf{W}_{21}+\cdots+\frac{1}{f_2-\alpha}\mathbf{W}_{2K_c}+\mathbf{Z}_{21}+\cdots+(f_2-\alpha)^{X-1}\mathbf{Z}_{2X}$ $\vdots$ $\frac{1}{(f_L-\alpha)^{K_c}}\mathbf{W}_{L1}+\cdots+\frac{1}{f_L-\alpha}\mathbf{W}_{L'K_c}+\mathbf{Z}_{L1}+\cdots+(f_L-\alpha)^{X-1}\mathbf{Z}_{LX}$
Query $(Q_n^{[\theta]})$	$(f_1-\alpha)^{K_c-1}\mathbf{Q}_\theta+(f_1-\alpha)^{K_c}\mathbf{Z}'^{1}_{11}+\cdots+(f_1-\alpha)^{K_c+T-1}\mathbf{Z}'^{1}_{1T}$ $(f_2-\alpha)^{K_c-1}\mathbf{Q}_\theta+(f_2-\alpha)^{K_c}\mathbf{Z}'^{1}_{21}+\cdots+(f_2-\alpha)^{K_c+T-1}\mathbf{Z}'^{1}_{2T}$ $\vdots$ $(f_L-\alpha)^{K_c-1}\mathbf{Q}_\theta+(f_L-\alpha)^{K_c}\mathbf{Z}'^{1}_{L1}+\cdots+(f_L-\alpha)^{K_c+T-1}\mathbf{Z}'^{1}_{LT}$
	- -
	$(f_1-\alpha)^{K_c-2}\mathbf{Q}_\theta+(f_1-\alpha)^{K_c}\mathbf{Z}'^{2}_{11}+\cdots+(f_1-\alpha)^{K_c+T-1}\mathbf{Z}'^{2}_{1T}$ $(f_2-\alpha)^{K_c-2}\mathbf{Q}_\theta+(f_2-\alpha)^{K_c}\mathbf{Z}'^{2}_{21}+\cdots+(f_2-\alpha)^{K_c+T-1}\mathbf{Z}'^{2}_{2T}$ $\vdots$ $(f_L-\alpha)^{K_c-2}\mathbf{Q}_\theta+(f_L-\alpha)^{K_c}\mathbf{Z}'^{2}_{L'1}+\cdots+(f_L-\alpha)^{K_c+T-1}\mathbf{Z}'^{2}_{LT}$
	- -
	$\vdots$
	- -
	$\mathbf{Q}_\theta+(f_1-\alpha)^{K_c}\mathbf{Z}'^{K_c}_{11}+\cdots+(f_1-\alpha)^{K_c+T-1}\mathbf{Z}'^{K_c}_{1T}$ $\mathbf{Q}_\theta+(f_2-\alpha)^{K_c}\mathbf{Z}'^{K_c}_{21}+\cdots+(f_2-\alpha)^{K_c+T-1}\mathbf{Z}'^{K_c}_{2T}$ $\vdots$ $\mathbf{Q}_\theta+(f_L-\alpha)^{K_c}\mathbf{Z}'^{K_c}_{f_{L'}1}+\cdots+(f_L-\alpha)^{K_c+T-1}\mathbf{Z}'^{K_c}_{LT}$
Desired symbols appear along vectors $\overrightarrow{(f_1-\alpha)^{-1}},\cdots,\overrightarrow{(f_L-\alpha)^{-1}}$	
- -	
$\boxed{\overrightarrow{(f_1-\alpha)^{-2}}},\cdots,\boxed{\overrightarrow{(f_L-\alpha)^{-2}}},\overrightarrow{(f_1-\alpha)^{-1}},\cdots,\overrightarrow{(f_L-\alpha)^{-1}}$	
- -	
$\vdots$	
- -	
$\boxed{\overrightarrow{(f_1-\alpha)^{-K_c}}},\cdots,\boxed{\overrightarrow{(f_L-\alpha)^{-K_c}}},\cdots,\boxed{\overrightarrow{(f_1-\alpha)^{-2}}},\cdots,\boxed{\overrightarrow{(f_L-\alpha)^{-2}}},$ $\overrightarrow{(f_1-\alpha)^{-1}},\cdots,\overrightarrow{(f_L-\alpha)^{-1}}$	
Interference appears along vectors $\overrightarrow{1},\overrightarrow{(f_1-\alpha)},\cdots,\overrightarrow{(f_1-\alpha)^{X+T+K_c-2}},\cdots,\overrightarrow{(f_L-\alpha)},\cdots,\overrightarrow{(f_L-\alpha)^{X+T+K_c-2}}$ $\equiv\overrightarrow{1},\overrightarrow{\alpha},\cdots,\overrightarrow{\alpha^{X+T+K_c-2}}$	

Table 4.3: A summary of the general MDS-XSTPIR scheme showing storage at each server, the queries, and a partitioning of signal and interference dimensions contained in the answers from each server.

4.4.2 Arbitrary U, B

Now let us generalize the scheme to non-trivial U and B, i.e., for U unresponsive servers and up to B byzantine servers. For this generalization, let us set

$$L = (N-U)-(K_c+X+T+2B-1) \tag{4.65}$$

$$\ell = LK_c. \tag{4.66}$$

Even though now the values of U,B are non-trivial, the construction of storage, queries and answers remains identical to the description provided previously for $U=B=0$. So let us consider any $(N-U)$ responsive servers, say servers $n_1,n_2,\cdots,n_{N-U}$. Instead of the $N\times N$ square matrix $\mathbf{M}_{L,N}$ in (4.63), we now have the $(N-U)\times(N-U-2B)$ decoding matrix,

$$\mathbf{M}_{(N-U)\times(N-U-2B)} = \begin{bmatrix} \frac{1}{f_1-\alpha_{n_1}} & \cdots & \frac{1}{f_L-\alpha_{n_1}} & 1 & \alpha_{n_1} & \cdots & \alpha_{n_1}^{K_c+X+T-2} \\ \frac{1}{f_1-\alpha_{n_2}} & \cdots & \frac{1}{f_L-\alpha_{n_2}} & 1 & \alpha_{n_2} & \cdots & \alpha_{n_2}^{K_c+X+T-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_{n_{N-U}}} & \cdots & \frac{1}{f_L-\alpha_{n_{N-U}}} & 1 & \alpha_{n_{N-U}} & \cdots & \alpha_{n_{N-U}}^{K_c+X+T-2} \end{bmatrix}. \tag{4.67}$$

Note that if we consider any $N-U-2B$ rows of $\mathbf{M}_{(N-U)\times(N-U-2B)}$ then we obtain an invertible square matrix because of Lemma 2.5. Therefore, $\mathbf{M}_{(N-U)\times(N-U-2B)}$ is the generator matrix of an MDS$(N-U,N-U-2B)$ code, and it is can correct up to $((N-U)-(N-U-2B))/2=B$ errors. Thus by this construction, we establish a scheme that works with U unresponsive servers and up to B Byzantine servers, while achieving the rate of

$$R = 1-\left(\frac{K_c+X+T+2B-1}{N-U}\right). \tag{4.68}$$

This completes the proof of Theorem 4.1.

4.5 Private and Secure Distributed Matrix Multiplication

Recently in [63, 17], the problem of private and secure matrix multiplication (PSDMM) is proposed, where a user wishes to compute the product of a confidential matrix $\mathbf{A}$ with a matrix $\mathbf{B}_\theta, \theta \in [M]$ with the aid of N distributed servers. In [17], it is assumed that the set of matrices $\mathbf{B}_{[M]}$ are public and available to the N servers, however, the confidential matrix $\mathbf{A}$ is shared secretly among all N servers, such that no information about $\mathbf{A}$ is leaked to any server. Besides, the user wants to keep the index θ private from each server. The goal of the problem is to minimize (i) the upload cost from the source of the confidential matrix $\mathbf{A}$ to the N servers and (ii) the download cost from the N servers to the user. In [17], the authors exploit the MDS-PIR scheme proposed in [11] to construct the PSDMM scheme, and characterize the lower convex hull of (upload, download) pairs.

Using the MDS-XSTPIR scheme present in Section 4.4, we now present a novel PSDMM scheme for a generalized model. In our model, the index θ is T-private, while the confidential matrix $\mathbf{A}$ is X_A-secure. Furthermore, we also allow matrices $\mathbf{B}_{[M]}$ to be X_B-secure. Note that the model in [17] is obtained as a special case of our generalized model by setting $X_A = T = 1, X_B = 0$.

4.5.1 PSDMM: Problem Statement

Let $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_\ell)$ represent ℓ random matrices, each of dimension $\lambda \times \chi$, that are independently and uniformly distributed over $\mathbb{F}_q^{\lambda \times \chi}$. Let $\mathbf{B}_{[M]}$ be M random matrices independently and uniformly distributed over $\mathbb{F}_q^{\chi \times \mu}$. The independence between matrices $\mathbf{A}_{[\ell]}$

and $\mathbf{B}_{[M]}$ is formalized as follows.

$$H(\mathbf{A},\mathbf{B}_{[M]}) = \sum_{l\in[\ell]} H(\mathbf{A}_l) + \sum_{m\in[M]} H(\mathbf{B}_m). \tag{4.69}$$

The matrices $\mathbf{A}$ and $\mathbf{B}_{[M]}$ are made available at N distributed servers through secret sharing schemes with security levels X_A and X_B, respectively. That is, any group of up to X_A colluding servers can learn nothing about $\mathbf{A}$, and any group of up to X_B servers can learn nothing about $\mathbf{B}_{[M]}$. To this end, matrices $\mathbf{A}$ and $\mathbf{B}_{[M]}$ are separately coded according to secret sharing schemes that generate shares $\tilde{A}_n$, $\tilde{B}_n, n \in [N]$, and these shares are made available to the n-th server. Furthermore, we assume that the upload cost of $\tilde{A}_{[N]}$ is to be optimized, while that of $\tilde{B}_{[N]}$ and $Q^{\theta}_{[N]}$ is ignored, presumably because $\mathbf{A}$ matrices are frequently updated while $\mathbf{B}_{[M]}$ are static, and the size of queries does not scale with ℓ.

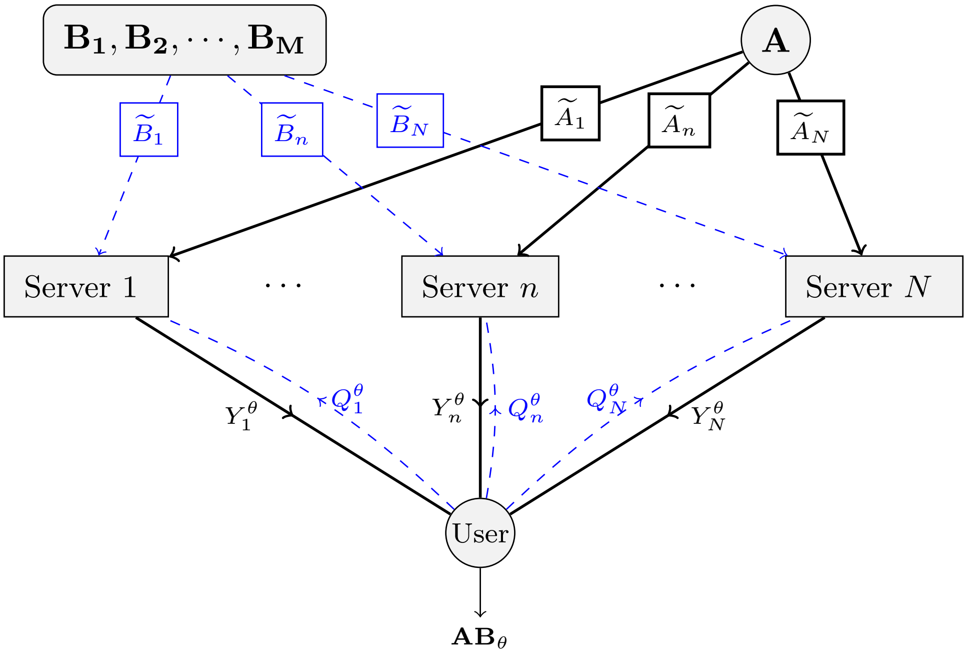


Figure 4.2: Model for private secure distributed matrix multiplication (PSDMM). $\mathbf{A}$ matrices are X_A secure, while $\mathbf{B}$ matrices are X_B secure. The uploads to be optimized are the $\widetilde{A}$ terms and the downloads to be optimized are the Y^θ terms.

The independence between the securely coded matrices is specified as follows.

$$I(\mathbf{A},\tilde{A}_{[N]};\mathbf{B}_{[M]},\tilde{B}_{[N]})=0. \tag{4.70}$$

Matrices must be recoverable from their secret shares.

$$H(\mathbf{A}\mid\tilde{A}_{[N]})=0, \tag{4.71}$$

$$H(\mathbf{B}_{[M]}\mid\tilde{B}_{[N]})=0. \tag{4.72}$$

The matrices must be perfectly secure from any set of secret shares that can be accessed by a set of up to X_A,X_B colluding servers, respectively.

$$I(\mathbf{A};\tilde{A}_{\mathcal{X}})=0 \quad \mathcal{X}\subset[N],|\mathcal{X}|=X_A, \tag{4.73}$$

$$I(\mathbf{B}_{[M]};\tilde{B}_{\mathcal{X}})=0 \quad \mathcal{X}\subset[N],|\mathcal{X}|=X_B. \tag{4.74}$$

The user generates an index $\theta\in[M]$ privately and uniformly, and wishes to compute the product

$$\mathbf{A}\mathbf{B}_\theta=(\mathbf{A}_1\mathbf{B}_\theta,\mathbf{A}_2\mathbf{B}_\theta,\ldots,\mathbf{A}_\ell\mathbf{B}_\theta). \tag{4.75}$$

To this end, the user generates N queries $Q^\theta_{[N]}$. The n-th query Q^θ_n is sent to the n-th server. The user has no prior knowledge of matrices $\mathbf{A}$ and $\mathbf{B}_{[M]}$ and their secret shares, i.e.,

$$I(\theta,Q^\theta_{[N]};\tilde{A}_{[N]},\tilde{B}_{[N]})=0. \tag{4.76}$$

T-privacy, $0\leq T\leq N$, guarantees that any group of up to T colluding servers learn nothing about θ.

$$I(Q^\theta_{\mathcal{T}},\tilde{A}_{\mathcal{T}},\tilde{B}_{\mathcal{T}};\theta)=0. \tag{4.77}$$

Upon receiving the user's query Q_n^θ, the n-th server responds with an answer Y_n^θ, which is a function of all information available to it.

$$H(Y_n^\theta|Q_n^\theta,\tilde{A}_n,\tilde{B}_n)=0. \tag{4.78}$$

The user must be able to recover the product $\mathbf{AB}_\theta$ from all N answers, i.e.,

$$H(\mathbf{AB}_\theta|Y_{[N]}^\theta,Q_{[N]}^\theta)=0. \tag{4.79}$$

The upload cost and download cost are defined as follows.

$$U=\frac{\sum_{n\in[N]}H(\tilde{A}_n)}{H(\mathbf{A})}, \tag{4.80}$$

$$D=\frac{\sum_{n\in[N]}H(Y_n^\theta)}{H(\mathbf{AB}_\theta)}. \tag{4.81}$$

4.5.2 A New Scheme for PSDMM

In this section, we will present a PSDMM scheme to show that the lower convex hull of (upload, download) pairs

$$(U,D)=\left(\frac{N}{K_c},\ \frac{N}{N-(2K_c+X_A+X_B+T-2)}\right) \tag{4.82}$$

for

$$K_c=1,2,\ldots,\lfloor(N+1-X_A-X_B-T)/2\rfloor \tag{4.83}$$

is achievable when $q\to\infty$ and $\chi\geq\min(\lambda,\mu)$. Furthermore, when $X_B=0$, i.e., there are no security constraints on matrices $\mathbf{B}_{[M]}$, and $\chi\geq\min(\lambda,\mu)$, then the lower convex hull of (upload, download) pairs

$$(U,D)=\left(\frac{N}{K_c},\frac{N}{N-(K_c+X_A+T-1)}\right) \tag{4.84}$$

for

$$K_c=1,2,\dots,(N+1-X_A-T) \tag{4.85}$$

is achievable as $q\to\infty$.

First, let us consider the case $X_B\neq 0$. For this setting, let us set

$$L=N-(X_A+X_B+T+2K_c-2), \tag{4.86}$$

$$\ell=K_cL. \tag{4.87}$$

For all $l\in[L],k\in[K_c]$, let us define

$$\mathbf{A}_{lk}=\mathbf{A}_{L(k-1)+l}. \tag{4.88}$$

We will also set

$$\mathbf{B}=\begin{bmatrix}\mathbf{B}_1\ \mathbf{B}_2\ \dots\ \mathbf{B}_M\end{bmatrix} \tag{4.89}$$

to be an $\chi\times M\mu$ matrix that contains all $\mathbf{B}_{[M]}$. Let us also define $\mathbf{Q}_\theta$ be a $M\mu\times\mu$ matrix as follows.

$$\mathbf{Q}_\theta=[\ \underbrace{\mathbf{0}_\mu\quad\dots\quad\mathbf{0}_\mu}_{\text{A total of }(\theta-1)\mathbf{0}_\mu\text{'s}}\quad \mathbf{I}_\mu\quad \underbrace{\mathbf{0}_\mu\quad\dots\quad\mathbf{0}_\mu}_{\text{A total of }(M-\theta)\mathbf{0}_\mu\text{'s}}\ ]^T \tag{4.90}$$

where $\mathbf{0}_\mu$ is the $\mu\times\mu$ square zero matrix, and $\mathbf{I}_\mu$ is the $\mu\times\mu$ identity matrix. We note that by construction, $\mathbf{A}\mathbf{B}\mathbf{Q}_\theta = (\mathbf{A}_1\mathbf{B}\mathbf{Q}_\theta,\ldots,\mathbf{A}_\ell\mathbf{B}\mathbf{Q}_\theta) = (\mathbf{A}_{lk}\mathbf{B}\mathbf{Q}_\theta)_{l\in[L],k\in[K_c]}$ is the desired product. Let $(\mathbf{Z}_{lx})_{l\in[L],x\in[X_A]}$ and $(\mathbf{Z}'_{lx'})_{l\in[L],x'\in[X_B]}$ be independent, uniformly distributed random noise matrices from $\mathbb{F}_q^{\lambda\times\chi}$ and $\mathbb{F}_q^{\chi\times M\mu}$ that will be used to guarantee X_A and X_B security levels for $\mathbf{A},\mathbf{B}_{[M]}$, respectively. Let $(\mathbf{Z}''^{\kappa}_{lt})_{l\in[L],t\in[T],\kappa\in[K_c]}$ be independent, uniformly distributed random noise matrices from $\mathbb{F}_q^{M\mu\times\mu}$, that will be used to guarantee T-privacy of queries. The independence between $\mathbf{A},\mathbf{B}_{[M]}$, noise matrices and θ is formalized as follows.

$$\begin{aligned}
&H(\mathbf{A},\mathbf{B}_{[M]},(\mathbf{Z}_{lx})_{l\in[L],x\in[X_A]},(\mathbf{Z}'_{lx'})_{l\in[L],x'\in[X_B]},(\mathbf{Z}''^{\kappa}_{lt})_{l\in[L],t\in[T],\kappa\in[K_c]},\theta)\\
&=\sum_{l\in[L],k\in[K_c]} H(\mathbf{A}_{lk}) + \sum_{m\in[M]} H(\mathbf{B}_m) + \sum_{l\in[L],x\in[X_A]} H(\mathbf{Z}_{lx})\\
&\quad+\sum_{l\in[L],x'\in[X_B]} H(\mathbf{Z}'_{lx'}) + \sum_{l\in[L],t\in[T],\kappa\in[K_c]} H(\mathbf{Z}''^{\kappa}_{lt}) + H(\theta). \qquad (4.91)
\end{aligned}$$

Let $f_1,f_2,\cdots,f_L,\alpha_1,\alpha_2,\cdots,\alpha_N$ be distinct elements of $\mathbb{F}_q$. We require $q \geq L+N$ so these elements must exist. Now we are ready to construct the scheme. The secret share of $\mathbf{B}_{[M]}$ at the n-th server, $\tilde{B}_n$ is constructed as follows.

$$\tilde{B}_n = (\tilde{B}_{n1},\tilde{B}_{n2},\ldots,\tilde{B}_{nL}), \qquad (4.92)$$

where $\forall l\in[L]$,

$$\tilde{B}_{nl} = \mathbf{B} + \sum_{x'\in[X_B]} (f_l-\alpha_n)^{K_c+x'-1}\mathbf{Z}'_{lx'}. \qquad (4.93)$$

The secret share of $\mathbf{A}$ at the n^{th} server is constructed as follows.

$$\tilde{A}_n = (\tilde{A}_{n1},\tilde{A}_{n2},\ldots,\tilde{A}_{nL}), \qquad (4.94)$$

where $\forall l \in [L]$,

$$\tilde{A}_{nl} = \sum_{k \in [K_c]} \frac{1}{(f_l - \alpha_n)^{K_c - k + 1}} \mathbf{A}_{lk} + \sum_{x \in [X_A]} (f_l - \alpha_n)^{x-1} \mathbf{Z}_{lx}. \tag{4.95}$$

The query sent by the user to the n^{th} server, is comprised of K_c rounds, $Q_n^{\theta} = (Q_n^{\theta,\kappa})_{\kappa \in [K_c]}$. For all $\kappa \in [K_c]$, we construct the queries as follows.

$$Q_n^{\theta,\kappa} = (Q_{n1}^{\theta,\kappa}, Q_{n2}^{\theta,\kappa}, \ldots, Q_{nL}^{\theta,\kappa}), \tag{4.96}$$

where $\forall l \in [L]$, we set

$$Q_{nl}^{\theta,\kappa} = (f_l - \alpha_n)^{K_c - \kappa} \mathbf{Q}_{\theta} + \sum_{t \in [T]} (f_l - \alpha_n)^{K_c + t - 1} \mathbf{Z}_{lt}''^{\kappa}. \tag{4.97}$$

Upon receiving the query from the user, the n^{th} server responds with the following K_c symbols.

$$Y_n^{\theta} = (\tilde{A}_{n1} \tilde{B}_{n1} Q_{n1}^{\theta,\kappa} + \tilde{A}_{n2} \tilde{B}_{n2} Q_{n2}^{\theta,\kappa} + \cdots + \tilde{A}_{nL} \tilde{B}_{nL} Q_{nL}^{\theta,\kappa})_{\kappa \in [K_c]}. \tag{4.98}$$

To show the correctness of the scheme, let us consider $\tilde{A}_{nl} \tilde{B}_{nl}, \forall l \in [L]$.

$$\begin{aligned} \tilde{A}_{nl} \tilde{B}_{nl} = & \left(\sum_{k \in [K_c]} \frac{1}{(f_l - \alpha_n)^{K_c - k + 1}} \mathbf{A}_{lk} + \sum_{x \in [X_A]} (f_l - \alpha_n)^{x-1} \mathbf{Z}_{lx} \right) \\ & \qquad \left(\mathbf{B} + \sum_{x' \in [X_B]} (f_l - \alpha_n)^{K_c + x' - 1} \mathbf{Z}_{lx'}' \right) && (4.99) \\ = & \sum_{k \in [K_c]} \frac{1}{(f_l - \alpha_n)^{K_c - k + 1}} \mathbf{A}_{lk} \mathbf{B} + \sum_{x \in [X_A]} (f_l - \alpha_n)^{x-1} \mathbf{Z}_{lx} \mathbf{B} \\ & + \sum_{k \in [K_c]} \sum_{x' \in [X_B]} (f_l - \alpha_n)^{x' + k - 2} \mathbf{A}_{lk} \mathbf{Z}_{\ell x'}' + \sum_{x \in [X_A]} \sum_{x' \in [X_B]} (f_l - \alpha_n)^{K_c + x + x' - 2} \mathbf{Z}_{lx} \mathbf{Z}_{lx'}' && (4.100) \end{aligned}$$

$$= \sum_{k\in[K_c]} \frac{1}{(f_l-\alpha_n)^{K_c-k+1}}\mathbf{A}_{lk}\mathbf{B} + \sum_{\xi\in[K_c+X_A+X_B-1]} (f_l-\alpha_n)^{\xi-1}\bar{\mathbf{Z}}_{l\xi} \tag{4.101}$$

In (4.101) we rearranged the last three terms of (4.100) grouping them into weighted sums of terms of the form $(f_l-\alpha_n)^i$, $i\in\{0,1,\ldots,K_c+X_A+X_B-2\}$. The grouped terms $\bar{\mathbf{Z}}_{l\xi}$ can be calculated explicitly but as it turns out the precise form of these terms is inconsequential. Now note that if we regard $(\mathbf{A}_{lk}\mathbf{B})_{l\in[L],k\in[K_c]}$ terms as messages, and other terms as noise, then (4.101) has the same form as (4.48), the storage construction in the MDS-XSTPIR scheme presented in Section 4.4.[2] Also note that the construction of queries is also the same as the MDS-XSTPIR scheme, thus the correctness follows directly from the proof presented in Section 4.4, which means the user is able to recover the product $\mathbf{A}\mathbf{B}\mathbf{Q}_\theta = (\mathbf{A}_{lk}\mathbf{B}\mathbf{Q}_\theta)_{l\in[L],k\in[K_c]}$. Privacy and security follows from the fact that $\mathbf{Q}_\theta$, $\mathbf{A}$, $\mathbf{B}_{[M]}$ are protected by the i.i.d. uniformly distributed noise matrices coded according to MDS(N,T), MDS(X_A,T), MDS(X_B,T) codes, respectively. This completes the construction of the scheme for $X_B\neq 0$. Note that when $q\to\infty$ and $\chi\geq\min(\lambda,\mu)$, then $H(\mathbf{A}\mathbf{B}_\theta)=\ell\lambda\mu$ in q-ary units according to Lemma 5.2 (also see [51], Lemma 2), and the download cost is

$$D=\frac{NK_c\lambda\mu}{\ell\lambda\mu}=\frac{N}{L}=\frac{N}{N-(2K_c+X_A+X_B+T-2)}. \tag{4.102}$$

Now let us consider the case $X_B=0$. For this setting, let us set

$$L=N-(X_A+X_B+T+K_c-1), \tag{4.103}$$

$$\ell=K_cL. \tag{4.104}$$

We will continue using other definitions as before, but since there is no security constraint

[2] Note that X in (4.48) corresponds to $K_c+X_A+X_B-1$ in (4.101), so that $L=N-(X+T+K_c-1)$ in Section 4.4 corresponds to $L=N-(2K_c+X_A+X_B+T-2)$ in this section. The condition on K_c becomes $K_c=\frac{N-(X_A+X_B+T+L-2)}{2}$. However, since we must have $L\geq 1$ and $K_c\geq 1$ can only take integer values, it follows that the feasible values of K_c are $1\leq K_c\leq\lfloor\frac{N-(X_A+X_B+T-1)}{2}\rfloor$.

on $\mathbf{B}$ matrices, let us replace $\tilde{B}_n$ as

$$\tilde{B}_n = \mathbf{B}. \tag{4.105}$$

Now we have

$$\tilde{A}_{nl}\tilde{B}_{nl} = \sum_{k\in[K_c]} \frac{1}{(f_l-\alpha_n)^{K_c-k+1}}\mathbf{A}_{lk}\mathbf{B} + \sum_{x\in[X_A]} (f_l-\alpha_n)^{x-1}\mathbf{Z}_{lx}\mathbf{B}, \tag{4.106}$$

which is coded according to an MDS(N, K_c+X_A) code. Thus the correctness, privacy and security follows from that proof in Section 4.4. The download cost is

$$D = \frac{NK_c\lambda\mu}{L\lambda\mu} = \frac{N}{L} = \frac{N}{N-(K_c+X_A+X_B+T-1)}. \tag{4.107}$$

Now let us consider the upload cost of the scheme. Note that by the construction of $\tilde{A}_n$, it is coded according to an MDS(N,K_c) code. Therefore, the upload cost is $\frac{N}{K_c}$.

It is shown in [17] that when $X_A = T = 1, X_B = 0$, the lower convex hull of (upload, download) pairs

$$(U,D) = \left(\frac{N}{K_c}, \frac{K_c+1}{K_c}\left(1+\left(\frac{K_c+1}{N}\right)+\cdots+\left(\frac{K_c+1}{N}\right)^{M-1}\right)\right) \tag{4.108}$$

is achievable for $K_c = 1,2,\ldots,N-1$. For the asymptotic setting, i.e., $M\to\infty$, we have from [17] that $D = \frac{K_c+1}{K_c}\frac{N}{N-(K_c+1)}$, which is strictly worse than the (upload,download) pairs characterized in this work. This is because the scheme in [17] allows the user to decode noise matrices protecting $\mathbf{A}$, whereas in our scheme, because of *cross-subspace alignment*, the user is only able to decode desired matrices, thus the penalty term $\frac{K_c+1}{K_c}$ disappears.

4.6 Discussion

The problem of U-B-MDS-XSTPIR, i.e., X-secure T-private information retrieval from MDS coded storage, with N servers out of which U are unresponsive and up to B may be Byzantine, is studied in this chapter. A lower bound on achievable rates of U-B-MDS-XSTPIR is characterized by presenting a *cross-subspace alignment* and successive decoding based scheme. We also adapt the scheme to the problem of private and secure distributed matrix multiplication that is recently proposed in [63, 17]. The presented MDS-XSTPIR scheme is shown to be applicable to PSDMM problem, even if we allow security concerns for all constituent matrices.

Chapter 5

Secure Distributed Matrix Multiplication

The problem of secure distributed matrix multiplication (SDMM) studies the communication efficiency of retrieving a sequence of desired matrix products $\mathbf{AB}=(\mathbf{A}_1\mathbf{B}_1,\mathbf{A_2B_2},\cdots,\mathbf{A}_S\mathbf{B}_S)$ from N distributed servers where the constituent matrices $\mathbf{A}=(\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_S)$ and $\mathbf{B}=(\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_S)$ are stored in X-secure coded form, i.e., any group of up to X colluding servers learn nothing about $\mathbf{A},\mathbf{B}$. It is assumed that $\mathbf{A}_s\in\mathbb{F}_q^{L\times K},\mathbf{B}_s\in\mathbb{F}_q^{K\times M},s\in\{1,2,\cdots,S\}$ are uniformly and independently distributed and $\mathbb{F}_q$ is a large finite field. The rate of an SDMM scheme is defined as the ratio of the number of bits of desired information that is retrieved, to the total number of bits downloaded on average. The supremum of achievable rates is called the capacity of SDMM. In this chapter we explore the capacity of SDMM, as well as several of its variants, e.g., where the user may already have either $\mathbf{A}$ or $\mathbf{B}$ available as side-information, and/or where the security constraint for either $\mathbf{A}$ or $\mathbf{B}$ may be relaxed. As our main contribution, we obtain new converse bounds, as well as new achievable schemes for various cases of SDMM, depending on the L,K,M,N,X parameters, and identify parameter regimes where these bounds match. A remarkable aspect of our upper bounds is a connection between SDMM and a form of private information retrieval (PIR) problem, known as multi-message X-secure T-private information retrieval (MM-XSTPIR).

Notable features of our achievable schemes include the use of cross-subspace alignment and a transformation argument that converts a scalar multiplication problem into a scalar addition problem, allowing a surprisingly efficient solution.

5.1 Introduction

Distributed matrix multiplication is a key building block for a variety of applications that include collaborative filtering, object recognition, sensing and data fusion, cloud computing, augmented reality and machine learning. Coding techniques, such as MDS codes [69], Polynomial codes [140], Entangled Polynomial codes [141], MatDot and PolyDot codes [31] and Generalized PolyDot Codes [28] have been shown to be capable of improving the efficiency of distributed matrix multiplication. However, with the expanding scope of distributed computing applications, there are mounting security concerns [138, 129, 18, 61, 32, 3] about sharing information with external servers. The problem of secure distributed matrix multiplication (SDMM) is motivated by these security concerns.

As defined in this chapter, SDMM studies the communication efficiency of retrieving desired matrix products from distributed servers where the constituent matrices are securely stored. Specifically, suppose $\mathbf{A}=(\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_S)$ and $\mathbf{B}=(\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_S)$ are collections of random matrices that are stored across N servers subject to an X-security guarantee, i.e., any colluding group of up to X servers can learn nothing about the $\mathbf{A}$ and $\mathbf{B}$ matrices. Specifically, $\mathbf{A}_s\in\mathbb{F}_q^{L\times K}$, $\mathbf{B}_s\in\mathbb{F}_q^{K\times M}$ for all $s\in\{1,2,\cdots,S\}$, are independent and uniformly distributed, and $\mathbb{F}_q$ is assumed to be a large finite field. A user wishes to retrieve $\mathbf{AB}=(\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\cdots,\mathbf{A}_S\mathbf{B}_S)$, where each $\mathbf{A}_s\mathbf{B}_s, s\in\{1,2,\cdots,S\}$ is an $L\times M$ matrix in $\mathbb{F}_q$, while downloading as little information from the N servers as possible. The rate of an SDMM scheme is defined as the ratio $H(\mathbf{AB})/D$, where $H(\mathbf{AB})$ is the number of bits of desired information that is retrieved, and D is the total number of bits downloaded on average. The

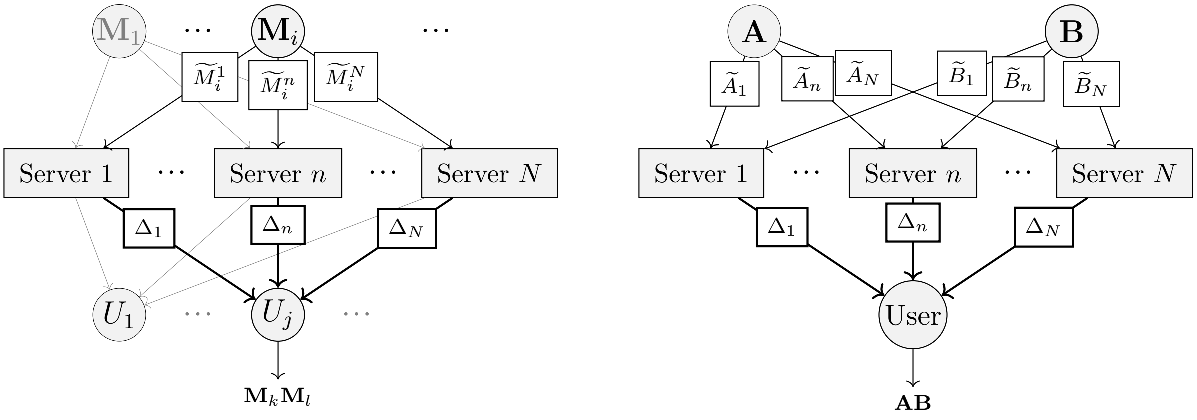


Figure 5.1: (Left) General context for SDMM showing various sources that produce large amounts of data represented as matrices $\mathbf{M}_1,\mathbf{M}_2,\cdots$, and store it at N distributed servers in X-secure form, coded independently as $\widetilde{M}_i^n$. Various authorized users access these servers and retrieve products of their desired matrices based on the downloads that they request from all N servers. Unlike PIR (private information retrieval) [23] problems there are no privacy constraints in SDMM, so users can publicly announce which matrix products they wish to retrieve. (Right) The SDMM problem considered in this chapter, where the goal is to minimize the average size of the total download for a generic user whose desired matrices are labeled $\mathbf{A},\mathbf{B}$.

supremum of achievable rates is called the capacity of SDMM. In this chapter we study the capacity of SDMM, as well as several of its variants, e.g., where the user may already have either $\mathbf{A}$ or $\mathbf{B}$ available as side-information,[1] and/or where the security constraint for either $\mathbf{A}$ or $\mathbf{B}$ may be relaxed. As our main contribution, we obtain new converse bounds, as well as new achievable schemes for various cases of SDMM, depending on the L,K,M,N,X parameters, and identify parameter regimes where these bounds match. A notable aspect of our upper bounds is a connection between SDMM and a form of private information retrieval (PIR) problem, known as multi-message X-secure T-private information retrieval (MM-XSTPIR).[2] Interesting features of our achievable schemes include the idea of cross-subspace alignment that was introduced in Chapter 2 and was recently applied to SDMM in [61], and a novel transformation argument that converts a scalar multiplication problem

[1] The definition of rate takes into account the side-information available at the user. For example, say, the user has $\mathbf{B}$ available as side-information, then the rate is defined as $H(\mathbf{AB}\,|\,\mathbf{B})/D$.

[2] Notably while the capacity of multi-message PIR has been explored in [10, 70] and that of X-secure T-private PIR has been explored in Chapter 2, to our knowledge there has been no prior work on MM-XSTPIR.

into a scalar addition problem. The transformation allows a surprisingly[3] efficient (and capacity optimal) solution for scalar multiplication, outer products of vectors, and Hadamard products of matrices.

Information-theoretic study of SDMM was introduced recently in [18] under a closely related model with a few subtle differences. Motivated by a master-worker model of distributed computation, it is assumed in [18] that the $\mathbf{A}$,$\mathbf{B}$ matrices originate at the user (master), who securely encodes and sends these matrices to N servers (workers), and then from just the downloads that he receives from the servers in return, the user is able to compute $\mathbf{AB}$. The goal in [18], as in this chapter, is to maximize the rate of SDMM, defined as $H(\mathbf{AB})/D$. The same model is pursued in [61, 32, 3], where new coding schemes are proposed for SDMM. However, certain aspects of the model appear inconsonant. For example, a key assumption in these models is that the user must not use his prior knowledge of $\mathbf{A}$,$\mathbf{B}$ and must decode $\mathbf{AB}$ only from the downloads. Since the goal is to minimize the communication cost, and the user already knows $\mathbf{A}$,$\mathbf{B}$, why not do the computation locally and avoid all communication entirely? Indeed, the question is not merely philosophical, because as shown in this chapter, in some cases, the best scheme even with the model of [18, 61, 32, 3] turns out to be one that allows the user to retrieve both $\mathbf{A}$,$\mathbf{B}$ from the downloads, and then compute $\mathbf{AB}$ locally — something that could be done without the need for any communication if indeed $\mathbf{A}$,$\mathbf{B}$ originated at the user. On the other hand, the SDMM model assumed in this chapter assumes that $\mathbf{A}$,$\mathbf{B}$ do not both originate at the user,[4] rather they are stored securely and remotely at the N servers,[5] thus eliminating this concern. Another difference between our model

[3] The achieved rate exceeds an upper bound previously obtained in literature [61]. See the discussion following Theorem 5.2.

[4] One of $\mathbf{A}$,$\mathbf{B}$ may be available to the user as side-information in some variants of SDMM studied in this chapter.

[5] We envision that the matrices $\mathbf{A}$ and $\mathbf{B}$ comprise sensitive data that originates at other sources, and is securely stored at the N servers. The communication cost of uploading the data to the servers is not a focus of this chapter, because the upload requires communication between a different set of entities (sources and servers) with their own separate communication channels and cost dynamics. Furthermore, if the data is of interest to many (authorized) users who perform their desired computations, then the repeated cost of such downloads could very well outweigh the one-time cost of uploading the secured data to the servers.

of SDMM and previous works is that the precise assumptions regarding matrix dimensions L,K,M are left unclear in [18, 61], indeed these dimensions do not appear in the capacity results in [18, 61]. However, our model allows L,K,M to take arbitrary values, including large values. As it turns out, our results reveal that the relative size of L,K,M does matter. For example, capacity-achieving schemes for SDMM from [18, 61] fall short if $K/L<1$, and a converse bound for two-sided security that is derived in [61] is violated under certain conditions as well (see discussion following Theorem 5.2 and Theorem 5.3 in this chapter). Indeed, we find that in general the capacity depends on all matrix dimensions as well as the parameters N,X. Another minor distinction between this chapter and [18, 61] is that we allow joint retrieval of a block of S matrix products $\mathbf{AB}=(\mathbf{A_1B_1},\mathbf{A_2B_2},\cdots,\mathbf{A_SB_S})$ where S can be chosen arbitrarily by the coding scheme. On the other hand, [18, 61] assume one-shot matrix multiplication, corresponding to $S=1$. This leads to different approaches to achievable schemes. For example, while both this chapter and [61] use cross-subspace alignment, we use it to code across S blocks while [61] relies on a matrix partitioning approach.

5.2 Problem Statement: SDMM

Let $\mathbf{A}=(\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_S)$ represent S random matrices, chosen independently and uniformly from all matrices over $\mathbb{F}_q^{L\times K}$. Similarly, let $\mathbf{B}=(\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_S)$ represent S random matrices, chosen independently and uniformly from all matrices over $\mathbb{F}_q^{K\times M}$. The user wishes to compute the products $\mathbf{AB}=(\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\cdots,\mathbf{A}_S\mathbf{B}_S)$.

The independence between matrices $\mathbf{A}_{[S]}$, $\mathbf{B}_{[S]}$ is formalized as follows.

$$H(\mathbf{A},\mathbf{B})=H(\mathbf{A})+H(\mathbf{B})=\sum_{s\in[S]}H(\mathbf{A}_s)+\sum_{s\in[S]}H(\mathbf{B}_s). \tag{5.1}$$

Since we are operating over $\mathbb{F}_q$, let us express all entropies in base q units.

The $\mathbf{A}$,$\mathbf{B}$ matrices are available at N servers with security levels X_A,X_B, respectively. This means that any group of up to X_A colluding servers can learn nothing about $\mathbf{A}$ matrices, and any group of up to X_B colluding servers can learn nothing about the $\mathbf{B}$ matrices.[6] Security is achieved by coding according to secret sharing schemes that separately generate shares $\widetilde{A}_s^n$,$\widetilde{B}_s^n$ corresponding to each $\mathbf{A}_s$,$\mathbf{B}_s$, and make these shares[7] available to the n^{th} server, for all $n\in[N]$. The independence between these securely coded matrices is formalized as,

$$I(\mathbf{A},\widetilde{A}_{[S]}^{[N]};\mathbf{B},\widetilde{B}_{[S]}^{[N]})=0, \tag{5.2}$$

$$H\left(\widetilde{A}_{[S]}^{[N]},\widetilde{B}_{[S]}^{[N]}\right)=\sum_{s\in[S]}H(\widetilde{A}_s^{[N]})+\sum_{s\in[S]}H(\widetilde{B}_s^{[N]}). \tag{5.3}$$

Each matrix must be recoverable from all its secret shares,

$$H(\mathbf{A}_s\mid\widetilde{A}_s^{[N]})=0, \qquad H(\mathbf{B}_s\mid\widetilde{B}_s^{[N]})=0, \qquad \forall s\in[S]. \tag{5.4}$$

The $\mathbf{A}$,$\mathbf{B}$ matrices must be perfectly secure from any set of secret shares that can be accessed by a set of up to X_A,X_B colluding servers, respectively.

$$I\left(\mathbf{A};\widetilde{A}_{[S]}^{\mathcal{X}}\right)=0, \qquad \mathcal{X}\subset[N], \qquad |\mathcal{X}|=X_A \tag{5.5}$$

$$I\left(\mathbf{B};\widetilde{B}_{[S]}^{\mathcal{X}}\right)=0, \qquad \mathcal{X}\subset[N], \qquad |\mathcal{X}|=X_B \tag{5.6}$$

$$I(\mathbf{A},\mathbf{B};\widetilde{A}_{[S]}^{\mathcal{X}},\widetilde{B}_{[S]}^{\mathcal{X}})=0, \qquad \mathcal{X}\subset[N], \qquad |\mathcal{X}|=\min(X_A,X_B) \tag{5.7}$$

In order to retrieve the products $\mathbf{AB}$, from each server $n\in[N]$, the user downloads Δ_n which is function of $\widetilde{A}_{[S]}^n$,$\widetilde{B}_{[S]}^n$.

$$H\left(\Delta_n\mid\widetilde{A}_{[S]}^n,\widetilde{B}_{[S]}^n\right)=0. \tag{5.8}$$

[6] For the most part, we will focus on cases with $X_A,X_B\in\{0,X\}$, i.e., the security level can either be $X>0$ or zero, where a security level zero implies that there is no security constraint for that set of matrices.

[7] If $X_A=0$ then we could choose $\widetilde{A}_s^n=\mathbf{A}_s$. Similarly, if $X_B=0$, then it is possible to have $\widetilde{B}_s^n=\mathbf{B}_s$.

The side-information available to the user *apriori* is denoted Ψ, which can be either the $\mathbf{A}$ matrices, or the $\mathbf{B}$ matrices, or null (ϕ) if the user has no side-information. Given the downloads from all N servers and the side-information, the user must be able to recover the matrix products $\mathbf{AB}$.

$$H\left(\mathbf{AB} \mid \Delta_{[N]}, \Psi\right) = 0. \tag{5.9}$$

Let us define the rate of an SDMM scheme as follows.

$$R = \frac{H(\mathbf{AB} \mid \Psi)}{D} \tag{5.10}$$

where D is the average value (over all realizations of $\mathbf{A}$,$\mathbf{B}$ matrices) of the total number of q-ary symbols downloaded by the user from all N servers. In order to steer away from the field-size concerns that are best left to coding-theoretic studies, we will only allow the field size to be asymptotically large, i.e., $q \to \infty$. The capacity of SDMM is the supremum of achievable rate values over all SDMM schemes and over all S.

Remark 5.1. *The goal of the SDMM problem is to design schemes to minimize D. The normalization factor $H(\mathbf{AB} \mid \Psi)$ is not particularly important since it does not depend on the scheme, it is simply a baseline that is chosen to represent the average download needed from a centralized server that directly sends $\mathbf{AB}$ to the user in the absence of security constraints. Other baselines, e.g., $H(\mathbf{AB})$ may be chosen instead as in [18], or one could equivalently formulate the problem directly as a minimization of download cost D. We prefer the formulation as a rate maximization because it allows a more direct connection to the capacity of PIR, one of the main themes of this chapter.*

Finally, depending upon which matrices are secured and/or available as side-information, we have the following versions of the SDMM problem.

SDMM version	secure	side-information Ψ	capacity
SDMM$_{(\mathbf{AB},\phi)}$	$\mathbf{A}$,$\mathbf{B}$	ϕ	$C_{(\mathbf{AB},\phi)}$
SDMM$_{(\mathbf{AB},\mathbf{B})}$	$\mathbf{A}$,$\mathbf{B}$	$\mathbf{B}$	$C_{(\mathbf{AB},\mathbf{B})}$
SDMM$_{(\mathbf{B},\phi)}$	$\mathbf{B}$	ϕ	$C_{(\mathbf{B},\phi)}$
SDMM$_{(\mathbf{B},\mathbf{A})}$	$\mathbf{B}$	$\mathbf{A}$	$C_{(\mathbf{B},\mathbf{A})}$
SDMM$_{(\mathbf{B},\mathbf{B})}$	$\mathbf{B}$	$\mathbf{B}$	$C_{(\mathbf{B},\mathbf{B})}$

(5.11)

Thus, the version of SDMM is indicated by the subscript which has two elements, the first representing the matrices that are secured and the second representing the matrices available to the user as side-information. Note that other cases, such as SDMM$_{(\mathbf{AB},\mathbf{A})}$, SDMM$_{(\mathbf{A},\phi)}$, SDMM$_{(\mathbf{A},\mathbf{A})}$, SDMM$_{(\mathbf{A},\mathbf{B})}$, are equivalent to, SDMM$_{(\mathbf{AB},\mathbf{B})}$, SDMM$_{(\mathbf{B},\phi)}$, SDMM$_{(\mathbf{B},\mathbf{B})}$, SDMM$_{(\mathbf{B},\mathbf{A})}$, respectively, by the inherent symmetry of the problem, leaving us with just the 5 cases tabulated above.

5.3 Results

5.3.1 A Connection between SDMM and MM-XSTPIR

Let us begin by identifying a connection between SDMM and multi-message X-secure T-private information retrieval (MM-XSTPIR). We refer the reader to Appendix C.1 for a formal definition of MM-XSTPIR.

Lemma 5.1. *The following bounds apply.*

$$K \geq M \implies \max\big(C_{(\mathbf{AB},\mathbf{B})}, C_{(\mathbf{B},\mathbf{B})}, C_{(\mathbf{AB},\phi)}, C_{(\mathbf{B},\phi)}\big) \leq C_{\textit{MM-XSTPIR}}(N, X_A, X_B, K, M), \tag{5.12}$$

$$K \geq L \quad \Longrightarrow \qquad\qquad C_{(\mathbf{AB},\phi)} \leq C_{\textit{MM-XSTPIR}}(N,X_B,X_A,K,L), \tag{5.13}$$

where $C_{\textit{MM-XSTPIR}}(N,X,T,K,M)$ is the capacity of MM-XSTPIR with N servers, X-secure storage and T-private queries, retrieving M out of K messages.

Proof. Let us first prove the bound in (5.12), by showing that when $K \geq M$, then any SDMM scheme where the side-information available to the user is not[8] $\mathbf{A}$, and where $X_B \neq 0$, automatically yields an MM-XSTPIR(N,X_A,X_B,K,M) scheme with the same rate, essentially by thinking of $\mathbf{A}$ as the data and $\mathbf{B}$ as the query. Consider MM-XSTPIR with K independent messages, each of which consists of L i.i.d. uniform symbols in $\mathbb{F}_q$, say arranged in a column. For all $k \in [K]$, arrange these columns to form the matrix A_1 so that the k^{th} column of A_1 represents the k^{th} message. Let the M desired message indices be represented by the corresponding columns of the $K \times K$ identity matrix, and let these M columns be arranged to form the $K \times M$ matrix B_1. Note that retrieving the matrix product A_1B_1 is identical to retrieving the M desired messages. Now any (X_A,X_B) secure SDMM scheme with $S=1$ that does not have $\mathbf{A}$ as side-information, conditioned on the realizations $\mathbf{A}_1 = A_1, \mathbf{B}_1 = B_1$, yields an MM-XSTPIR scheme by treating $\widetilde{A}_1^n$ as the X_A-secure data stored at the n^{th} server and $\widetilde{B}_1^n$ as the X_B-private query sent by the user to the n^{th} server, for all $n \in [N]$. For arbitrary $S > 1$ we can simply extend the data by a factor of S, i.e., each message is comprised of SL symbols, so that the k^{th} message is represented by the k^{th} columns of $A_1, A_2, \cdots, A_S$, treated as realizations of $\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_S$. Thus, conditioning on the realization $\mathbf{B}_1 = \mathbf{B}_2 = \cdots = \mathbf{B}_S = B_1$ gives us the S-fold extension of the same scheme. Furthermore, since $\mathbf{B}$ matrices are secure, i.e., $X_B > 0$ for all SDMM settings that appear in (5.12), it follows that $\mathbf{B}$ is independent

[8] $\mathbf{A}$ cannot be the side-information because as noted, the transformation from SDMM to MM-XSTPIR interprets $\mathbf{A}$ as the data, which cannot be already available to the user in MM-XSTPIR. On the other hand, $\mathbf{B}$ may be included in the side-information because it is interpreted as the queries, which are automatically known to the user in MM-XSTPIR. Note that if $\mathbf{B}$ is also not included in the side-information, that just means that the user can retrieve $\mathbf{AB}$ from the downloads without using the knowledge of $\mathbf{B}$ in the resulting MM-XSTPIR scheme.

of $\widetilde{B}^n_{[S]}$ for any $n\in[N]$. This in turn implies that $\mathbf{B}$ is independent of the download Δ_n received from Server n. Therefore, conditioning on $\mathbf{B}$ taking values in the set that corresponds to MM-XSTPIR (i.e., B_1 restricted to any choice of M columns of the $K\times K$ identity matrix) does not affect the distribution of Δ_n, or the entropy $H(\Delta_n)$. In other words, the average download of the SDMM scheme remains unchanged as it is specialized to yield an MM-XSTPIR scheme as described above. Now, since the number of desired q-ary symbols retrieved by this feasible MM-XSTPIR scheme is SLM, the average download, D for the SDMM scheme cannot be less than $SLM/C_{\text{MM-XSTPIR}}(N,X_A,X_B,K,M)$. Therefore, we have a bound on the rate of the SDMM scheme as

$$\begin{aligned}
R &= \frac{H(\mathbf{AB}\,|\,\mathbf{B})}{D} && (5.14)\\
&\leq \frac{H(\mathbf{AB})}{D} && (5.15)\\
&\leq \frac{SLM}{D} && (5.16)\\
&\leq \frac{SLM}{SLM/C_{\text{MM-XSTPIR}}(N,X_A,X_B,K,M)} && (5.17)\\
&= C_{\text{MM-XSTPIR}}(N,X_A,X_B,K,M) && (5.18)
\end{aligned}$$

In (5.15) we used the fact that conditioning reduces entropy so that $H(\mathbf{AB}\,|\,\mathbf{B})\leq H(\mathbf{AB})$. In (5.16) we used the fact that the matrix $\mathbf{AB}$ has SLM elements from $\mathbb{F}_q$, and since the uniform distribution maximizes entropy, $H(\mathbf{AB})\leq SLM$. The bound in (5.13) is similarly shown, by treating $\widetilde{B}^n_1$ as the X_A-secure data stored at the n^{th} server and $\widetilde{A}^n_1$ as the X_B-private query sent by the user to the n^{th} server. This completes the proof of Lemma 5.1. □

5.3.2 An Upperbound on the Capacity of MM-XSTPIR

Motivated by Lemma 5.1, an upper bound on the capacity of MM-XSTPIR is presented in the following theorem.

Theorem 5.1. *The capacity of MM-XSTPIR is bounded as follows.*

$$C_{\textit{MM-XSTPIR}}(N,X,T,K,M) \leq \begin{cases} 0, & N \leq X, \\ \dfrac{M(N-X)}{KN}, & X < N \leq X+T, \\ \dfrac{N-X}{N}\left(1+\left(\dfrac{T}{N-X}\right)+\cdots+\left(\dfrac{T}{N-X}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right)^{-1}, & N > X+T. \end{cases} \tag{5.19}$$

The proof of Theorem 5.1 is presented in Appendix C.1.2. Note that Theorem 5.1 also works for trivial security or privacy, i.e., when $X=0$ or $T=0$.

Remark 5.2. *In fact a closer connection exists between SDMM and MM-XSTPC, i.e., multi-message X-secure T-private (linear) computation problem that is an extension of the private computation problem studied in [107]. However, we use only the connection to MM-XSTPIR because this setting is simpler and the connection between SDMM and MM-XSTPIR suffices for our purpose .*

5.3.3 Entropies of Products of Random Matrices

The following lemma is needed to evaluate the numerator in the rate expressions for SDMM schemes.

Lemma 5.2. *Let* $\mathbf{A}$, $\mathbf{B}$ *be random matrices independently and uniformly distributed over* $\mathbb{F}_q^{L\times K}$, $\mathbb{F}_q^{K\times M}$, *respectively. As* $q\to\infty$, *we have*

$$H(\mathbf{AB}) = \begin{cases} LM, & K \geq \min(L,M) \\ LK+KM-K^2, & K < \min(L,M) \end{cases}, \tag{5.20}$$

$$H(\mathbf{AB} \mid \mathbf{A}) = \min(LM,KM), \tag{5.21}$$

$$H(\mathbf{AB}\,|\,\mathbf{B}) = \min(LM, LK), \tag{5.22}$$

in q-ary units.

The proof of Lemma 5.2 appears in Appendix C.2.

We now proceed to capacity characterizations for the various SDMM models.

5.3.4 Capacity of SDMM$_{(\mathbf{AB},\phi)}$

Let us start with the basic SDMM setting, where both matrices $\mathbf{A}, \mathbf{B}$ are X-secured, and there is no prior side-information available to the user. This is essentially the two-sided secure SDMM setting considered previously in [18, 61].

Theorem 5.2. *The capacity of SDMM$_{(\mathbf{AB},\phi)}$, with $X_A = X_B = X$, is characterized under various settings as follows.*

$$\begin{aligned}
N \le X &\implies C_{(\mathbf{AB},\phi)} = 0 && (5.23)\\
N > X, K = 1 &\implies C_{(\mathbf{AB},\phi)} = 1 - \frac{X}{N} && (5.24)\\
2X \ge N > X, \frac{K}{\min(L,M)} \to \infty &\implies C_{(\mathbf{AB},\phi)} = 0 && (5.25)\\
N > 2X, \frac{K}{\min(L,M)} \to \infty &\implies C_{(\mathbf{AB},\phi)} = 1 - \frac{2X}{N} && (5.26)\\
N > X, K \le \min(L,M), \frac{\max(L,M)}{K} \to \infty &\implies C_{(\mathbf{AB},\phi)} = 1 - \frac{X}{N} && (5.27)\\
N > X, \frac{K}{\min(L,M)} < 1 &\implies C_{(\mathbf{AB},\phi)} \le 1 - \frac{X}{N} && (5.28)\\
2X \ge N > X, \frac{K}{\min(L,M)} \ge 1 &\implies C_{(\mathbf{AB},\phi)} \le \left(1 - \frac{X}{N}\right)\frac{\min(L,M)}{K} && (5.29)
\end{aligned}$$

$$N > 2X, \frac{K}{\min(L,M)} \ge 1 \implies$$

$$C_{(\mathbf{AB},\phi)} \leq \left(1-\frac{X}{N}\right)\left(1+\left(\frac{X}{N-X}\right)+\cdots+\left(\frac{X}{N-X}\right)^{\lfloor\frac{K}{\min(L,M)}\rfloor-1}\right)^{-1} \tag{5.30}$$

Case (5.23) is trivial because $\mathbf{A},\mathbf{B}$ are X-secure, and nothing is available to the user as side-information, which means that even if the user and the servers fully combine their knowledge, $\mathbf{A},\mathbf{B}$ remain a perfect secret. The converse proof for cases (5.24) and (5.27) of Theorem 5.2 is presented in Section 5.4.2. Converse proofs for all other cases follow from Lemma 5.1 and Theorem 5.1. For example, consider case (5.25). According to (5.12),(5.13), we have $C_{(\mathbf{AB},\phi)} \leq C_{\text{MM-XSTPIR}}(N,X,X,K,\min(L,M))$ which in turn is bounded by $\min(L,M)(N-X)/(KN)$ according to Theorem 5.1. Therefore, if $K/\min(L,M)\to\infty$, then we have the bound $C_{(\mathbf{AB},\phi)}=0$. Converse bounds for other cases are found similarly. The proof of achievability for case (5.24) is presented in Section 5.5.4. All other achievability results are presented in Section 5.5.3.

Remark 5.3. *The capacity of* 2*-sided SDMM problem is characterized in [61] as* $\left(1-\frac{2X}{N}\right)^+$. *Our capacity characterizations for cases* (5.24) *and* (5.27) *present a contradiction that calls into question*[9] *the converse bound in [61]. To further highlight the contradiction, note that in the SDMM*$_{\mathbf{AB},\phi}$ *problem, for arbitrary* L,K,M, *just by retrieving each of* $\mathbf{A},\mathbf{B}$ *separately using the scheme described in Section 5.5.1, it is possible to achieve a rate equal to* $\frac{H(\mathbf{AB})}{(LK+KM)}\left(1-\frac{X}{N}\right) \geq \frac{\min(LM,LK+KM-K^2)}{(LK+KM)}\left(1-\frac{X}{N}\right)$ *which can be larger than* $\left(1-\frac{2X}{N}\right)^+$. *Since [61] assumes* $S=1$, *consider for example,* $L=K=M$, *and* $N=X+1$. *Then with* $S=1$ *this simple scheme achieves a rate* $\frac{1}{2}\left(1-\frac{X}{N}\right)=\frac{1}{2N}$ *which exceeds* $\left(1-\frac{2X}{N}\right)^+=0$ *for all* $X>1$. *On the other hand, the achievable scheme presented in [61] does not*[10] *achieve the rate* $\left(1-\frac{2X}{N}\right)^+$ *when* $K<\min(L,M)$, *so it remains unknown if* $\left(1-\frac{2X}{N}\right)^+$ *is even a lower bound on capacity*

[9] The information provided by the genie to the user in the converse proof of [61] is subsequently considered useless on the basis that it is independent of $\mathbf{AB}$. However, it turns out this independent side-information can still be useful in decoding $\mathbf{AB}$, just as a noise term $\mathbf{Z}$ that is independent of $\mathbf{AB}$ can still be useful in decoding $\mathbf{AB}$ from the value $\mathbf{AB}+\mathbf{Z}$.

[10] This is because $H(\mathbf{AB})\neq LM$ when $K<\min(L,M)$. Instead, according to Lemma 5.2, $H(\mathbf{AB})=LK+KM-M^2$. So while the download for the scheme in [61] is $LMN/(N-2X)$, the rate achieved when $K<\min(L,M)$ is $H(\mathbf{AB})/D=\frac{LK+KM-K^2}{LM}\left(1-\frac{2X}{N}\right)^+$, which is strictly smaller than $\left(1-\frac{2X}{N}\right)^+$ for $K<\min(L,M)$.

in general.

Corollary 5.1. *Consider a modification of the SDMM$_{(\mathbf{AB},\phi)}$ problem, where instead of* $\mathbf{AB}$*, the user wants to retrieve the Hadamard product* $\mathbf{A}\circ\mathbf{B}$*. The capacity of this problem, i.e., the supremum of* $H(\mathbf{A}\circ\mathbf{B})/D$*, as* $q\to\infty$*, is* $1-X/N$*.*

The converse for Corollary 5.1 follows directly from the converse proof of Theorem 5.2, case (5.24) in Section 5.4.2 where we replace $\mathbf{AB}$ with $\mathbf{A}\circ\mathbf{B}$. On the other hand, since the Hadamard product is the entrywise product of matrices, thus the scalar multiplication scheme presented in Section 5.5.4 achieves the capacity.

5.3.5 Capacity of SDMM$_{(\mathbf{B},\mathbf{A})}$

The next SDMM model we consider corresponds to the one-sided SDMM problem considered in [18]. Recall that the one-sided security model in [18] assumes that one of the matrices is a constant matrix known to everyone. This corresponds to the $\mathbf{A}$ matrix in our model of SDMM$_{(\mathbf{B},\mathbf{A})}$ because $\mathbf{A}$ is not secured and is available to the user as side-information. Here our capacity result is consistent with [18].

Theorem 5.3. *The capacity of SDMM*$_{(\mathbf{B},\mathbf{A})}$ *with* $X_A=0, X_B=X$ *is*

$$C_{(\mathbf{B},\mathbf{A})}=\begin{cases}0, & N\leq X,\\ 1-\dfrac{X}{N}, & N>X.\end{cases} \tag{5.31}$$

Theorem 5.3 fully characterizes the capacity of SDMM$_{(\mathbf{B},\mathbf{A})}$. The converse for Theorem 5.3 follows along the same lines as the converse presented in [18], but for the sake of completeness we present the converse in Section 5.4.1. The proof of achievability provided in [18] is tight only[11] if $K \geq L$ and L is a multiple of $N-X$. Therefore, a complete proof of achievability is needed for Theorem 5.3. Such a proof is presented in Section 5.5.3.

5.3.6 Capacity of SDMM$_{(\mathbf{B},\mathbf{B})}$

Theorem 5.4. *The capacity of SDMM$_{(\mathbf{B},\mathbf{B})}$, with $X_A=0, X_B=X$, is characterized under various settings as follows.*

$$\begin{aligned} K &\leq M && \Longrightarrow C_{(\mathbf{B},\mathbf{B})} = 1 && (5.32)\\ K > M, N &\leq X && \Longrightarrow C_{(\mathbf{B},\mathbf{B})} = \frac{M}{K} && (5.33)\\ N > X, \frac{K}{M} &\to \infty && \Longrightarrow C_{(\mathbf{B},\mathbf{B})} = 1-\frac{X}{N} && (5.34)\\ K > M, N &> X && \Longrightarrow C_{(\mathbf{B},\mathbf{B})} \leq \left(1+\left(\frac{X}{N}\right)+\cdots+\left(\frac{X}{N}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right)^{-1} && (5.35) \end{aligned}$$

The converse for $K \leq M$ is trivial because the capacity by definition cannot exceed 1. The converse for the remaining cases follows directly from Lemma 5.1 and Theorem 5.1. For example, consider the case (5.33). According to Lemma 5.1, $\mathrm{C}_{\mathbf{B},\mathbf{B}} \leq C_{\text{MM-XSTPIR}}(N,0,X,K,M)$ which is bounded by M/K according to Theorem 5.1. Other cases follow similarly. The proof of achievability for Theorem 5.4 is provided in Section 5.5.3.

[11] The achievable scheme for the one-sided secure setting in [18] always downloads $NLM/(N-X)$ q-ary symbols, whereas the capacity achieving scheme needs to download only $NKM/(N-X)$ q-ary symbols when $K < L$.

5.3.7 Capacity of SDMM$_{(\mathbf{B},\phi)}$

Theorem 5.5. *The capacity of SDMM$_{(\mathbf{B},\phi)}$, with $X_A=0, X_B=X$, is characterized under various settings as follows.*

$$\begin{aligned}
N \le X \quad & \Longrightarrow C_{(\mathbf{B},\phi)} = 0 && (5.36)\\
K \ge L, N > X \quad & \Longrightarrow C_{(\mathbf{B},\phi)} = \left(1-\frac{X}{N}\right)\frac{H(\mathbf{AB})}{H(\mathbf{AB}\mid\mathbf{A})} = \left(1-\frac{X}{N}\right) && (5.37)\\
K < L, N > X, \frac{K}{M} \to \infty \quad & \Longrightarrow C_{(\mathbf{B},\phi)} = 1-\frac{X}{N} && (5.38)\\
K \le M, N > X, \frac{L}{M} \to \infty \quad & \Longrightarrow C_{(\mathbf{B},\phi)} = 1 && (5.39)\\
K < L, N > X, \frac{M}{L} \to \infty \quad & \Longrightarrow C_{(\mathbf{B},\phi)} = \left(1-\frac{X}{N}\right)\frac{H(\mathbf{AB})}{H(\mathbf{AB}\mid\mathbf{A})} = \left(1-\frac{X}{N}\right) && (5.40)\\
L > K \ge M, N > X \quad & \Longrightarrow C_{(\mathbf{B},\phi)} \le \left(1+\left(\frac{X}{N}\right)+\cdots+\left(\frac{X}{N}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right)^{-1} && (5.41)
\end{aligned}$$

The case $N \le X$ is trivial because $\mathbf{B}$ must be X-secure and there is no side-information at the user, which means that neither the user, nor all servers together have any knowledge of $\mathbf{B}$. The converse for (5.37) and (5.40) follows from the fact that any SDMM$_{\mathbf{B},\phi}$ scheme is also a valid SDMM$_{\mathbf{B},\mathbf{A}}$ scheme, so the download, say $D_{\mathbf{B},\phi}$ for the best SDMM$_{\mathbf{B},\phi}$ scheme cannot be less than the download, say $D_{\mathbf{B},\mathbf{A}}$ for the best SDMM$_{\mathbf{B},\mathbf{A}}$ scheme. For (5.38) the converse follows directly from Lemma 5.1 and Theorem 5.1. The converse for (5.39) is trivial because the capacity can never be more than 1 by definition. Finally, the converse for (5.41) also follows from Lemma 5.1 and Theorem 5.1. The achievability results for Theorem 5.5 are proved in Section 5.5.3.

5.3.8 Capacity of $\mathrm{SDMM}_{\mathbf{AB},\mathbf{B}}$

Theorem 5.6. *The capacity of SDMM$_{(\mathbf{AB},\mathbf{B})}$, with $X_A = X_B = X$, is characterized under various settings as follows.*

$$\begin{aligned}
N \le X \quad &\Longrightarrow C_{(\mathbf{AB},\mathbf{B})} = 0 && (5.42)\\
N > X, K \le M \quad &\Longrightarrow C_{(\mathbf{AB},\mathbf{B})} = 1 - \frac{X}{N} && (5.43)\\
2X \ge N > X, K > M \quad &\Longrightarrow C_{(\mathbf{AB},\mathbf{B})} = \frac{M(N-X)}{KN} && (5.44)\\
N > 2X, K > M, \frac{K}{M} \to \infty \quad &\Longrightarrow C_{(\mathbf{AB},\mathbf{B})} = 1 - \frac{2X}{N} && (5.45)\\
N > 2X, K > M \quad &\Longrightarrow
\end{aligned}$$

$$C_{(\mathbf{AB},\mathbf{B})} \le \frac{N-X}{N}\left(1 + \left(\frac{X}{N-X}\right) + \cdots + \left(\frac{X}{N-X}\right)^{\lfloor \frac{K}{M} \rfloor - 1}\right)^{-1} \tag{5.46}$$

The case (5.42) with $N \le X$ is trivial because the $\mathbf{A}$ is X-secure and not available to the user as side-information, which means that it is unknown to both the user and all servers. The converse for (5.43) follows from the observation that relaxing the security constraint for $\mathbf{B}$ cannot hurt, so $C_{(\mathbf{AB},\mathbf{B})}$ is bounded above by $C_{(\mathbf{A},\mathbf{B})}$, which is equal to $C_{(\mathbf{B},\mathbf{A})} = 1 - \frac{X}{N}$ by the symmetry of the problem and the result of Theorem 5.3. The converse for (5.44), (5.45), (5.46) follows from Lemma 5.1 and Theorem 5.1. The proof of achievability for Theorem 5.6 appears in Section 5.5.3.

5.4 Converse

5.4.1 Proof of Converse for Theorem 5.3

The case $N \leq X$ is trivial because $\mathbf{B}$ must be X-secure and not available to the user as side-information, which means that neither the user, nor all servers together have any knowledge of $\mathbf{B}$. Now let us consider the case $N > X$. Let $\mathcal{X}$ denote any subset of $[N]$ such that $|\mathcal{X}| = X$. We start with the following lemma.

Lemma 5.3. $I(\Delta_{\mathcal{X}}; \mathbf{AB} \mid \mathbf{A}) = 0$.

Proof.

$$
\begin{aligned}
& I(\Delta_{\mathcal{X}}; \mathbf{AB} \mid \mathbf{A}) \\
& = H(\Delta_{\mathcal{X}} \mid \mathbf{A}) - H(\Delta_{\mathcal{X}} \mid \mathbf{AB}, \mathbf{A}) && (5.47) \\
& \leq H(\Delta_{\mathcal{X}} \mid \mathbf{A}) - H(\Delta_{\mathcal{X}} \mid \mathbf{A}, \mathbf{B}) && (5.48) \\
& = I(\Delta_{\mathcal{X}}; \mathbf{B} \mid \mathbf{A}) && (5.49) \\
& \leq I\left(\widetilde{B}_{[S]}^{\mathcal{X}}, \mathbf{A}; \mathbf{B} \mid \mathbf{A}\right) && (5.50) \\
& \leq I\left(\widetilde{B}_{[S]}^{\mathcal{X}}, \mathbf{A}; \mathbf{B}\right) && (5.51) \\
& = I\left(\widetilde{B}_{[S]}^{\mathcal{X}}; \mathbf{B}\right) + I\left(\mathbf{A}; \mathbf{B} \mid \widetilde{B}_{[S]}^{\mathcal{X}}\right) && (5.52) \\
& \leq I\left(\widetilde{B}_{[S]}^{\mathcal{X}}; \mathbf{B}\right) + I\left(\mathbf{A}; \mathbf{B}, \widetilde{B}_{[S]}^{\mathcal{T}}\right) && (5.53) \\
& = 0. && (5.54)
\end{aligned}
$$

The steps in the proof are justified as follows. Step (5.47) applies the definition of mutual information. Step (5.48) follows from the fact that $(\mathbf{AB}, \mathbf{A})$ is function of $(\mathbf{A}, \mathbf{B})$ and conditioning reduces entropy. Step (5.49) applies the definition of mutual information, and (5.50) holds because $\Delta_{\mathcal{X}}$ is function of $\left(\widetilde{B}_{[S]}^{\mathcal{T}}, \mathbf{A}\right)$. In (5.51), (5.52) and (5.53), we repeatedly used the chain rule and non-negativity of mutual information. The last step follows from the security constraint defined in (5.6) and separate encoding of matrices (5.2). The proof is completed by the non-negativity of mutual information. □

The proof of converse of Theorem 5.3 is now presented as follows.

$$\begin{aligned}
&H(\mathbf{AB} \mid \mathbf{A}) \\
&= H(\mathbf{AB} \mid \mathbf{A}) - H\left(\mathbf{AB}|\Delta_{[N]}, \mathbf{A}\right) + H\left(\mathbf{AB} \mid \Delta_{[N]}, \mathbf{A}\right) && (5.55)\\
&= H(\mathbf{AB} \mid \mathbf{A}) - H\left(\mathbf{AB} \mid \Delta_{[N]}, \mathbf{A}\right) && (5.56)\\
&= I\left(\mathbf{AB}; \Delta_{[N]} \mid \mathbf{A}\right) && (5.57)\\
&= H(\Delta_{[N]} \mid \mathbf{A}) - H\left(\Delta_{[N]} \mid \mathbf{AB}, \mathbf{A}\right) && (5.58)\\
&\leq H(\Delta_{[N]} \mid \mathbf{A}) - H(\Delta_{\mathcal{X}} \mid \mathbf{AB}, \mathbf{A}) && (5.59)\\
&= H(\Delta_{[N]} \mid \mathbf{A}) - H(\Delta_{\mathcal{X}} \mid \mathbf{A}). && (5.60)
\end{aligned}$$

Steps are justified as follows. (5.55) subtracts and adds the same term so nothing changes. (5.56) follows from the correctness constraint,(5.9). Steps (5.57) and (5.58) follow from the definition of mutual information. In (5.59), we used the fact that dropping terms reduces entropy. The last step holds from Lemma 5.3.

Averaging (5.60) over all choices of $\mathcal{X}$ and applying Han's inequality (Theorem 17.6.1 in [26]), we have

$$\begin{aligned}
&H(\mathbf{AB} \mid \mathbf{A}) \\
&\leq H(\Delta_{[N]} \mid \mathbf{A}) - \frac{X}{N} H\left(\Delta_{[N]} \mid \mathbf{A}\right) && (5.61)\\
&= \left(1 - \frac{X}{N}\right) H(\Delta_{[N]}|\mathbf{A}) && (5.62)\\
&\leq \left(1 - \frac{X}{N}\right) H(\Delta_{[N]}) && (5.63)
\end{aligned}$$

$$\leq \left(1-\frac{X}{N}\right)\sum_{n\in[N]} H(\Delta_n). \tag{5.64}$$

Thus we obtain

$$C_{\mathbf{B},\mathbf{A}} = \sup \frac{H(\mathbf{AB}\,|\,\mathbf{A})}{D} \tag{5.65}$$

$$\leq \sup \frac{H(\mathbf{AB}\,|\,\mathbf{A})}{\sum_{n\in[N]} H(\Delta_n)} \tag{5.66}$$

$$\leq 1-\frac{X}{N}. \tag{5.67}$$

5.4.2 Converse of Theorem 5.2: (5.24),(5.27)

Let $\mathcal{X}$ denote any subset of $[N]$ such that $|\mathcal{X}| = X$. The proof of converse is as follows.

$$H(\mathbf{AB})$$

$$= H(\mathbf{AB}) - H\big(\mathbf{AB}\,|\,\Delta_{[N]}\big) + H\big(\mathbf{AB}\,|\,\Delta_{[N]}\big) \tag{5.68}$$

$$= H(\mathbf{AB}) - H\big(\mathbf{AB}\,|\,\Delta_{[N]}\big) \tag{5.69}$$

$$= I\big(\mathbf{AB};\Delta_{[N]}\big) \tag{5.70}$$

$$= H\big(\Delta_{[N]}\big) - H\big(\Delta_{[N]}\,|\,\mathbf{AB}\big) \tag{5.71}$$

$$\leq H(\Delta_{[N]}) - H\big(\Delta_{[N]}\,|\,\mathbf{A},\mathbf{B}\big) \tag{5.72}$$

$$\leq H(\Delta_{[N]}) - H(\Delta_{\mathcal{X}}\,|\,\mathbf{A},\mathbf{B}) \tag{5.73}$$

$$= H(\Delta_{[N]}) - H(\Delta_{\mathcal{X}}). \tag{5.74}$$

Steps are justified as follows. (5.68) subtracts and adds the same term so nothing changes. (5.69) follows from (5.9), while (5.70) and (5.71) follow from the definition of mutual information. (5.72) holds because adding conditioning reduces entropy and **AB** is function of (**A**,**B**). (5.73) holds because dropping terms reduces entropy. The last step simply follows from the following fact,

$$
\begin{aligned}
0 &= I(\mathbf{A},\mathbf{B};\widetilde{A}_{[S]}^{\mathcal{X}},\widetilde{B}_{[S]}^{\mathcal{X}}) && (5.75)\\
&= I(\mathbf{A},\mathbf{B};\Delta_{\mathcal{X}}) && (5.76)\\
&= H(\Delta_{\mathcal{X}}) - H(\Delta_{\mathcal{X}} \mid \mathbf{A},\mathbf{B}) && (5.77)
\end{aligned}
$$

where (5.75) is the security constraint defined in (5.7). (5.76) follows from non-negativity of mutual information and the fact that $\Delta_{\mathcal{X}}$ is function of $(\widetilde{A}_{[S]}^{\mathcal{X}},\widetilde{B}_{[S]}^{\mathcal{X}})$. (5.77) is the definition of mutual information.

Averaging (5.74) over all choices of $\mathcal{X}$ and applying Han's inequality, we have

$$
\begin{aligned}
H(\mathbf{AB}) &\leq H\left(\Delta_{[N]}\right) - \frac{X}{N} H\left(\Delta_{[N]}\right) && (5.78)\\
&= \left(1 - \frac{X}{N}\right) H\left(\Delta_{[N]}\right) && (5.79)\\
&\leq \left(1 - \frac{X}{N}\right) \sum_{n \in [N]} H(\Delta_n). && (5.80)
\end{aligned}
$$

Thus we obtain

$$
\begin{aligned}
C_{(\mathbf{AB},\phi)} &= \sup \frac{H(\mathbf{AB})}{D} && (5.81)\\
&\leq \sup \frac{H(\mathbf{AB})}{\sum_{n \in [N]} H(\Delta_n)} && (5.82)\\
&\leq 1 - \frac{X}{N}. && (5.83)
\end{aligned}
$$

5.5 Achievability

Let us present two basic schemes that are essential ingredients of the proofs of achievability.

5.5.1 A General Scheme

This scheme allows the user to retrieve all $\mathbf{A}$,$\mathbf{B}$, after which he can locally compute $\mathbf{AB}$. Let $S=N-X$, and let $\mathbf{Z}_{sx}, \mathbf{Z}'_{sx'}$, $s\in[S], x\in[X_A], x'\in[X_B]$ be uniformly distributed random matrices over $\mathbb{F}_q^{L\times K}$ and $\mathbb{F}_q^{K\times M}$ respectively. Note that $X_A, X_B \in \{0, X\}$. The independence of these random matrices and matrices $\mathbf{A}_{[S]}, \mathbf{B}_{[S]}$ is specified as follows.

$$
\begin{aligned}
&H\big((\mathbf{Z}_{sx}, \mathbf{Z}'_{sx'})_{s\in[S],x\in[X_A],x'\in[X_B]}, \mathbf{A}_{[S]}, \mathbf{B}_{[S]}\big)\\
=&\sum_{s\in[S],x\in[X_A],x'\in[X_B]} H(\mathbf{Z}_{sx})+H(\mathbf{Z}'_{sx'})+\sum_{s\in[S]} H(\mathbf{A}_s)+H(\mathbf{B}_s). \qquad (5.84)
\end{aligned}
$$

Let $\alpha_n, n\in[N]$ be N distinct elements from $\mathbb{F}_q$. The construction of securely encoded matrices $\widetilde{A}_s^n$ and $\widetilde{B}_s^n$ for any $s\in[S]$ and $n\in[N]$ is provided below.

$$
\begin{aligned}
\widetilde{A}_s^n &= \alpha_n^s \mathbf{A}_s + \sum_{x\in[X_A]} \alpha_n^{S+x} \mathbf{Z}_{xt} && (5.85)\\
&= \alpha_n^s \mathbf{A}_s + \alpha_n^{S+1} \mathbf{Z}_{s1} + \cdots + \alpha_n^{S+X_A} \mathbf{Z}_{sX_A} && (5.86)\\
\widetilde{B}_s^n &= \alpha_n^s \mathbf{B}_s + \sum_{x'\in[T_B]} \alpha_n^{S+x'} \mathbf{Z}'_{sx'} && (5.87)\\
&= \alpha_n^s \mathbf{B}_s + \alpha_n^{S+1} \mathbf{Z}'_{s1} + \cdots + \alpha_n^{S+X_B} \mathbf{Z}'_{sX_B}. && (5.88)
\end{aligned}
$$

The answer from the n-th server is specified as follows

$$
\begin{aligned}
\Delta_n &= \begin{bmatrix} \widetilde{A}_1^n + \cdots + \widetilde{A}_S^n \\ \widetilde{B}_1^n + \cdots + \widetilde{B}_S^n \end{bmatrix} && (5.89)\\
&= \begin{bmatrix} \alpha_n \mathbf{A}_1 + \cdots + \alpha_n^S \mathbf{A}_S + \alpha_n^{S+1} \sum_{s\in[S]} \mathbf{Z}_{s1} + \cdots + \alpha_n^{S+X_A} \sum_{s\in[S]} \mathbf{Z}_{sX_A} \\ \alpha_n \mathbf{B}_1 + \cdots + \alpha_n^S \mathbf{B}_S + \alpha_n^{S+1} \sum_{s\in[S]} \mathbf{Z}'_{s1} + \cdots + \alpha_n^{S+X_B} \sum_{s\in[S]} \mathbf{Z}'_{sX_B} \end{bmatrix}. && (5.90)
\end{aligned}
$$

Note that the desired matrices and the random matrices are coded with an RS code. Therefore, from the answers provided by all N servers, the user is able to decode all matrices

$\mathbf{A}_{[S]},\mathbf{B}_{[S]}$, and then determine $\mathbf{AB}=(\mathbf{A}_s\times\mathbf{B}_s)_{s\in[S]}$. Note that X_A-security is guaranteed for matrices $\mathbf{A}_s$ because they are protected by the X_A noise matrices $\mathbf{Z}_{sx},x\in[X_A]$, that are i.i.d. uniform and coded according to MDS(X_A,N). Similarly, X_B-security is guaranteed for matrices $\mathbf{B}_s$.

5.5.2 Cross Subspace Alignment Based Scheme

For this scheme, let us set

$$S=N-X_A-X_B. \tag{5.91}$$

And let $\mathbf{Z}_{sx},\mathbf{Z}'_{sx'}$, $s\in[1:S],x\in[T_A],x'\in[T_B]$ be uniformly distributed random matrices over $\mathbb{F}_q^{L\times K}$ and $\mathbb{F}_q^{K\times M}$ respectively. The independence of random matrices and matrices $\mathbf{A}_{[S]},\mathbf{B}_{[S]}$ is specified as follows.

$$\begin{aligned}&H\big((\mathbf{Z}_{sx},\mathbf{Z}'_{sx'})_{s\in[S],x\in[T_A],x'\in[T_B]},\mathbf{A}_{[S]},\mathbf{B}_{[S]}\big)\\&=\sum_{s\in[S],x\in[X_A]}H(\mathbf{Z}_{sx})+\sum_{s\in[S],x'\in[X_B]}H(\mathbf{Z}'_{sx'})+\sum_{s\in[S]}H(\mathbf{A}_s)+\sum_{s\in[S]}H(\mathbf{B}_s).\end{aligned} \tag{5.92}$$

For the construction of this scheme, we will need $N+S$ distinct constants $\alpha_n,n\in[N],f_s\in\mathbb{F}_q,s\in[S]$, that are elements of $\mathbb{F}$. The securely encoded matrix $\widetilde{A}_s^n$ for any $s\in[S]$ and $n\in[N]$ is provided below.

$$\widetilde{A}_s^n=\mathbf{A}_s+\sum_{x\in[X_A]}(f_s-\alpha_n)^x\mathbf{Z}_{sx} \tag{5.93}$$

$$=\mathbf{A}_s+(f_s-\alpha_n)\mathbf{Z}_{s1}+\cdots+(f_s-\alpha_n)^{X_A}\mathbf{Z}_{sX_A}. \tag{5.94}$$

Similarly, the securely encoded matrix $\widetilde{B}_s^n$ for any $s \in [S]$ and $n \in [N]$ is as follows,

$$\widetilde{B}_s^n = \frac{1}{f_s - \alpha_n}\left(\mathbf{B}_s + \sum_{x' \in [X_B]} (f_s - \alpha_n)^{x'} \mathbf{Z}'_{sx'}\right) \tag{5.95}$$

$$= \frac{1}{f_s - \alpha_n}\left(\mathbf{B}_s + (f_s - \alpha_n)\mathbf{Z}'_{s1} + \cdots + (f_s - \alpha_n)^{X_B}\mathbf{Z}'_{sX_B}\right). \tag{5.96}$$

The download from any server n, $n \in [N]$ is constructed as follows.

$$\Delta_n = \sum_{s \in [S]} \widetilde{A}_s^n \widetilde{B}_s^n \tag{5.97}$$

$$= \sum_{s \in [S]} \left(\frac{1}{f_s - \alpha_n}\mathbf{A}_s\mathbf{B}_s\right) + \sum_{s \in [S]}\sum_{x \in [X_A]} (f_s - \alpha_n)^{x-1}\mathbf{Z}_{sx}\mathbf{B}_s$$

$$+ \sum_{s \in [S]}\sum_{x' \in [X_B]} (f_s - \alpha_n)^{x'-1}\mathbf{A}_s\mathbf{Z}'_{sx'} + \sum_{s \in [S]}\sum_{x \in [X_A], x' \in [X_B]} (f_s - \alpha_n)^{x+x'-1}\mathbf{Z}_{sx}\mathbf{Z}'_{sx'}. \tag{5.98}$$

Each of the last three terms can be expanded into weighted sums of terms of the form α_n^t, $t \in \{0, 1, \ldots, X_A + X_B - 1\}$. Thus, upon receiving all N answers from servers, the user is able to decode all S desired product matrices $(\mathbf{A}_s \times \mathbf{B}_s)_{s \in [S]}$, as long as the following $N \times N$ matrix is invertible,

$$\mathbf{M}_N = \begin{bmatrix} \frac{1}{f_1 - \alpha_1} & \cdots & \frac{1}{f_S - \alpha_1} & 1 & \alpha_1 & \cdots & \alpha_1^{X_A + X_B - 1} \\ \frac{1}{f_1 - \alpha_2} & \cdots & \frac{1}{f_S - \alpha_2} & 1 & \alpha_2 & \cdots & \alpha_2^{X_A + X_B - 1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1 - \alpha_n} & \cdots & \frac{1}{f_S - \alpha_n} & 1 & \alpha_N & \cdots & \alpha_N^{X_A + X_B - 1} \end{bmatrix}, \tag{5.99}$$

which is shown to be true by Lemma 2.5. X_A-security is guaranteed for matrices $\mathbf{A}_s$ because they are protected by the X_A noise matrices $\mathbf{Z}_{sx}, x \in [X_A]$, that are i.i.d. uniform and coded according to MDS(X_A, N) codes. X_B-security is similarly guaranteed for the matrices $\mathbf{B}_s, s \in [S]$.

5.5.3 Proofs of Achievability

Throughout these proofs, we will allow $q \to \infty$. Furthermore, we will use Lemma 5.2 to calculate the entropy of random matrices.

Achievability Proof of Theorem 5.3

First, let us consider the setting when $K \geq L$. For this setting, let us apply the cross subspace alignment based scheme presented in Section 5.5.2. Note that $X_A = 0, X_B = X$, and the total number of downloaded q-ary symbols is NLM, so the rate achieved is

$$\begin{aligned} R &= \frac{H(\mathbf{AB} \mid \mathbf{A})}{NLM} && (5.100) \\ &= \frac{SLM}{NLM} && (5.101) \\ &= 1 - \frac{X}{N}, && (5.102) \end{aligned}$$

which matches the capacity for this setting. On the other hand, when $K < L$, let us apply the general scheme presented in Section 5.5.1. Since $X_A = 0, X_B = X$, we have

$$\Delta_n = \begin{bmatrix} \alpha_n \mathbf{A}_1 + \cdots + \alpha_n^S \mathbf{A}_S \\ \alpha_n \mathbf{B}_1 + \cdots + \alpha_n^S \mathbf{B}_S + \alpha_n^{S+1} \sum_{s \in [S]} \mathbf{Z}'_{s1} + \cdots + \alpha_n^{S+X} \sum_{s \in [S]} \mathbf{Z}'_{sX} \end{bmatrix}. \tag{5.103}$$

But note that left matrices $\mathbf{A}_{[S]}$ are already available to the user as side information, so it is not necessary to download $(\alpha_n \mathbf{A}_1 + \cdots - \alpha_n^S \mathbf{A}_S)$ terms. Therefore, the total number of downloaded q-ary symbols is NKM, and the rate achieved is

$$\begin{aligned} R &= \frac{H(\mathbf{AB} \mid \mathbf{A})}{NKM} && (5.104) \\ &= \frac{SKM}{NKM} && (5.105) \\ &= 1 - \frac{X}{N}, && (5.106) \end{aligned}$$

which matches the capacity for this setting. This completes the achievability proof of Theorem 5.3.

Achievability Proof for Theorem 5.4

First consider the trivial scheme with $S=1$ that downloads the matrices $\mathbf{A}_{[S]}$ directly from any one out of N servers, since there is no security constraint on these matrices ($X_A=0$). Since $\mathbf{B}_{[S]}$ is already available as side information, downloading $\mathbf{A}_{[S]}$ allows the user to compute $\mathbf{AB}$ locally. The rate achieved with this scheme is

$$\begin{aligned}
R=&\frac{H(\mathbf{AB}\,|\,\mathbf{B})}{LK} &(5.107)\\
=&\frac{\min(LM,LK)}{LK} &(5.108)\\
=&\begin{cases}1, & K\leq M,\\ \dfrac{M}{K}, & K>M.\end{cases} &(5.109)
\end{aligned}$$

Thus, this simple scheme is optimal for $K\leq M$ and for $(K>M,N\leq X)$.

Next let us consider $N>X$ as $K/M\to\infty$. For this, let us apply the cross subspace alignment based scheme presented in Section 5.5.2 with $S=N-X$. Note that $X_A=0,X_B=X$, and the total number of downloaded q-ary symbols is NLM, so the rate achieved is

$$\begin{aligned}
R=&\frac{H(\mathbf{AB}\,|\,\mathbf{B})}{NLM} &(5.110)\\
=&\frac{SLM}{NLM} &(5.111)\\
=&1-\frac{X}{N}, &(5.112)
\end{aligned}$$

which matches the capacity for this setting. This completes the achievability proof of Theorem 5.4.

Achievability Proof for Theorem 5.5

For the cases (5.37), (5.38), let us apply the cross subspace alignment based scheme presented in Section 5.5.2 with $S=N-X$. Since $X_A=0, X_B=X$, and the total number of downloaded q-ary symbol is NLM, the rate achieved is

$$\begin{aligned} R &= \frac{H(\mathbf{AB})}{NLM} && (5.113)\\ &= \frac{SLM}{NLM} && (5.114)\\ &= 1-\frac{X}{N}. && (5.115) \end{aligned}$$

This completes the achievability proof of Theorem 5.5 for the cases (5.37), (5.38).

Now consider the cases (5.39) and (5.40). For these cases, let us apply the general scheme presented in Section 5.5.1. Since $X_A=0, X_B=X$, we have

$$\Delta_n = \begin{bmatrix} \alpha_n\mathbf{A}_1+\cdots+\alpha_n^S\mathbf{A}_S \\ \alpha_n\mathbf{B}_1+\cdots+\alpha_n^S\mathbf{B}_S+\alpha_n^{S+1}\sum_{s\in[S]}\mathbf{Z}'_{s1}+\cdots+\alpha_n^{S+X}\sum_{s\in[S]}\mathbf{Z}'_{sX} \end{bmatrix}. \tag{5.116}$$

Note that from the downloads Δ_n of any S servers, we are able to recover the matrices $\mathbf{A}_{[S]}$, so we can eliminate the first part from the remaining $N-S$ redundant downloads while preserving decodability. Therefore, the total number of downloaded q-ary symbols is $SLK+NKM$. Thus, as $q\to\infty$, the rate achieved is

$$\begin{aligned} R &= \frac{H(\mathbf{AB})}{SLK+NKM} && (5.117)\\ &= \frac{S(LK+KM-K^2)}{SLK+NKM} && (5.118) \end{aligned}$$

As $L/M\to\infty$ and when $M\geq K$, we have $R=1$. This completes the proof of achievability of (5.39). On the other hand, when $M/L\to\infty$ and $K<L$, we have

$$\begin{aligned} R &= \frac{S(LK+KM-K^2)}{SLK+NKM} && (5.119)\\ &\overset{M/L\to\infty}{=} \frac{S}{N} && (5.120)\\ &= 1-\frac{X}{N} && (5.121)\end{aligned}$$

This proves achievability for (5.40), thus completing the proof of achievability for Theorem 5.5.

Achievability Proof of Theorem 5.6

Let us start with the cases (5.43) and (5.44), for which we apply the general scheme presented in Section 5.5.1. Since $X_A=X_B=X$, we have

$$\Delta_n=\begin{bmatrix}\alpha_n\mathbf{A}_1+\cdots+\alpha_n^S\mathbf{A}_S+\alpha_n^{S+1}\sum_{s\in[S]}\mathbf{Z}_{s1}+\cdots+\alpha_n^{S+X}\sum_{s\in[S]}\mathbf{Z}_{sX}\\ \alpha_n\mathbf{B}_1+\cdots+\alpha_n^S\mathbf{B}_S+\alpha_n^{S+1}\sum_{s\in[S]}\mathbf{Z}'_{s1}+\cdots+\alpha_n^{S+X}\sum_{s\in[S]}\mathbf{Z}'_{sX}\end{bmatrix}. \quad (5.122)$$

Now we note that since the matrices $\mathbf{B}_{[S]}$ are available to user as side information, it is not necessary to download the second part of Δ_n. Therefore, the total number of downloaded q-ary symbols is NLK, and the rate achieved is

$$\begin{aligned} R&=\frac{H(\mathbf{AB}\,|\,\mathbf{B})}{NLK} && (5.123)\\ &=\frac{S\min(LK,LM)}{NLK} && (5.124)\\ &=\begin{cases}1-\dfrac{X}{N}, & K\le M\\ \dfrac{M(N-X)}{KN}, & K>M\end{cases}. && (5.125)\end{aligned}$$

This completes the achievability proof for cases (5.43) and (5.44).

Next, let us consider case (5.45), and for this setting let us apply the cross subspace alignment based scheme presented in Section 5.5.2. Note that $X_A = X_B = X$, and the total number of downloaded q-ary symbols is NLM, so the rate achieved is

$$\begin{aligned} R &= \frac{H(\mathbf{AB} \mid \mathbf{B})}{NLM} && (5.126) \\ &= \frac{SLM}{NLM} && (5.127) \\ &= 1 - \frac{2X}{N} && (5.128) \end{aligned}$$

which matches the capacity for this setting.

Achievability Proof of Theorem 5.2

Let us start with case (5.26), for which we apply the cross subspace alignment based scheme that was presented in Section 5.5.2. Note that $X_A = X_B = X$, and the total number of downloaded q-ary symbols is NLM, so the rate achieved is

$$\begin{aligned} R &= \frac{H(\mathbf{AB})}{NLM} && (5.129) \\ &= \frac{SLM}{NLM} && (5.130) \\ &= 1 - \frac{2X}{N}, && (5.131) \end{aligned}$$

which matches the capacity for this setting.

Next, consider case (5.27). For this setting, let us apply the general scheme presented in

Section 5.5.1. Since $X_A = X_B = X$, we have

$$\Delta_n = \begin{bmatrix} \alpha_n \mathbf{A}_1 + \cdots + \alpha_n^S \mathbf{A}_S + \alpha_n^{S+1} \sum_{s\in[S]} \mathbf{Z}_{s1} + \cdots + \alpha_n^{S+X} \sum_{s\in[S]} \mathbf{Z}_{sX} \\ \alpha_n \mathbf{B}_1 + \cdots + \alpha_n^S \mathbf{B}_S + \alpha_n^{S+1} \sum_{s\in[S]} \mathbf{Z}'_{s1} + \cdots + \alpha_n^{S+X} \sum_{s\in[S]} \mathbf{Z}'_{sX} \end{bmatrix}. \tag{5.132}$$

Thus the total number of downloaded q-ary symbols is $N(LK+KM)$. Therefore, when $K \leq \min(L,M)$, we have

$$\begin{aligned} R &= \frac{H(\mathbf{AB})}{N(LK+KM)} && (5.133) \\ &= \frac{S(LK+KM-K^2)}{N(LK+KM)} && (5.134) \\ &\overset{\max(L,M)/K \to \infty}{=} 1 - \frac{X}{N}. && (5.135) \end{aligned}$$

This completes the achievability proof of case (5.27).

5.5.4 Achievability Proof of Theorem 5.2: Case (5.24)

$K = L = M = 1$

Let us first consider the setting where $K = L = M = 1$, and let us set $S = N - X$. Note that in this setting, $\mathbf{A}_s$, $\mathbf{B}_s$ are independent scalars drawn uniformly from the finite field $\mathbb{F}_q$. Let us first present a solution based on the assumption that $\mathbf{A}_s$, $\mathbf{B}_s$ take only non-zero values for all $s \in [S]$. It is well-known that the multiplicative group $\mathbb{F}_q^\times = \mathbb{F}_q \setminus \{0\}$ is a cyclic group. Moreover, every finite cyclic group of order $q-1$ is isomorphic to the additive group of $\mathbb{Z}/(q-1)\mathbb{Z}$ (i.e., addition modulo $(q-1)$). Therefore it is possible to translate scalar multiplication over $\mathbb{F}_q^\times$ into addition modulo $(q-1)$. However, the additive group of $\mathbb{Z}/(q-1)\mathbb{Z}$ is not a field, and our scheme will further require the properties of a field. This problem is circumvented by using a prime field $\mathbb{F}_p$ for a prime p such that $p>2(q-1)$ and noting that for any two integers $a,b\in\{0,1,\dots,q-2\}$, we have

$$(a+b)\mod(q-1)=((a+b)\mod p)\mod(q-1). \tag{5.136}$$

In other words, suppose the isomorphism between the multiplicative group $\mathbb{F}_q^\times$ and the additive group $\mathbb{Z}/(q-1)\mathbb{Z}$ maps all $a\in\mathbb{F}_q^\times$ to $f(a)\in\mathbb{Z}/(q-1)\mathbb{Z}$. Then for all $a,b,c\in\mathbb{F}_q^\times$ such that $c=a\times b$, we have $f(c)=f(a)+f(b)$ in $\mathbb{Z}/(q-1)\mathbb{Z}$, and furthermore, under the natural interpretation of all $f(a)$ as elements of $\mathbb{F}_p$, we have $c'=f(a)+f(b)$ in $\mathbb{F}_p$ such that $f(c)=c'$ mod $(q-1)$. Thus, we are able to transform the problem of scalar multiplication in $\mathbb{F}_q^\times$ to scalar addition over $\mathbb{F}_p$, i.e., instead of $c=a\times b\in\mathbb{F}_q^\times$, the user will retrieve $c'=f(a)+f(b)\in\mathbb{F}_p$ from which he can compute $f(c)$ by a mod $q-1$ operation, and then from $f(c)$ the user can compute c by inverting the isomorphic mapping.

To account for potential zero values of $\mathbf{A}_s,\mathbf{B}_s\in\mathbb{F}_q$, let us define $f(0)=0$. In light of this discussion, let us assume $f(\mathbf{A}_s),f(\mathbf{B}_s)$ are scalars in $\mathbb{F}_p$ and the user wishes to retrieve $f(\mathbf{A}_s)+f(\mathbf{B}_s)\in\mathbb{F}_p$ for all those $s\in[S]$ where $\mathbf{A}_s,\mathbf{B}_s$ are both non-zero, and he wishes to retrieve the answer 0 for all those $s\in[S]$ where either one of $\mathbf{A}_s,\mathbf{B}_s$ is zero. Now let us present a scheme to achieve this task. For this scheme let us choose p to be the minimum prime number such that $p>2(q-1)$. Let $\mathbf{Z}_{sx},\mathbf{Z}'_{sx}$, $s\in[S],x\in[X]$ be uniformly distributed random (noise) scalars over $\mathbb{F}_p$. The independence of these random scalars and the scalars $f(\mathbf{A}_s),f(\mathbf{B}_s)$ is specified as follows.

$$\begin{aligned}&H\big((\mathbf{Z}_{sx},\mathbf{Z}'_{sx})_{s\in[S],x\in[X]},(f(\mathbf{A}_s),f(\mathbf{B}_s))_{s\in[S]}\big)\\&=\sum_{s\in[S],x\in[X]}H(\mathbf{Z}_{sx})+\sum_{s\in[S],x\in[X]}H(\mathbf{Z}'_{sx})+\sum_{s\in[S]}H(f(\mathbf{A}_s))+\sum_{s\in[S]}H(f(\mathbf{B}_s)).\end{aligned} \tag{5.137}$$

Let $\alpha_n,n\in[N]$ be N distinct elements from $\mathbb{F}_p$. The construction of $\widetilde{A}_s^n$ and $\widetilde{B}_s^n$ for any

$s \in [S]$ and $n \in [N]$ is provided as follows.

$$\widetilde{A}_s^n = \alpha_n^s f(\mathbf{A}_s) + \sum_{x \in [X]} \alpha_n^{S+x} \mathbf{Z}_{sx} \tag{5.138}$$

$$= \alpha_n^s f(\mathbf{A}_s) + \alpha_n^{S+1} \mathbf{Z}_{s1} + \cdots + \alpha_n^{S+X} \mathbf{Z}_{sX} \tag{5.139}$$

$$\widetilde{B}_s^n = \alpha_n^s f(\mathbf{B}_s) + \sum_{x \in [X]} \alpha_n^{S+x} \mathbf{Z}'_{sx} \tag{5.140}$$

$$= \alpha_n^s f(\mathbf{B}_s) + \alpha_n^{S+1} \mathbf{Z}'_{s1} + \cdots + \alpha_n^{S+X} \mathbf{Z}'_{sX}. \tag{5.141}$$

The answer from the n-th server is obtained as follows

$$\Delta_n = \widetilde{A}_1^n + \cdots + \widetilde{A}_S^n + \widetilde{B}_1^n + \cdots + \widetilde{B}_S^n \tag{5.142}$$

$$\begin{aligned} &= \alpha_n (f(\mathbf{A}_1) + f(\mathbf{B}_1)) + \cdots + \alpha_n^S (f(\mathbf{A}_S) + f(\mathbf{B}_S)) \\ &\quad + \alpha_n^{S+1} \sum_{s \in [S]} (\mathbf{Z}_{s1} + \mathbf{Z}'_{s1}) + \cdots + \alpha_n^{S+X} \sum_{s \in [S]} (\mathbf{Z}_{sX} + \mathbf{Z}'_{sX}). \end{aligned} \tag{5.143}$$

X-security is guaranteed because matrices $\mathbf{A}_s, \mathbf{B}_s$ are protected by noise terms that are i.i.d. uniform and coded according to MDS(X,N) codes. Note that desired scalars and random scalars are coded with an RS code, and $S+X=N$. Therefore, from the answers provided by all N servers, the user is able to decode $(\mathbf{A}_s + \mathbf{B}_s)_{s \in [S]}$. By the transformation argument, correctness is guaranteed for all those $s \in [S]$, where $\mathbf{A}_s \neq 0$ and $\mathbf{B}_s \neq 0$. However, correctness is not yet guaranteed for those $s \in [S]$ where either $\mathbf{A}_s = 0$ or $\mathbf{B}_s = 0$. For this we will implement a separate mechanism to let the user know which $\mathbf{A}_s$ and $\mathbf{B}_s$ are equal to zero, so he can infer correctly that $\mathbf{A}_s \mathbf{B}_s = 0$ for those instances. Specifically, for each scalar $\mathbf{A}_s$ and $\mathbf{B}_s$, let us define binary symbols $\eta_{\mathbf{A}_s}, \eta_{\mathbf{B}_s}$ that indicate whether or not $\mathbf{A}_s, \mathbf{B}_s$ are equal to zero. These $\eta_{\mathbf{A}_s}, \eta_{\mathbf{B}_s}$ are also secret-shared among the N servers in an X-secure fashion, and retrieved by the user at negligible increase in download cost as $q \rightarrow \infty$. Now, by the Bertrand–Chebyshev theorem, for every integer $\nu > 1$ there is always at least one prime p' such that $\nu < p' < 2\nu$, thus we must have p such that $2(q-1) < p < 4(q-1)$. Therefore, as

$q \to \infty$, the rate achieved is

$$
\begin{aligned}
R &= \frac{H(\mathbf{A}\times\mathbf{B})}{D} && (5.144)\\
&\geq \frac{H(\mathbf{AB}\,|\,\mathbf{B})}{N\log_q(p)+2SN\log_q(2)} && (5.145)\\
&\geq \frac{S(\frac{q-1}{q})}{N\log_q(4(q-1))+2SN\log_q(2)} && (5.146)\\
&= \frac{(N-X)(\frac{q-1}{q})}{N\log_q(4(q-1))+2(N-X)N\log_q(2)} && (5.147)
\end{aligned}
$$

which approaches $1-X/N$ as $q \to \infty$.

$K=1$, **Arbitrary** L,M

Now let us consider the setting with arbitrary L,M, and with $K=1$, i.e., the user wishes to compute the outer product of vectors $\mathbf{A}_s,\mathbf{B}_s$ for all $s\in[S]$. As before, let us set $S=N-X$, and choose p to be the smallest prime such that $p>2(q-1)$. We will allow the user to download a normalized version of each $\mathbf{A}_s$ and $\mathbf{B}_s$ vector, along with the product of the normalizing factors, from which the user can construct $\mathbf{A}_s\mathbf{B}_s$. To this end, let us define i_s,j_s as the index of the first non-zero element in $\mathbf{A}_s,\mathbf{B}_s$, respectively, and normalize each vector $\mathbf{A}_s$ by it's i_s^{th} element $A_s(i_s)$, each vector $\mathbf{B}_s$ by it's j_s^{th} element $B_s(j_s)$ $\forall s\in[S]$. Now $\forall s\in[S]$ such that $\mathbf{A}_s$ is not the zero vector, we have

$$
\mathbf{A}_s = A_s(i_s)\underbrace{[0,\cdots,0,1,A_s'(i_s+1),\ldots,A_s'(L)]'}_{\mathbf{A}_s'}, \tag{5.148}
$$

where the vector $\mathbf{A}_s'$ is the normalized vector. Similarly, $\forall s\in[S]$ such that $\mathbf{B}_s$ is not the zero vector, we have

$$
\mathbf{B}_s = B_s(j_s)\underbrace{[0,\cdots,0,1,B_s'(j_s+1),\ldots,B_s'(M)]}_{\mathbf{B}_s'}. \tag{5.149}
$$

Note that if $\mathbf{A}_s$ or $\mathbf{B}_s$ is the zero vector, then we simply set $i_s=0, j_s=0$ and $\mathbf{A}'_s=0, \mathbf{B}'_s=0$, and $A_s(i_s)=1, B_s(j_s)=1$, respectively. The scheme is constructed as follows. Separate X-secure secret sharing schemes are used to distribute the secrets $i_s, j_s, A_s(i_s), B_s(j_s), \bar{\mathbf{A}}'_s, \bar{\mathbf{B}}'_s$, among the N servers, where $\bar{\mathbf{A}}'_s, \bar{\mathbf{B}}'_s$ are length $L-1, M-1$ vectors respectively, obtained by eliminating the leading 1 term from each of $\mathbf{A}'_s, \mathbf{B}'_s$ (or one of the zeros if $\mathbf{A}'_s, \mathbf{B}'_s$ are zero vectors). The vectors $\bar{\mathbf{A}}'_s, \bar{\mathbf{B}}'_s$ are retrieved by the user according to the scheme presented in Section 5.5.1, with total download cost equal to $N(L-1)+N(M-1)=N(L+M-2)$ q-ary symbols. The indices i_s, j_s are retrieved according to same scheme presented in Section 5.5.1, with total download cost not exceeding $N\log_q(L+1)+N\log_q(M+1)$ in units of q-ary symbols because the alphabet size for the indices i_s, j_s is $L+1, M+1$, respectively. The scheme presented in Section 5.5.4 is utilized by the user to retrieve the scalar products $A_s(i_s)B_s(j_s)$ with total download from N servers not exceeding $N\log_q(4(q-1))$. Note that since $A_s(i_s)B_s(j_s)$ is always non-zero there is no need for downloading additional indicators needed to identify zero values. The correctness follows from the fact that

$$\mathbf{A}_s \times \mathbf{B}_s = (A_s(i_s)B_s(j_s))\mathbf{A}'_s \times \mathbf{B}'_s, \tag{5.150}$$

and by the construction of the scheme, $(A_s(i_s)B_s(j_s))$ and $\mathbf{A}'_s \times \mathbf{B}'_s$ are recoverable for all $s \in [S]$. Therefore, the rate achieved is

$$\begin{aligned} R &= \frac{H\left((\mathbf{A}_s \times \mathbf{B}_s)_{s\in[S]}\right)}{D} && (5.151)\\ &\geq \frac{H\left((\mathbf{A}_s \times \mathbf{B}_s)_{s\in[S]}\right)}{N(L+M-2)+N\log_q(L+1)+N\log_q(M+1)+N\log_q(4(q-1))} && (5.152)\\ &= \frac{S(L+M-1)}{N(L+M-2)+N\log_q(L+1)+N\log_q(M+1)+N\log_q(4(q-1))} && (5.153)\\ &= \frac{(N-X)(L+M-1)}{N(L+M-2)+N\log_q(L+1)+N\log_q(M+1)+N\log_q(4(q-1))} && (5.154) \end{aligned}$$

which approaches $1-\frac{X}{N}$ as $q\to\infty$. This proves achievability of case (5.24), and completes the achievability proof of Theorem 5.2.

5.6 Discussion

A class of Secure Distributed Matrix Multiplication (SDMM) problems was defined in this chapter, and its capacity characterized in various parameter regimes depending on the security level X, number of servers N, matrix dimensions L,M,K and the set of matrices that are secured or available to the users as side-information. Notable aspects include connections between SDMM and a form of PIR known as MM-XSTPIR that led us to various converse bounds, and cross-subspace alignment schemes along with monomorphic transformations from scalar multiplication to scalar addition that formed the basis of some of the achievable schemes. Note that most of the achievable schemes in this chapter can also be adapted to the one-shot matrix multiplication framework of [61, 18], say where L,K,M all approach infinity and the ratios L/K, K/M are fixed constants. The converse parts follow directly, for the achievability parts, we can adapt our schemes to one-shot matrix multiplication based on matrix partitioning and zero-padding. For example, consider the *cross subspace alignment* based scheme in Section 5.5.2. Let us define $L_1 = \lfloor L/S' \rfloor S'$, where $S' = N - X_A - X_B$. Now let us partition matrix $\mathbf{A}$ as follows.

$$\mathbf{A} = [\mathbf{A}_{1,1} \quad \dots \quad \mathbf{A}_{1,S'} \quad \mathbf{A}_{2,1} \quad \dots \quad \mathbf{A}_{2,(L \mod S')}]^T, \tag{5.155}$$

where $(\mathbf{A}_{1,s'})_{s\in[S']}$ are $L_1/S' \times K$ matrices, $(\mathbf{A}_{2,s'})_{s\in[L \mod S']}$ are $1\times K$ vectors. Now with the schemes presented in this chapter, for all $s' \in [S']$, by setting $\mathbf{A}_{s'} = \mathbf{A}_{1,s'}$ and $\mathbf{B}_{s'} = \mathbf{B}$, one can recover $(\mathbf{A}_{1,s'}\mathbf{B})_{s\in[S']}$. On the other hand, we can then set $\mathbf{A}_{s'} = \mathbf{A}_{2,s'}, s' \in [L \mod S']$, $\mathbf{A}_{s'} = \mathbf{0}, s \in \{(L \mod S')+1,\dots,S'\}$ and also $\mathbf{B}_{s'} = \mathbf{B}, s \in [S']$ to recover $(\mathbf{A}_{2,s'}\mathbf{B})_{s\in[K \mod S']}$. Note that the extra cost of zero-padding is upper bounded by $KS'/LK = S'/L$, which goes to zero as $L \to \infty$. Thus the desired rates are still achievable.

Chapter 6

Cross Subspace Alignment Codes for Coded Distributed Batch Computation

The goal of coded distributed computation is to efficiently distribute a computation task, such as matrix multiplication, N-linear computation, or multivariate polynomial evaluation, across S servers through a coding scheme, such that the response from any R servers (R is called the recovery threshold) is sufficient for the user to recover the desired computed value. Current state-of-art approaches are based on either exclusively matrix-partitioning (Entangled Polynomial (EP) Codes for matrix multiplication), or exclusively batch processing (Lagrange Coded Computing (LCC) for N-linear computations or multivariate polynomial evaluations). We present three related classes of codes, based on the idea of Cross-Subspace Alignment (CSA) which was introduced originally in the context of secure and private information retrieval. CSA codes are characterized by a Cauchy-Vandermonde matrix structure that facilitates interference alignment along Vandermonde terms, while the desired computations remain resolvable along the Cauchy terms. These codes are shown to unify, generalize and improve upon the state-of-art codes for distributed computing. First we introduce CSA codes for matrix multiplication, which yield LCC codes as a special case, and are shown to outperform LCC codes in general in download-limited settings. While matrix-partitioning approaches (EP codes) for distributed matrix multiplication have the advantage of flexible server computation latency, batch processing approaches (CSA, LCC) have significant advantages in communication costs as well as encoding and decoding complexity per matrix multiplication. In order to combine the benefits of these approaches, we introduce Generalized CSA (GCSA) codes for matrix multiplication that bridge the extremes of matrix-partitioning and batch processing approaches and demonstrate synergistic gains due to cross subspace alignment. Finally, we introduce N-CSA codes for N-linear distributed batch computations and multivariate batch polynomial evaluations. N-CSA codes include LCC codes as a special case, and are in general capable of outperforming LCC codes in download-constrained settings by upto a factor of N. Generalizations of N-CSA codes to include X-secure data and B-byzantine servers are also provided.

6.1 Introduction

In the era of big data and cloud computing along with massive parallelization, there is particular interest in algorithms for coded distributed computation that are resilient to stragglers [140, 31, 28, 141, 138, 91, 69, 68, 29, 30, 139, 48, 6, 100, 122, 77, 121, 94, 44, 96, 49, 65, 86, 72]. The goal in coded distributed computation is to distribute the computation task according to a coding scheme across S servers (also known as workers or processors), such that the response from any R servers is sufficient for the user to recover (decode) the result of the computation. The parameter R is called the recovery threshold. Coded distributed computing offers the advantage of reduced latency from massive parallelization, because the tasks assigned to each server are smaller, and the redundancy added by coding helps avoid bottlenecks due to stragglers. The main metrics of interest for coded distributed computation include: the encoding and decoding complexity, latency[1] and complexity of server computation, the recovery threshold, and the upload and download costs (communication costs). With high end communication speeds approaching Gbps and computing speeds (processor clock speeds) commonly of the order of GHz, communication and computation costs may be comparable for many applications, allowing meaningful tradeoffs between the two. On the other hand, since communication bottlenecks are quite common, communication costs remain a key concern in distributed computing. Note that even with higher communication costs distributed computing may be necessary if, e.g., the computation task is too large to be efficiently carried out locally, or if the sources that generate the inputs for computation are not the same as the destination where the output of computation is desired, i.e., communication is unavoidable. Figure 6.1 shows such a setting for coded distributed batch matrix multiplication (CDBMM). Another notable aspect of such settings is that the cost dynamics for uploads and downloads may be different, e.g., if the input data is relatively static and multiple users request computations on different parts of the same dataset, then the download cost may be much more of a concern than upload cost. This will be significant when we compare different coding schemes in this chapter.

Distributed coded computing can be applied to a myriad of computational tasks. Of particular interest to this chapter are matrix multiplications, N-linear computations (e.g., computing the determinants of $N \times N$ matrices, or the product of N matrices), and evaluations of multivariate polynomials. These are some of the most fundamental building blocks of computation. Moreover, these problems are closely related. Indeed matrix multiplications are bilinear operations, so they are special cases of multilinear computations, and multilinear computations may be seen as special cases of multivariate polynomial evaluations. Several elegant coding schemes, or codes, have been proposed for solving these problems. Codes for distributed matrix multiplication evolved through MDS codes [68], Polynomial codes [140],

[1] Latency is the time it takes a server to complete a specific computation job. Unlike server computation complexity, it is not normalized by the size of the job, so it depends on the size of the job assigned to the server. Latency constraints are explored in the discussion following Theorem 6.2 in Section 6.5.

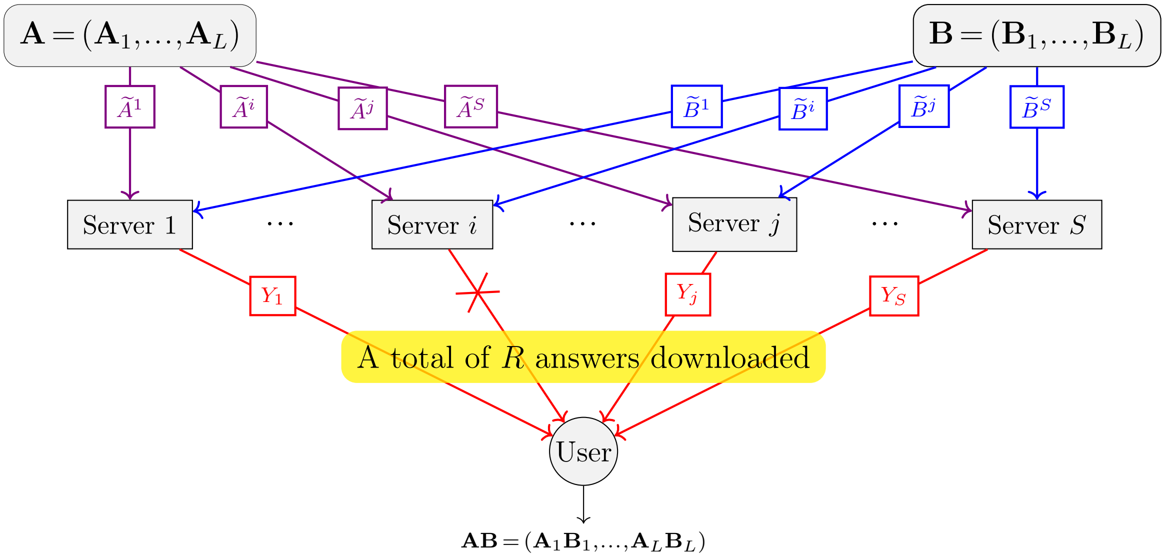


Figure 6.1: The CDBMM problem. Source (master) nodes generate matrices $\mathbf{A}=(\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_L)$ and $\mathbf{B}=(\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_L)$, and upload them to S distributed servers in coded form $\widetilde{A}^{[s]}$, $\widetilde{B}^{[s]}$, respectively. For all $l\in[L]$, $\mathbf{A}_l$ and $\mathbf{B}_l$ are $\lambda\times\kappa$ and $\kappa\times\mu$ matrices, respectively, over a field $\mathbb{F}$. The s^{th} server computes the answer Y_s, which is a function of all information available to it, i.e., $\widetilde{A}^s$ and $\widetilde{B}^s$. For effective straggler (e.g., Server i in the figure) mitigation, upon downloading answers from any R servers, where $R<S$, the user must be able to recover the product $\mathbf{AB}=(\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\ldots,\mathbf{A}_L\mathbf{B}_L)$.

MatDot and PolyDot codes [31] to the current state of art reflected in Generalized PolyDot codes [28] and Entangled Polynomial (EP) codes [141]. For multilinear computations and evaluations of multivariate polynomials, the state of art is represented by Lagrange Coded Computing (LCC), introduced in [138].

It is interesting to note that the solutions to these problems fall into two distinct categories — those based on partitioning of a single computation task [140, 31, 28, 141], and those based on batch processing of multiple computation tasks [138]. For example, consider the CDBMM problem shown in Figure 6.1 where the goal is to efficiently multiply L instances of $\lambda\times\kappa$ matrices, $\mathbf{A}=(\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_L)$, with L instances of $\kappa\times\mu$ matrices $\mathbf{B}=(\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_L)$, to compute the batch of L matrix products, $\mathbf{AB}=(\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\cdots,\mathbf{A}_L\mathbf{B}_L)$. Matrix-partitioning approaches compute each of the L products $\mathbf{A}_l\mathbf{B}_l$ one at a time by partitioning individual matrices $\mathbf{A}_l$ and $\mathbf{B}_l$ and coding across these partitions. Batch processing approaches do not partition individual matrices, instead they code across the batch of $\mathbf{A}$ matrices and across the batch of $\mathbf{B}$ matrices. The state-of-art for matrix-partitioning approaches is represented by Entangled Polynomial Codes (EP codes) [141], while Lagrange Coded Computing (LCC) [138] represents the state of art for batch processing. Since the problems are related, it is natural to ask, how do the matrix-partitioning solutions compare with the batch-processing solutions? Furthermore, can these solutions be improved, unified, generalized? These are the questions that we address in this chapter.

The essential ingredient in this chapter that allows us to compare, improve, unify and generalize the solutions to matrix multiplication, multilinear computation and multivariate polynomial evaluation, turns out to be the idea of cross-subspace alignment. Cross-subspace alignment (CSA) was originally introduced in the context of X-Secure T-Private Information Retrieval (XSTPIR) in Chapter 2. Coding schemes that exploit CSA have been used to improve upon and generalize the best known schemes for PIR with X-secure data, T-private queries and various forms of storage, e.g., fully replicated (Chapter 2, [56]), graph based replicated storage with limited replication factor (Chapter 3, [53]), or MDS coded storage (Chapter 4, [54]). CSA schemes have also recently been shown to be useful to minimize download communication cost for secure and/or private matrix multiplication (Chapter 5, [61, 60, 51, 54]). Building upon these efforts, in this chapter we introduce a new and generalized class of coded distributed computation codes, called CSA codes, that are inspired by the idea of cross-subspace alignment. The contributions of this chapter are summarized as follows.

1. **CSA Codes.** In Theorem 6.1 of this chapter that appears in Section 6.4, we introduce CSA codes for coded distributed batch matrix multiplication. These codes are used to multiply a batch of matrices $\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_L$ with $\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_L$ to recover the L desired matrix products $\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\cdots,\mathbf{A}_L\mathbf{B}_L$. There is no partitioning of individual matrices. Instead, coding is done across the matrices within a batch. CSA codes

partition a batch of L matrices into ℓ sub-batches of K_c matrices each ($L = \ell K_c$). Due to cross-subspace alignment, the interference is limited to $K_c - 1$ dimensions regardless of the number of sub-batches ℓ, so that the recovery threshold $R = L + K_c - 1$. The download per server does not depend on ℓ, although the upload and server computation complexity do scale with ℓ. Surprisingly, setting $\ell = 1$ recovers the Lagrange Coded Computing (LCC) solution to coded distributed batch matrix multiplication as a special case of CSA codes. Besides the additional flexibility, the main advantage of choosing $\ell > 1$ in CSA codes is to reduce the download cost relative to LCC codes (see Fig. 6.4 in Section 6.5.1). This advantage is especially significant in settings where the download cost is the primary bottleneck.

2. **EP vs CSA Codes.** We compare matrix partitioning approaches (say EP codes that generalize MatDot and Polynomial codes) with batch processing approaches (CSA codes that generalize Lagrange Coded Computing) for distributed matrix multiplication[2] (see Fig. 6.2 in Section 6.4.2). Remarkably, we find that batch processing presents a significant advantage in communication cost per matrix multiplication (i.e., normalized by the batch size L). As a function of the recovery threshold R, and for any fixed recovery ratio R/S, CSA codes have the same server computation complexity per matrix multiplication as EP codes, but CSA codes simultaneously achieve normalized (upload cost, download cost)=($\mathcal{O}(1)$,$\mathcal{O}(1)$), overcoming a key barrier of existing matrix-partitioning codes where upload cost of $\mathcal{O}(1)$ can only be achieved with download cost of $\mathcal{O}(R)$ and download cost of $\mathcal{O}(1)$ can only be achieved with upload cost of $\mathcal{O}(\sqrt{R})$. A corresponding improvement in the tradeoff between encoding and decoding

[2] Similar to Observation 2 in Section 6.4.2, let us reiterate that in this chapter we do not explore improvements that are possible by using more efficient matrix multiplication algorithms. This is in part because, as noted previously for $\lambda = \mu = \kappa$, optimally efficient matrix multiplication algorithms [67] represent a research avenue that is largely open, and the strongest advances in this direction are considered far from practical [45] due to large hidden constants in the $\mathcal{O}$ notation. For the sake of fairness, in this chapter we consider only coded distributed matrix multiplication schemes that are also built upon straightforward algorithms of matrix multiplication, e.g., the elementary version of EP codes. More elaborate matrix multiplication algorithms, e.g., Strassen's algorithm [99], can indeed be used for further improvements. See the discussion in Section 6.5.2.

complexity is also observed.

3. **GCSA Codes.** Since there is no partitioning of individual matrices in the aforementioned CSA codes, this means that each server must carry out a computational load equivalent to at least one full matrix multiplication before it can respond with an answer. This presents a latency barrier for batch processing schemes that cannot be overcome regardless of the number of servers and the batch size. For applications with stricter latency requirements such a solution may be infeasible, making it necessary to reduce the computational load per server by further parallelization, i.e., partitioning of individual matrices. To this end, in Theorem 6.2 that appears in Section 6.5.1 of this chapter, we present Generalized CSA codes (GCSA codes in short) that combine the matrix partitioning approach of, say EP codes, with the batch processing of CSA codes. GCSA codes bridge the two extremes by efficiently combining both matrix-partitioning and batch processing, and offer flexibility in how much of each approach is used. Both EP codes and LCC codes can be recovered as special cases of GCSA codes, but GCSA codes are capable of outperforming both EP and LCC codes in general (see Fig. 6.3 and Fig. 6.4 in Section 6.5.1). When no matrix partitioning is used, GCSA codes reduce to CSA codes, and if no batch processing is used then GCSA codes reduce to EP codes. With GCSA codes, the degree of matrix partitioning controls the server latency by limiting the computational load per server, while the batch partitioning on top yields the advantage of batch processing in communication costs. The combination is far from trivial. For example, consider a matrix partitioning approach that splits the task among 10 servers such that any $R_1 = 7$ need to respond, and a similar batch processing approach that also splits the task among 10 servers such that any $R_2 = 7$ need to respond. Then if we simply take the 10 matrix-partitioned tasks and use batch processing on top to distribute each task among 10 servers, for a total of 100 servers, then the recovery threshold of the naive combination is $6 \times 10 + 4 \times 6 + 1 = 85$. However, GCSA codes achieve a significantly lower recovery threshold ($R \leq R_1 R_2 = 49$).

4. ***N*-CSA Codes.** As noted, CSA codes are a generalization of LCC codes for distributed batch matrix multiplication. However, the applications of LCC codes extend beyond matrix multiplication, to N-linear batch computation and multivariate polynomial batch evaluations, raising the question whether corresponding generalizations of LCC codes to CSA type codes exist for these applications as well. We answer this question in the affirmative, by introducing N-CSA codes for the problem of coded distributed N-linear batch computation as well as multivariate polynomial evaluations, that are strictly generalizations of LCC codes for both of these applications. This generalization for batch size $L=\ell K_c$ is done as follows. For all $n\in[N]$, the batch of L realizations of the n^{th} variable is split into ℓ sub-batches, each containing K_c realizations. The K_c realizations within each sub-batch are coded into an MDS (S,K_c) code according to a Cauchy structure, and distributed to the S servers. Each server evaluates the N-linear map function with the coded variables of each sub-batch, and returns a weighted sum of evaluations of these ℓ sub-batches. By cross-subspace alignment, undesired evaluations only occupy $(N-1)(K_c-1)$ dimensions, so that the recovery threshold is $R=L+(N-1)(K_c-1)$. Finally, because N-linear maps are fundamental construction blocks of multivariate polynomials of total degree N, it is straightforward to apply N-CSA codes for multivariate polynomial batch evaluation. Specifically, we can regard any multivariate polynomial of total degree N as a linear combination of various restricted evaluations of N-linear maps. Each server prepares answers for various N-linear maps that constitute the given multivariate polynomial, then returns the user with the linear combination of these answers according to the given polynomial. Once again, the N-CSA code based scheme for multivariate polynomial batch evaluation thus obtained, generalizes LCC codes, which can be recovered by setting $\ell=1$. The main advantage of choosing $\ell>1$ with CSA codes remains the download cost. N-CSA codes achieve normalized download cost $D=\frac{R}{L}=1+\left(\frac{N-1}{\ell}\right)\left(\frac{K_c-1}{K_c}\right)$. The special case of $\ell=1$ which gives us LCC codes corresponds to download cost of $\mathcal{O}(N)$,

but by using the full scope of values of ℓ the download cost can be reduced by up to a factor of N, albeit with increasing recovery threshold. Reducing download cost generally also reduces decoding complexity, which can be important when downlink and/or computational resource at the user side is limited.

Next we provide an overview of the state of art approaches for coded distributed computing, summarize the key ideas behind cross-subspace alignment, and tabulate the comparisons between the codes proposed in this chapter and the prior state of art.

6.2 EP Codes, LCC Codes, CSA Codes

6.2.1 Matrix Partitioning: EP Codes

EP codes [141] for coded distributed matrix multiplication problem are based on matrix partitioning. The constituent matrices $\mathbf{A}$ and $\mathbf{B}$ are partitioned into $m \times p$ blocks and $p \times n$ blocks, respectively, as shown below, so that the desired matrix product involves a total of mn linear combinations of products of block matrices.

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}^{1,1} & \mathbf{A}^{1,2} & \cdots & \mathbf{A}^{1,p} \\ \mathbf{A}^{2,1} & \mathbf{A}^{2,2} & \cdots & \mathbf{A}^{2,p} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{A}^{m,1} & \mathbf{A}^{m,2} & \cdots & \mathbf{A}^{m,p} \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \mathbf{B}^{1,1} & \mathbf{B}^{1,2} & \cdots & \mathbf{B}^{1,n} \\ \mathbf{B}^{2,1} & \mathbf{B}^{2,2} & \cdots & \mathbf{B}^{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{B}^{p,1} & \mathbf{B}^{p,2} & \cdots & \mathbf{B}^{p,n} \end{pmatrix} \tag{6.1}$$

$$\mathbf{AB} = \begin{pmatrix} \sum_{j=1}^{p} \mathbf{A}^{1,j}\mathbf{B}^{j,1} & \sum_{j=1}^{p} \mathbf{A}^{1,j}\mathbf{B}^{j,2} & \cdots & \sum_{j=1}^{p} \mathbf{A}^{1,j}\mathbf{B}^{j,n} \\ \sum_{j=1}^{p} \mathbf{A}^{2,j}\mathbf{B}^{j,1} & \sum_{j=1}^{p} \mathbf{A}^{2,j}\mathbf{B}^{j,2} & \cdots & \sum_{j=1}^{p} \mathbf{A}^{2,j}\mathbf{B}^{j,n} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{j=1}^{p} \mathbf{A}^{m,j}\mathbf{B}^{j,1} & \sum_{j=1}^{p} \mathbf{A}^{m,j}\mathbf{B}^{j,2} & \cdots & \sum_{j=1}^{p} \mathbf{A}^{m,j}\mathbf{B}^{j,n} \end{pmatrix} \tag{6.2}$$

Coded matrices are constructed as follows,

$$\widetilde{A}(\alpha)=\sum_{m'\in[m]}\sum_{p'\in[p]}\mathbf{A}^{m',p'}\alpha^{p'-1+p(m'-1)}, \tag{6.3}$$

$$\widetilde{B}(\alpha)=\sum_{p'\in[p]}\sum_{n'\in[n]}\mathbf{B}^{p',n'}\alpha^{p-p'+pm(n'-1)}, \tag{6.4}$$

and the s^{th} server is sent the values $\widetilde{A}(\alpha_s)$ and $\widetilde{B}(\alpha_s)$. Here $\alpha_1,\alpha_2,\cdots,\alpha_S$ are distinct elements from the operating field $\mathbb{F}$. Each server produces the answer $\widetilde{A}(\alpha_s)\widetilde{B}(\alpha_s)$, which can be expressed as

$$\widetilde{A}(\alpha)\widetilde{B}(\alpha)=\sum_{i=1}^{R}\mathbf{C}^{(i)}\alpha^{i-1}, \tag{6.5}$$

where $R=pmn+p-1$ is the recovery threshold, and $\mathbf{C}^{(1)},\mathbf{C}^{(2)},\cdots,\mathbf{C}^{(R)}$ are various linear combinations of products of matrix blocks. Note that for all $i\in[R]$, $\mathbf{C}^{(i)}$ are distributed over $1,\alpha,\cdots,\alpha^{R-1}$, thus from the answers of any R servers, $\mathbf{C}^{(1)},\mathbf{C}^{(2)},\cdots,\mathbf{C}^{(R)}$ are recoverable by inverting a Vandermonde matrix. Furthermore, it is proved in [141] that by the construction of $\widetilde{A}(\alpha)$ and $\widetilde{B}(\alpha)$, the $\mathbf{C}^{(1)},\mathbf{C}^{(2)},\cdots,\mathbf{C}^{(R)}$ terms include the mn desired terms, while the remaining undesired terms (interference) align into the remaining $R-mn$ dimensions.

For example, suppose $p=m=n=2$, so that the coded matrices are constructed as follows.

$$\widetilde{A}(\alpha)=\mathbf{A}^{1,1}+\alpha\mathbf{A}^{1,2}+\alpha^2\mathbf{A}^{2,1}+\alpha^3\mathbf{A}^{2,2}, \tag{6.6}$$

$$\widetilde{B}(\alpha)=\alpha\mathbf{B}^{1,1}+\alpha^5\mathbf{B}^{1,2}+\mathbf{B}^{2,1}+\alpha^4\mathbf{B}^{2,2}. \tag{6.7}$$

And the answer can be expressed as follows.

$$\widetilde{A}(\alpha)\widetilde{B}(\alpha)=\underbrace{\mathbf{A}^{1,1}\mathbf{B}^{2,1}}_{\mathbf{C}^{(1)}}+\alpha\underbrace{(\mathbf{A}^{1,1}\mathbf{B}^{1,1}+\mathbf{A}^{1,2}\mathbf{B}^{2,1})}_{\mathbf{C}^{(2)}}+\alpha^2\underbrace{(\mathbf{A}^{2,1}\mathbf{B}^{2,1}+\mathbf{A}^{1,2}\mathbf{B}^{1,1})}_{\mathbf{C}^{(3)}}$$

$$
\begin{aligned}
&+\alpha^3\underbrace{(\mathbf{A}^{2,1}\mathbf{B}^{1,1}+\mathbf{A}^{2,2}\mathbf{B}^{2,1})}_{\mathbf{C}^{(4)}}+\alpha^4\underbrace{(\mathbf{A}^{1,1}\mathbf{B}^{2,2}+\mathbf{A}^{2,2}\mathbf{B}^{1,1})}_{\mathbf{C}^{(5)}}\\
&+\alpha^5\underbrace{(\mathbf{A}^{1,1}\mathbf{B}^{1,2}+\mathbf{A}^{1,2}\mathbf{B}^{2,2})}_{\mathbf{C}^{(6)}}+\alpha^6\underbrace{(\mathbf{A}^{1,2}\mathbf{B}^{1,2}+\mathbf{A}^{2,1}\mathbf{B}^{2,2})}_{\mathbf{C}^{(7)}}\\
&+\alpha^7\underbrace{(\mathbf{A}^{2,1}\mathbf{B}^{1,2}+\mathbf{A}^{2,2}\mathbf{B}^{2,2})}_{\mathbf{C}^{(8)}}+\alpha^8\underbrace{(\mathbf{A}^{2,2}\mathbf{B}^{1,2})}_{\mathbf{C}^{(9)}}.
\end{aligned}
\tag{6.8}
$$

Note that the desired product $\mathbf{AB}$ corresponds to the $mn=4$ terms $\mathbf{C}^{(2)},\mathbf{C}^{(4)},\mathbf{C}^{(6)},\mathbf{C}^{(8)}$, which appear along $\alpha,\alpha^3,\alpha^5,\alpha^7$. The remaining $R-mn=(pmn+p-1)-mn=5$ terms, i.e., $\mathbf{C}^{(1)},\mathbf{C}^{(3)},\mathbf{C}^{(5)},\mathbf{C}^{(7)},\mathbf{C}^{(9)}$ are undesired terms (interference).

In particular, note that the term $\mathbf{C}^{(9)}$, which is interference, has a higher order (α^8) than all desired terms. In general, EP codes produce $p-1$ such terms, namely $\mathbf{C}^{(pmn+1)},\cdots,\mathbf{C}^{(R)}$, that have a higher order than all desired terms. It turns out this is useful in the construction of GCSA codes to achieve better Interference Alignment (because these higher order terms produced by EP codes naturally align with the interference terms that result from batch processing).

EP codes may be seen as bridging the extremes of Polynomial codes and MatDot codes. Polynomial codes [140] can be recovered from EP codes by setting $p=1$, and MatDot codes [31] can be obtained from EP codes by setting $m=n=1$. EP codes also represent an improvement of PolyDot codes [31] within a factor of 2 in terms of recovery threshold, due to better interference alignment. Finally, EP codes have similar performance as Generalized PolyDot codes [28]. Thus, EP codes represent the state of art of prior work in terms of matrix partitioning approaches to coded distributed matrix multiplication.

6.2.2 Batch Processing: LCC Codes

Lagrange Coded Computing (LCC) codes [138] represent the state of art of prior work in terms of batch processing approaches for coded distributed batch multivariate polynomial evaluation, which includes as special cases distributed batch matrix multiplication as well as distributed batch N-linear computation. LCC codes are so named because they exploit the Lagrange interpolation polynomial to encode input data. For example, consider the multivariate polynomial $\Phi(\cdot)$ of total degree N, and suppose we are interested in batch evaluations of the polynomial, $\Phi(\mathbf{x}_1),\Phi(\mathbf{x}_2),\cdots,\Phi(\mathbf{x}_L)$ over the given batch of data points $\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_L$. Note that for matrix multiplication, $\mathbf{x}_l=(\mathbf{A}_l,\mathbf{B}_l)$ and $\Phi(\mathbf{x}_l)=\mathbf{A}_l\mathbf{B}_l$, which is a bilinear operation ($N=2$). LCC codes encode the dataset according to the Lagrange interpolation polynomial,

$$\widetilde{X}(\alpha)=\sum_{l\in[L]}\mathbf{x}_l\prod_{l'\in[L]\backslash\{l\}}\frac{\alpha-\beta_{l'}}{\beta_l-\beta_{l'}}, \tag{6.9}$$

and the s^{th} server is sent the evaluation $\widetilde{X}(\alpha_s)$. Here $\alpha_1,\alpha_2,\cdots,\alpha_S,\beta_1,\beta_2,\cdots,\beta_L$ are $(S+L)$ distinct elements from the operation field $\mathbb{F}$. The s^{th} server returns the user with the answer $\Phi(\widetilde{X}(\alpha_s))$. Note that the degree of the polynomial $\Phi(\widetilde{X}(\alpha))$ is less than or equal to $N(L-1)=NL-N$. Therefore, from the answers of any $R=NL-N+1$ servers, the user is able to reconstruct the polynomial $\Phi(\widetilde{X}(\alpha))$ by polynomial interpolation. Upon obtaining the polynomial $\Phi(\widetilde{X}(\alpha))$, the user evaluates it at β_l for every $l\in[L]$ to obtain $\Phi(\widetilde{X}(\beta_l))=\Phi(\mathbf{x}_l)$.

6.2.3 Cross Subspace Alignment: CSA Codes

The distinguishing feature of CSA codes is a Cauchy-Vandermonde structure that facilitates a form of interference alignment (labeled cross-subspace-alignment in Chapter 2), such that the desired symbols occupy dimensions corresponding to the Cauchy part, and everything else (interference) aligns within the higher order terms that constitute the Vandermonde part. As a simple example of the CSA codes introduced in this chapter, consider the problem of coded distributed batch matrix multiplication, and suppose we wish to compute the batch of $L=4$ matrix products $\mathbf{A}_1\mathbf{B}_1$, $\mathbf{A}_2\mathbf{B}_2$, $\mathbf{A}_3\mathbf{B}_3$, $\mathbf{A}_4\mathbf{B}_4$. For this, the $\mathbf{A}$ and $\mathbf{B}$ matrices are encoded into the form

$$\widetilde{A}(\alpha)=\Delta(\alpha)\left(\frac{1}{\underline{1}-\alpha}\mathbf{A}_1+\frac{1}{\underline{2}-\alpha}\mathbf{A}_2+\frac{1}{\underline{3}-\alpha}\mathbf{A}_3+\frac{1}{\underline{4}-\alpha}\mathbf{A}_4\right), \tag{6.10}$$

$$\widetilde{B}(\alpha)=\frac{1}{\underline{1}-\alpha}\mathbf{B}_1+\frac{1}{\underline{2}-\alpha}\mathbf{B}_2+\frac{1}{\underline{3}-\alpha}\mathbf{B}_3+\frac{1}{\underline{4}-\alpha}\mathbf{B}_4, \tag{6.11}$$

and the s^{th} server is sent the evaluations $\widetilde{A}(\alpha_s),\widetilde{B}(\alpha_s)$. Here the values $\alpha_1,\alpha_2,\cdots,\alpha_S,\underline{1},\underline{2},\cdots,\underline{4}$ represent any $S+4$ distinct elements of the operational field $\mathbb{F}$, and $\Delta(\alpha)=(\underline{1}-\alpha)(\underline{2}-\alpha)(\underline{3}-\alpha)(\underline{4}-\alpha)$. Each server multiplies its $\widetilde{A}(\alpha_s)$ with $\widetilde{B}(\alpha_s)$ producing an answer which (after some algebraic manipulation) can be expressed as

$$\begin{aligned}\widetilde{A}(\alpha)\widetilde{B}(\alpha)=c_1\left(\frac{1}{\underline{1}-\alpha}\right)\mathbf{A}_1\mathbf{B}_1+c_2\left(\frac{1}{\underline{2}-\alpha}\right)\mathbf{A}_2\mathbf{B}_2+c_3\left(\frac{1}{\underline{3}-\alpha}\right)\mathbf{A}_3\mathbf{B}_3\\ +c_4\left(\frac{1}{\underline{4}-\alpha}\right)\mathbf{A}_4\mathbf{B}_4+\mathbf{I}_1+\alpha\mathbf{I}_2+\alpha^2\mathbf{I}_3,\end{aligned} \tag{6.12}$$

where c_1,c_2,c_3,c_4 are non-zero constants. The desired matrix products $\mathbf{A}_i\mathbf{B}_i$ appear along $\left(\frac{1}{\underline{i}-\alpha}\right)$ (the Cauchy terms), and everything else (interference) can be distributed over the higher order terms $1,\alpha,\alpha^2$ (the Vandermonde terms) and consolidated into $\mathbf{I}_1,\mathbf{I}_2,\mathbf{I}_3$. The full-rank property of the Cauchy-Vandermonde matrix ensures that the desired symbols are separable from interference provided we have at least $R=7$ responding servers to resolve the 7 total dimensions (4 desired and 3 interference dimensions). Surprisingly, upon close inspection this special case of CSA codes turns out to be equivalent to the Lagrange Coded computing scheme for distributed matrix multiplication. However, CSA codes further generalize and improve upon the Lagrange Coded Computing approach as explained next.

Suppose we double the batch size from $L=4$ to $L=8$, i.e., we wish to compute the matrix products $\mathbf{A}_1\mathbf{B}_1$, $\mathbf{A}_2\mathbf{B}_2$, $\cdots$, $\mathbf{A}_8\mathbf{B}_8$. A straightforward extension is to simply use the previous scheme twice, which would double all costs. This could be accomplished equivalently with CSA codes or with Lagrange Coded Computing. However, because CSA codes generalize Lagrange Coded Computing, they offer much more flexibility. For example, we can partition the batch of $L=8$ $\mathbf{A}$,$\mathbf{B}$ matrices into $\ell=2$ sub-batches of $K_c=4$ matrices each, and then proceed as before, so that we have,

$$\widetilde{A}_1(\alpha)=\Delta_1(\alpha)\left(\frac{1}{\underline{1}-\alpha}\mathbf{A}_1+\frac{1}{\underline{2}-\alpha}\mathbf{A}_2+\frac{1}{\underline{3}-\alpha}\mathbf{A}_3+\frac{1}{\underline{4}-\alpha}\mathbf{A}_4\right), \tag{6.13}$$

$$\widetilde{A}_2(\alpha)=\Delta_2(\alpha)\left(\frac{1}{\underline{5}-\alpha}\mathbf{A}_5+\frac{1}{\underline{6}-\alpha}\mathbf{A}_6+\frac{1}{\underline{7}-\alpha}\mathbf{A}_7+\frac{1}{\underline{8}-\alpha}\mathbf{A}_8\right), \tag{6.14}$$

$$\widetilde{B}_1(\alpha)=\frac{1}{\underline{1}-\alpha}\mathbf{B}_1+\frac{1}{\underline{2}-\alpha}\mathbf{B}_2+\frac{1}{\underline{3}-\alpha}\mathbf{B}_3+\frac{1}{\underline{4}-\alpha}\mathbf{B}_4, \tag{6.15}$$

$$\widetilde{B}_2(\alpha)=\frac{1}{\underline{5}-\alpha}\mathbf{B}_5+\frac{1}{\underline{6}-\alpha}\mathbf{B}_6+\frac{1}{\underline{7}-\alpha}\mathbf{B}_7+\frac{1}{\underline{8}-\alpha}\mathbf{B}_8, \tag{6.16}$$

where $\Delta_1(\alpha)=(\underline{1}-\alpha)(\underline{2}-\alpha)(\underline{3}-\alpha)(\underline{4}-\alpha)$ and $\Delta_2(\alpha)=(\underline{5}-\alpha)(\underline{6}-\alpha)(\underline{7}-\alpha)(\underline{8}-\alpha)$. Evidently the upload cost is doubled. However, we will see that the download cost remains unchanged. This is because each server computes and (if responsive) returns (for its corresponding realization of α)

$$\widetilde{A}_1(\alpha)\widetilde{B}_1(\alpha)+\widetilde{A}_2(\alpha)\widetilde{B}_2(\alpha) \tag{6.17}$$

$$=c_1\left(\frac{1}{\underline{1}-\alpha}\right)\mathbf{A}_1\mathbf{B}_1+c_2\left(\frac{1}{\underline{2}-\alpha}\right)\mathbf{A}_2\mathbf{B}_2+\cdots+c_8\left(\frac{1}{\underline{8}-\alpha}\right)\mathbf{A}_8\mathbf{B}_8+\mathbf{I}_1'+\alpha\mathbf{I}_2'+\alpha^2\mathbf{I}_3'. \tag{6.18}$$

Since we have 8 desired dimensions and 3 interference dimensions, responses from any $R=11$ servers suffice to separate desired matrix products from interference. Remarkably, while the number of desired matrix products has doubled, the number of interference dimensions have not increased at all. This is why 4 additional responding servers allow us to recover 4 additional desired matrix products. This is an advantage unique to cross-subspace alignment, that cannot be achieved with other coding approaches, such as Lagrange Coded computing.

CSA codes for distributed matrix multiplication based on batch processing are introduced in this chapter in Theorem 6.1, a generalization to include matrix partitioning is presented in Theorem 6.2, and another generalization for N-linear batch computations and multivariate batch polynomial evaluations is presented in Theorem 6.3.

For ease of reference, Table 6.1 and Table 6.2 compare EP codes, LCC codes, CSA codes, GCSA codes and N-CSA codes with respect to their recovery thresholds, communication costs for uploads and downloads, encoding and decoding complexity, and server computation complexity.

This chapter is organized as follows. Section 6.3 presents the problem statements and definitions for coded distributed batch matrix multiplication (CDBMM), coded distributed N-linear batch computation and coded distributed multivariate batch polynomial evaluations. CSA codes for CDBMM are introduced in Section 6.4. Section 6.5 presents GCSA codes. N-CSA codes are presented in Section 6.6. Appendix D.1 presents further generalizations to allow X-secure data and B-byzantine servers. Section 6.7 concludes the chapter.

6.3 Problem Statement

6.3.1 Coded Distributed Batch Matrix Multiplication (CDBMM)

As shown in Figure 6.1, consider two source (master) nodes, each of which generates a sequence of L matrices, denoted as $\mathbf{A}=(\mathbf{A}_1,\mathbf{A}_2,\dots,\mathbf{A}_L)$ and $\mathbf{B}=(\mathbf{B}_1,\mathbf{B}_2,\dots,\mathbf{B}_L)$, such that for all $l\in[L]$, we have $\mathbf{A}_l\in\mathbb{F}^{\lambda\times\kappa}$ and $\mathbf{B}_l\in\mathbb{F}^{\kappa\times\mu}$, i.e., $\mathbf{A}_l$ and $\mathbf{B}_l$ are $\lambda\times\kappa$ and $\kappa\times\mu$ matrices, respectively, over a finite[3] field $\mathbb{F}$. The sink node (user) is interested in the sequence of

[3] With the exception of the generalizations to X-security presented in Appendix D.1, our coding schemes are applicable over infinite fields ($\mathbb{R}$,$\mathbb{C}$) as well. However, our problem statement assumes that $\mathbb{F}$ is a finite field, because of the difficulty of defining communication costs or computation complexity over infinite fields.

	Recovery Threshold (R)	Upload Cost (U_A,U_B)	Download Cost (D)
EP	$pmn+p-1$	$S/(pm),S/(pn)$	$(pmn+p-1)/(mn)$
codes	R	$\mathcal{O}(m)$, $\mathcal{O}(m)$	$\mathcal{O}(R/m^2)$
LCC	$2K_c'-1$	$S/K_c',S/K_c'$	$(2K_c'-1)/K_c'$
codes	R	$\mathcal{O}(1)$, $\mathcal{O}(1)$	$\mathcal{O}(1)$
CSA	$(\ell+1)K_c-1$	$S/K_c,S/K_c$	$((\ell+1)K_c-1)/(\ell K_c)$
codes	R	$\mathcal{O}(1)$, $\mathcal{O}(1)$	$\mathcal{O}(1)$
GCSA	$pmn((\ell+1)K_c''-1)+p-1$	$S/(K_c''pm),S/(K_c''pn)$	$\frac{pmn((\ell+1)K_c''-1)+p-1}{mn\ell K_c''}$
codes	R	$\mathcal{O}(m)$, $\mathcal{O}(m)$	$\mathcal{O}(p)$
	Server Computation Complexity $(\mathcal{C}_s)$	Encoding Complexity $(\mathcal{C}_{eA},\mathcal{C}_{eB})$	Decoding Complexity $(\mathcal{C}_d)$
EP	$\mathcal{O}(\lambda\mu\kappa/(pmn))$	$\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{pm}\right)$, $\widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{pn}\right)$	$\widetilde{\mathcal{O}}(\lambda\mu p\log^2 R)$
codes	$\mathcal{O}\left(\lambda^3/R\right)$	$\widetilde{\mathcal{O}}\left(\lambda^2 m\log^2 S\right)$, $\widetilde{\mathcal{O}}\left(\lambda^2 m\log^2 S\right)$	$\widetilde{\mathcal{O}}\left(\frac{\lambda^2 R\log^2 R}{m^2}\right)$
LCC	$\mathcal{O}(\lambda\mu\kappa/K_c')$	$\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_c'}\right)$, $\widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_c'}\right)$	$\widetilde{\mathcal{O}}(\lambda\mu\log^2 R)$
codes	$\mathcal{O}\left(\lambda^3/R\right)$	$\widetilde{\mathcal{O}}\left(\lambda^2\log^2 S\right)$, $\widetilde{\mathcal{O}}\left(\lambda^2\log^2 S\right)$	$\widetilde{\mathcal{O}}(\lambda^2\log^2 R)$
CSA	$\mathcal{O}(\lambda\mu\kappa/K_c)$	$\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_c}\right)$, $\widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_c}\right)$	$\widetilde{\mathcal{O}}(\lambda\mu\log^2 R)$
codes	$\mathcal{O}\left(\lambda^3/R\right)$	$\widetilde{\mathcal{O}}\left(\lambda^2\log^2 S\right)$, $\widetilde{\mathcal{O}}\left(\lambda^2\log^2 S\right)$	$\widetilde{\mathcal{O}}(\lambda^2\log^2 R)$
GCSA	$\mathcal{O}(\lambda\mu\kappa/(K_c''pmn))$	$\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_c''pm}\right)$, $\widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_c''pn}\right)$	$\widetilde{\mathcal{O}}(\lambda\mu p\log^2 R)$
codes	$\mathcal{O}\left(\lambda^3/R\right)$	$\widetilde{\mathcal{O}}\left(\lambda^2 m\log^2 S\right)$, $\widetilde{\mathcal{O}}\left(\lambda^2 m\log^2 S\right)$	$\widetilde{\mathcal{O}}(\lambda^2 p\log^2 R)$

Table 6.1: Performance summary of EP [141], LCC [138], CSA and GCSA codes for CDBMM. Note that choosing $\ell=K_c=1$ reduces GCSA codes to EP codes, while setting $m=n=p=1$ reduces GCSA codes to CSA codes (further restricting $\ell=1$ recovers LCC codes). Shaded rows represent balanced settings with $m=n,\lambda=\mu=\kappa$, fixed positive integers ℓ,ℓ'', and fixed ratio R/S. The batch size is $L=\ell K_c$ for CSA codes, $L'=K_c'$ for LCC codes, and $L''=\ell K_c''$ for GCSA codes. Note that for arbitrary R, we can choose the batch size to guarantee the feasibility of LCC/CSA/GCSA solutions.

	Recovery Threshold (R)	Upload Cost $(U_{X^{(n)}},n\in[N])$	Download Cost (D)
LCC	$NK_c'-N+1$	$S/K_c',S/K_c'$	$(NK_c'-N+1)/K_c'$
codes	R	$\mathcal{O}(N)$	$\mathcal{O}(N)$
N-**CSA**	$(N+\ell-1)K_c-N+1$	$S/K_c,S/K_c$	$((N+\ell-1)K_c-N+1)/(\ell K_c)$
codes	R	$\mathcal{O}(N+\ell-1)$	$\mathcal{O}(1+\frac{N-1}{\ell})$
	Computational Complexity $(\mathcal{C}_s)$	Encoding Complexity $(\mathcal{C}_{eX^{(n)}},n\in[N])$	Decoding Complexity $(\mathcal{C}_d)$
LCC	$\mathcal{O}(\omega/K_c')$	$\widetilde{\mathcal{O}}\left(\frac{\dim(V_n)S\log^2 S}{K_c'}\right)$	$\widetilde{\mathcal{O}}(\dim(W)N\log^2 R)$
codes	$\mathcal{O}(N\omega/R)$	$\widetilde{\mathcal{O}}\left(N\dim(V_n)\log^2 S\right)$	$\widetilde{\mathcal{O}}(\dim(W)N\log^2 R)$
N-**CSA**	$\mathcal{O}(\omega/K_c)$	$\widetilde{\mathcal{O}}\left(\frac{\dim(V_n)S\log^2 S}{K_c}\right)$	$\widetilde{\mathcal{O}}\left(\left(1+\frac{N-1}{\ell}\right)\dim(W)\log^2 R\right)$
codes	$\mathcal{O}((N+\ell-1)\omega/R)$	$\widetilde{\mathcal{O}}\left((N+\ell-1)\dim(V_n)\log^2 S\right)$	$\widetilde{\mathcal{O}}\left(\left(1+\frac{N-1}{\ell}\right)\dim(W)\log^2 R\right)$

Table 6.2: Performance summary of LCC codes [138] and N-CSA codes for N-linear distributed batch computation. Setting $\ell=1$ reduces N-CSA codes to LCC codes as a special case. Shaded rows represent settings with fixed ratio R/S. ω is the number of arithmetic operations required to compute the N-linear map $\Omega(\cdot)$. $\dim(V_n)$ is the dimension of the n^{th} variable of $\Omega(\cdot)$, $\dim W$ is the dimension of the output of $\Omega(\cdot)$. The batch size is $L=\ell K_c$ for N-CSA codes, and $L'=K_c'$ for LCC codes.

product matrices, $\mathbf{AB}=(\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\ldots,\mathbf{A}_L\mathbf{B}_L)$. To help with this computation, there are S servers (worker nodes). Each of the sources encodes its matrices according to the functions $\boldsymbol{f}=(f_1,f_2,\ldots,f_S)$ and $\boldsymbol{g}=(g_1,g_2,\ldots,g_S)$, where f_s and g_s correspond to the s^{th} server. Specifically, let us denote the encoded matrices for the s^{th} server as $\widetilde{A}^s$ and $\widetilde{B}^s$, so we have

$$\widetilde{A}^s=f_s(\mathbf{A}), \tag{6.19}$$

$$\widetilde{B}^s=g_s(\mathbf{B}). \tag{6.20}$$

The encoded matrices, $\widetilde{A}^s,\widetilde{B}^s$, are uploaded to the s^{th} server. Let us denote the number of elements from $\mathbb{F}$ in $\widetilde{A}^s$ and $\widetilde{B}^s$ as $|\widetilde{A}^s|$ and $|\widetilde{B}^s|$, respectively.

Upon receiving the encoded matrices, each Server s, $s\in[S]$, prepares (computes) a response Y_s, that is a function of $\widetilde{A}^s$ and $\widetilde{B}^s$, i.e.,

$$Y_s=h_s(\widetilde{A}^s,\widetilde{B}^s), \tag{6.21}$$

where $h_s,s\in[S]$ are the functions used to produce the answer, and we denote them collectively as $\boldsymbol{h}=(h_1,h_2,\ldots,h_S)$. Some servers may fail to respond, such servers are called stragglers. The user downloads the responses from the remaining servers, from which, using a class of decoding functions (denoted $\boldsymbol{d}$), he attempts to recover the desired product $\mathbf{AB}$. Define

$$\boldsymbol{d}=\{d_{\mathcal{R}}:\mathcal{R}\subset[S]\}, \tag{6.22}$$

where $d_{\mathcal{R}}$ is the decoding function used when the set of responsive servers is $\mathcal{R}$. We say that $(\boldsymbol{f},\boldsymbol{g},\boldsymbol{h},\boldsymbol{d})$ form a CDBMM code. A CDBMM code is said to be r-recoverable if the user is able to recover the desired products from the answers obtained from any r servers. In particular, a CDBMM 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

$$\mathbf{AB} = d_{\mathcal{R}}(Y_{\mathcal{R}}). \tag{6.23}$$

Define the recovery threshold R of a CDBMM code $(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{d})$ to be the minimum integer r such that the CDBMM code is r-recoverable.

The communication cost of CDBMM is comprised of upload and download costs. The (normalized)[4] upload costs U_A and U_B are defined as follows.

$$U_A = \frac{\sum_{s\in[S]} |\widetilde{A}^s|}{L\lambda\kappa}, \tag{6.24}$$

$$U_B = \frac{\sum_{s\in[S]} |\widetilde{B}^s|}{L\kappa\mu}. \tag{6.25}$$

Similarly, the (normalized) download cost is defined as follows.

$$D = \max_{\mathcal{R}, \mathcal{R}\subset[S], |\mathcal{R}|=R} \frac{\sum_{s\in\mathcal{R}} |Y_s|}{L\lambda\mu}, \tag{6.26}$$

where $|Y_s|$ is the number of elements from $\mathbb{F}$ in Y_s.

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 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. Note

[4] We normalize the upload cost and download cost with the number of elements contained in the constituent matrices $\mathbf{A}$,$\mathbf{B}$, and the desired product $\mathbf{AB}$, respectively.

that normalizations[5] by L are needed to have fair comparisons between batch processing approaches and individual matrix-partitioning solutions *per matrix multiplication.*

6.3.2 Distributed N-linear Batch Computation

Consider an N-linear map, which is a function of N variables that is linear separately in each variable. Formally, a map $\Omega : V_1 \times V_2 \times \cdots \times V_N \to W$ is called N-linear if for all $n \in [N]$,

$$\begin{aligned}&\Omega(x^{(1)},\cdots,x^{(n-1)},c_1x^{(n)}+c_2x'^{(n)},x^{(n+1)},\cdots,x^{(N)})\\&=c_1\Omega(x^{(1)},\cdots,x^{(n-1)},x^{(n)},x^{(n+1)},\cdots,x^{(N)})+c_2\Omega(x^{(1)},\cdots,x^{(n-1)},x'^{(n)},x^{(n+1)},\cdots,x^{(N)}),\end{aligned} \tag{6.27}$$

where $V_{[N]}$ and W are vector spaces over the base field $\mathbb{F}$, for all $i \in [N], x^{(i)} \in V_i$, $x'^{(n)} \in V_n$ and $c_1, c_2 \in \mathbb{F}$. Consider N sources (master nodes), $n \in [N]$, such that the n^{th} source generates a sequence of L variables $\mathbf{x}^{(n)} = (x_1^{(n)}, x_2^{(n)}, \cdots, x_L^{(n)})$, $x_l^{(n)} \in V_n, \forall l \in [L]$. Let us define

$$\mathbf{x}_l = (x_l^{(1)}, x_l^{(2)}, \cdots, x_l^{(N)}), \tag{6.28}$$

for all $l \in [L]$. The sink node (user) is interested in the evaluations of the N-linear map Ω over $\mathbf{x}_{[L]}$, i.e., $\Omega(x_l^{(1)}, x_l^{(2)}, \cdots, x_l^{(N)}) = \Omega(\mathbf{x}_l)$, $l \in [L]$. To help with this computation, there are S servers (worker nodes). For all $n \in [N]$, the n^{th} source encodes its variables according to the functions $\boldsymbol{f}^{(n)} = (f_1^{(n)}, f_2^{(n)}, \cdots, f_S^{(n)})$, where $f_s^{(n)}$ corresponds to the s^{th} server. Let us denote $(\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \cdots, \boldsymbol{f}^{(N)})$ collectively as $\boldsymbol{f}$. Like the problem of CDBMM, for all $n \in [N], s \in [S]$, the coded share of the n^{th} source for the s^{th} server is denoted as $\widetilde{X^{(n)}}^s$, and we have

$$\widetilde{X^{(n)}}^s = f_s^{(n)}(\mathbf{x}^{(n)}). \tag{6.29}$$

[5] Absolute latency constraints without such normalizations are also quite important in practice. See the discussion following Theorem 6.2 in Section 6.5 leading to Fig. 6.3.

$\left(\widetilde{X^{(n)}}^{s}\right)_{n\in[N]}$ are uploaded to the s^{th} server. Let us denote the number of elements in $\widetilde{X^{(n)}}^{s}$ as $\left|\widetilde{X^{(n)}}^{s}\right|$, $s\in[S], n\in[N]$.

Upon receiving the coded shares, each server s, $s\in[S]$ prepares (computes) a response Y_s, that is a function of $\widetilde{X^{(n)}}^{s}, n\in[N]$.

$$Y_s = h_s(\widetilde{X^{(1)}}^{s}, \widetilde{X^{(2)}}^{s}, \cdots, \widetilde{X^{(N)}}^{s}), \tag{6.30}$$

where $h_s, s\in[S]$ are the functions used to produce the answer, and we denote them collectively as $\boldsymbol{h} = (h_1, h_2, \cdots, h_S)$. The user downloads the responses from the servers in the set $\mathcal{R}$, and exploits a class of decoding functions (denoted $\boldsymbol{d}$) to recover the desired evaluations $\Omega(\mathbf{x}_l)$, $l\in[L]$. Define

$$\boldsymbol{d} = \{d_{\mathcal{R}} : \mathcal{R}\subset[S]\}, \tag{6.31}$$

where $d_{\mathcal{R}}$ is the decoding function used when the set of responsive servers is $\mathcal{R}$. We say that $(\boldsymbol{f}, \boldsymbol{h}, \boldsymbol{d})$ form a distributed N-linear batch computation code. A distributed N-linear batch computation code is said to be r-recoverable if the user is able to recover the desired evaluations from the answers obtained from any r servers. In particular, a distributed N-linear batch computation code $(\boldsymbol{f}, \boldsymbol{h}, \boldsymbol{d})$ is r-recoverable if for every $\mathcal{R}\subset[S]$, $|\mathcal{R}| = r$, and for every realization of $\mathbf{x}_{[L]}$, we have

$$(\Omega(\mathbf{x}_l))_{l\in[L]} = d_{\mathcal{R}}(Y_{\mathcal{R}}). \tag{6.32}$$

Define the recovery threshold R of a distributed N-linear batch computation code $(\boldsymbol{f}, \boldsymbol{h}, \boldsymbol{d})$ to be the minimum integer r such that the distributed N-linear batch computation code is r-recoverable.

The communication cost of distributed N-linear batch computation is comprised of upload and download costs. For all $n \in [N]$, the (normalized) upload cost for $\widetilde{X^{(n)}}^{[S]}$, denoted as $U_{X^{(n)}}$, is defined as follows

$$U_{X^{(n)}} = \frac{\sum_{s\in[S]} \left| \widetilde{X^{(n)}}^{s} \right|}{L\dim(V_n)}. \tag{6.33}$$

Similarly, the (normalized) download cost is defined as follows.

$$D = \max_{\mathcal{R},\mathcal{R}\subset[S],|\mathcal{R}=R|} \frac{\sum_{s\in\mathcal{R}} |Y_s|}{L\dim(W)}, \tag{6.34}$$

where $|Y_s|$ is the number of elements from $\mathbb{F}$ in Y_s.

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. For all $n \in [N]$, we also define the (normalized) encoding computational complexity $\mathcal{C}_{eX^{(n)}}$ for $\widetilde{X^{(n)}}^{[S]}$ as the order of the number of arithmetic operations required to compute the functions $\boldsymbol{f}^{(n)}$, normalized by L. Similarly, define the (normalized) decoding computational complexity $\mathcal{C}_d$ to be the order of the number of arithmetic operations required to compute $d_{\mathcal{R}}(Y_{\mathcal{R}})$, maximized over $\mathcal{R},\mathcal{R}\subset[S],|\mathcal{R}|=R$, and normalized by L.

6.3.3 Distributed Multivariate Polynomial Batch Evaluation

Consider a multivariate polynomial $\Phi : V_1 \times V_2 \times \cdots \times V_M \to W$ with M variables of total degree N, where $V_{[M]}$ and W are vector spaces over the base field $\mathbb{F}$. Consider M sources (master nodes). For all $m \in [M]$, the m^{th} source generates a sequence of L variables $\mathbf{x}^{(m)} = (x_1^{(m)}, x_2^{(m)}, \cdots, x_L^{(m)})$, such that for all $l \in [L]$, $x_l^{(m)} \in V_m$. Similarly, we define

$$\mathbf{x}_l = (x_l^{(1)}, x_l^{(2)}, \cdots, x_l^{(M)}), \tag{6.35}$$

for all $l \in [L]$. The sink node (user) wishes to compute the evaluations of the multivariate polynomial Φ over $\mathbf{x}_{[L]}$, i.e., $\Phi(x_l^{(1)}, x_l^{(2)}, \cdots, x_l^{(M)}) = \Phi(\mathbf{x}_l)$, $l \in [L]$, with the help of S servers (worker nodes). To this end, for all $m \in [M]$, the m^{th} source encodes its variables according to the functions $\boldsymbol{f}^{(m)} = (f_1^{(m)}, f_2^{(m)}, \cdots, f_S^{(m)})$, where $f_s^{(m)}$ corresponds to the s^{th} server. And $(\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \cdots, \boldsymbol{f}^{(M)})$ are collectively denoted as $\boldsymbol{f}$. For all $n \in [N], s \in [S]$, the coded share of the m^{th} source for the s^{th} server is denoted as $\widetilde{X^{(m)}}^s$, and we have

$$\widetilde{X^{(m)}}^s = f_s^{(m)}(\mathbf{x}^{(m)}). \tag{6.36}$$

$\left(\widetilde{X^{(m)}}^s\right)_{m \in [M]}$ are uploaded to the s^{th} server. Let us denote the number of elements in $\widetilde{X^{(m)}}^s$ as $\left|\widetilde{X^{(m)}}^s\right|$, $s \in [S], m \in [M]$.

Upon receiving coded variables, each server s, $s \in [S]$ prepares (computes) a response Y_s, that is a function of $\widetilde{X^{(m)}}^s, m \in [N]$.

$$Y_s = h_s(\widetilde{X^{(1)}}^s, \widetilde{X^{(2)}}^s, \cdots, \widetilde{X^{(M)}}^s), \tag{6.37}$$

where $h_s, s \in [S]$ are the functions used to produce the answer, and we denote them collectively as $\boldsymbol{h} = (h_1, h_2, \cdots, h_S)$. The user downloads the responses from the servers in the set $\mathcal{R}$, and uses a class of decoding functions (denoted $\boldsymbol{d}$) to recover the desired evaluations $\Phi(\mathbf{x}_l)$, $l \in [L]$. Define

$$\boldsymbol{d} = \{d_{\mathcal{R}} : \mathcal{R} \subset [S]\}, \tag{6.38}$$

where $d_{\mathcal{R}}$ is the decoding function used when the set of responsive servers is $\mathcal{R}$. We say that $(\boldsymbol{f}, \boldsymbol{h}, \boldsymbol{d})$ form a distributed multivariate polynomial batch evaluation code. A distributed multivariate polynomial batch evaluation code is said to be r-recoverable if the user is able to recover the desired evaluations from the answers obtained from any r servers, i.e., for any

$\mathcal{R}\subset[S]$, $|\mathcal{R}|=r$, and for any realization of $\mathbf{x}_{[L]}$, we have

$$(\Phi(\mathbf{x}_l))_{l\in[L]}=d_{\mathcal{R}}(Y_{\mathcal{R}}). \tag{6.39}$$

Define the recovery threshold R of a distributed multivariate polynomial batch evaluation code $(\boldsymbol{f},\boldsymbol{h},\boldsymbol{d})$ to be the minimum integer r such that the distributed multivariate polynomial batch evaluation code is r-recoverable.

For all $m\in[M]$, the (normalized) upload cost for $\widetilde{X^{(m)}}^{[S]}$, denoted as $U_{X^{(m)}}$, is defined as follows

$$U_{X^{(m)}}=\frac{\sum_{s\in[S]}\left|\widetilde{X^{(m)}}^{s}\right|}{L\dim(V_m)}. \tag{6.40}$$

Similarly, the (normalized) download cost is defined as follows.

$$D=\max_{\mathcal{R},\mathcal{R}\subset[S],|\mathcal{R}=R|}\frac{\sum_{s\in\mathcal{R}}|Y_s|}{L\dim(W)}, \tag{6.41}$$

where $|Y_s|$ is the number of elements from $\mathbb{F}$ in Y_s.

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. For all $m\in[M]$, we also define the (normalized) encoding computational complexity $\mathcal{C}_{eX^{(m)}}$ for $\widetilde{X^{(m)}}^{[S]}$ as the order of the number of arithmetic operations required to compute the functions $\boldsymbol{f}^{(m)}$, normalized by L. Similarly, define the (normalized) decoding computational complexity $\mathcal{C}_d$ to be the order of the number of arithmetic operations required to compute $d_{\mathcal{R}}(Y_{\mathcal{R}})$, maximized over $\mathcal{R},\mathcal{R}\subset[S],|\mathcal{R}|=R$, and normalized by L.

6.4 CSA Codes for CDBMM

6.4.1 CSA Codes: Main Result

The main result of this section introduces CSA Codes, and is stated in the following theorem.

Theorem 6.1. *For CDBMM over a field $\mathbb{F}$ with S servers, and positive integers ℓ, K_c such that $L=\ell K_c \le |\mathbb{F}|-S$, the CSA codes introduced in this chapter achieve*

$$\begin{aligned}
&\textit{Recovery Threshold:} && R=(\ell+1)K_c-1, && (6.42)\\
&\textit{Upload Cost for } \widetilde{A}^{[S]},\widetilde{B}^{[S]}\textit{:} && (U_A,U_B)=\left(\frac{S}{K_c},\frac{S}{K_c}\right), && (6.43)\\
&\textit{Download Cost:} && D=\frac{(\ell+1)K_c-1}{\ell K_c}, && (6.44)\\
&\textit{Server Computation Complexity:} && \mathcal{C}_s=\mathcal{O}(\lambda\kappa\mu/K_c), && (6.45)\\
&\textit{Encoding Complexity for } \widetilde{A}^{[S]},\widetilde{B}^{[S]}\textit{:} && (\mathcal{C}_{eA},\mathcal{C}_{eB})=\left(\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_c}\right),\widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_c}\right)\right), && (6.46)\\
&\textit{Decoding Complexity:} && \mathcal{C}_d=\widetilde{\mathcal{O}}\left(\lambda\mu\log^2 R\right). && (6.47)
\end{aligned}$$

The proof of Theorem 6.1 appears in Section 6.4.3. A high level summary of the main ideas is provided here. CSA codes split the $L=\ell K_c$ instances of $\mathbf{A}_l$ matrices into ℓ groups, each containing K_c matrices. The K_c matrices within each group are coded into an MDS (S,K_c) code by a Cauchy encoding matrix to create S linear combinations of these K_c matrices. Multiplication with a Cauchy encoding matrix corresponds to the well studied Trummer's problem [42] for which fast algorithms have been found in [39, 38, 84] that limit the encoding complexity to $\mathcal{C}_{eA}=\widetilde{\mathcal{O}}(\frac{\lambda\kappa S\log^2 S}{K_c})$. The s^{th} coded linear combination from each of the ℓ groups is sent to the s^{th} server. The $\mathbf{B}_l$ matrices are similarly encoded and uploaded to the S servers. Note that because K_c matrices are linearly combined into one linear combination for each server, and there are S servers, the upload cost of CSA codes is S/K_c. Each server multiplies the corresponding instances of coded $\mathbf{A}$,$\mathbf{B}$ matrices and returns the sum of these ℓ products. With straightforward matrix multiplication algorithms, each of the ℓ matrix products has a computation complexity of $\mathcal{O}(\lambda\kappa\mu)$ for a total of $\mathcal{O}(\ell\lambda\kappa\mu)$, which upon normalization by $L=\ell K_c$, yields a complexity of $\mathcal{C}_s=\mathcal{O}(\lambda\kappa\mu/K_c)$ per server. The responses from any $R=(\ell+1)K_c-1$ servers provide R observations to the user, each comprised of linear combinations of various product matrices, including both desired products and undesired products (interference). Interpreting the R observations as occupying an R-dimensional vector space, the L desired matrix products $(\mathbf{A}_l\mathbf{B}_l)_{l\in[L]}$ occupy $L=\ell K_c$ of these R dimensions, leaving only $R-L=K_c-1$ dimensions for interference. Remarkably, while there are a total of $\ell K_c(K_c-1)$ undesired matrix products, $\mathbf{A}_l\mathbf{B}_{l'},l\neq l'$ that appear in the responses from the servers, they collectively occupy only a total of K_c-1 dimensions. This is because of *cross-subspace alignment*, facilitated by the specialized Cauchy structure of the encoding. Since $L=\ell K_c$ desired matrix products are recovered from a total of R that are downloaded, the normalized download cost is $\frac{R}{L}=\frac{(\ell+1)K_c-1}{\ell K_c}$. Note that the decoding operation involves inverting a Cauchy-Vandermonde matrix, where the Cauchy part spans the dimensions carrying desired signals while the Vandermonde part spans the dimensions carrying interference. Fast algorithms for inverting such matrices are also known [35], which limits the decoding complexity to $\widetilde{\mathcal{O}}(\lambda\mu\log^2 R)$.

6.4.2 Observations

In this section we present some observations to place CSA Codes into perspective. In particular we would like to compare CSA codes which generalize and improve upon the state of art of batch processing approaches (LCC codes), against EP codes which represent the state of art for matrix-partitioning approaches.

1. From the conditions of Theorem 6.1, the field size $|\mathbb{F}|$ must be at least equal to $S+L$. However, it is possible to reduce the field size requirement to $|\mathbb{F}| \geq S$ by constructing a *systematic* version of the code (see Section 6.4.4).

2. To estimate the complexity of computation at each server we use only straightforward matrix multiplication algorithms that require $\widetilde{\mathcal{O}}(\lambda\mu\kappa)$ arithmetic operations over $\mathbb{F}$ in order to compute the product of a $\lambda \times \mu$ matrix with a $\mu \times \kappa$ matrix. It is well known that this complexity can be improved upon by using more sophisticated[6] algorithms [99, 25, 67]. Such improvements do not constitute a relative advantage because they can be applied similarly to other codes, such as Entangled Polynomial codes as well.

3. We are primarily interested in balanced settings, e.g., $\lambda = \mu = \kappa$, that are typically studied for complexity analysis. While the achievability claims of Theorem 6.1 are also applicable to unbalanced settings, it is not difficult to improve upon Theorem 6.1 in certain aspects in highly unbalanced settings. For example, as shown recently in [51], when $\kappa \ll \min(\lambda, \mu)$, it may be significantly beneficial for the user in terms of download cost to retrieve the $\mathbf{A}$,$\mathbf{B}$ matrices separately from the distributed servers and do the computation locally.

4. First let us compare CSA codes with LCC codes, both of which are based on batch processing. Remarkably, setting $\ell = 1$ in CSA codes recovers the LCC code for CDBMM, i.e., LCC codes are a special case of CSA codes. The parameter ℓ in CSA codes is mainly[7] useful to reduce download cost (by choosing large ℓ). On the one hand, note that the download cost in (6.44) is always bounded between 1 and 2, so even the worst case choice of ℓ will at most double the download cost. So if the download cost is only important in the $\mathcal{O}$ sense (as a function of R), then it is desirable to set $\ell = 1$ and

[6] Notably, for $\lambda = \mu = \kappa$ the best known algorithms [67] thus far have computation complexity that is still super-quadratic (more than $\mathcal{O}(\lambda^{2.3})$), and are not considered practical [45] due to large hidden constants in the $\mathcal{O}$ notation.

[7] ℓ may be also useful for parallel processing within each server because the computation at each server is naturally split into ℓ independent computations.

$K_c = L$ which reduces the number of parameters for the coding scheme. On the other hand, for settings where the download cost is the dominating concern, the generalization to $\ell > 1$ is important. For example, suppose for some application due to latency concerns there is a hard threshold that the download from each server cannot exceed the equivalent of one matrix multiplication, i.e., no more than λ^2 elements of $\mathbb{F}$. Then for large batch sizes L, the lower download cost of CSA codes translates into a smaller recovery threshold by up to a factor of 2 relative to LCC codes (albeit at the cost of increased upload and server computation).

5. Next, let us compare the performance of CSA codes with Entangled Polynomial[8] codes[141]. For this comparison we will only focus on $\ell = 1$, so this also applies equivalently to LCC codes which are obtained as special cases of CSA codes when $\ell = 1$. In order to compute a batch of matrix products $(\mathbf{A}_l\mathbf{B}_l)_{l\in[L]}$, we will show that joint/batch processing of all L products with CSA codes achieves significantly better communication (upload-download) costs than separate application of Entangled Polynomial codes for each $l \in [L]$, under the same recovery-threshold-computational-complexity-trade-off. It is proved in [141] that for any positive integers (p,m,n), Entangled Polynomial codes achieve

$$\text{Recovery threshold:} \quad R = pmn + p - 1, \tag{6.48}$$

$$\text{Upload cost:} \quad (U_A, U_B) = (S/pm, S/pn), \tag{6.49}$$

$$\text{Download cost:} \quad D = \frac{pmn + p - 1}{mn}. \tag{6.50}$$

To simplify the order analysis, let us assume that $\lambda = \kappa = \mu$, and to balance the upload costs (U_A, U_B) let us choose $m = n$. Let us regard the recovery threshold R as a variable, and consider the upload cost U_A, U_B and the download cost D as functions of R. So

[8] Entangled Polynomial codes generalize MatDot codes and Polynomial codes, improve upon PolyDot codes, and have similar performance as Generalized PolyDot codes, so it suffices to compare CSA codes with Entangled Polynomial codes.

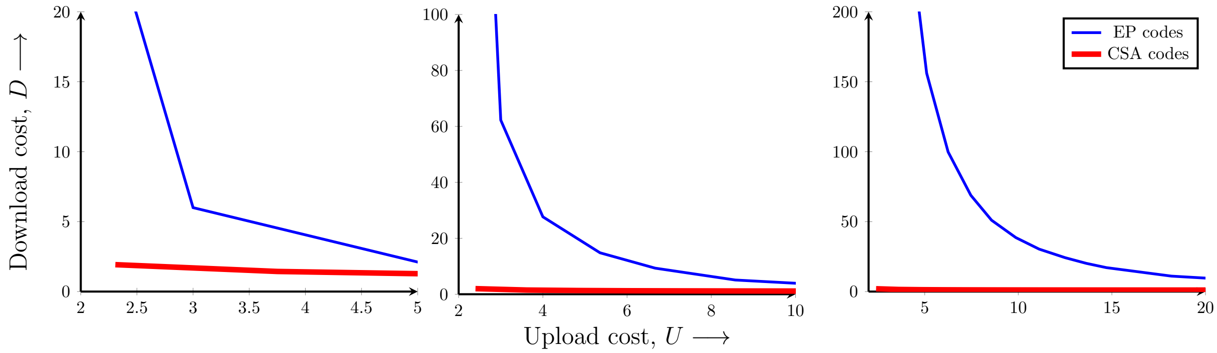
for the Entangled Polynomial codes [141], we have

$$U_A = U_B = U = \frac{mS}{pm^2} \geq m\left(\frac{S}{R}\right), \qquad\qquad D = \frac{R}{m^2}. \tag{6.51}$$

A tradeoff is evident. For example, if we want download cost of $\mathcal{O}(1)$, then we need $m = \Theta(\sqrt{R})$ which yields upload cost of $\mathcal{O}(S/\sqrt{R})$. On the other hand, if we want upload cost of $\mathcal{O}(S/R)$, then we should set $m = \Theta(1)$ which yields download cost of $\mathcal{O}(R)$. If S/R is held constant, then to best balance the upload and download cost, we need $m = \Theta(R^{1/3})$, which yields both upload cost and download cost of $\mathcal{O}(R^{1/3})$. Evidently, it is not possible to achieve both upload and download cost of $\mathcal{O}(1)$ with Entagled Polynomial codes. However, with CSA codes, setting $\ell = 1$, we have upload cost of $\mathcal{O}(1)$ and download cost of $\mathcal{O}(S/R)$. In particular, if S/R is a constant, then both upload and download costs are $\mathcal{O}(1)$. Note that CSA codes have the same server computational complexity of $\mathcal{O}(\lambda^3/R)$ *normalized by batch size* as EP codes.

6. Continuing with the comparison between CSA codes and EP codes, Figure 6.2a, 6.2b and 6.2c show lower convex hulls of achievable (balanced upload cost, download cost) pairs of Entangled Polynomial codes and CSA codes given the number of servers and the recovery threshold $(S = 30, R \leq 25)$, $(S = 300, R \leq 250)$ and $(S = 3000, R \leq 2500)$ respectively. Each value of (S,R) produces an achievable region in the (U,D) plane (including all possible choices of m,n,p parameters for Entagled Polynomial codes, and all choices of ℓ, K_c parameters for CSA codes). What is shown in the figure is the union of these regions for each case, e.g., in the first figure the union is over all (S,R) with $(S = 30, R \leq 25)$. Evidently, the advantage of CSA codes over Entangled Polynomial codes in terms of communication cost is significant and grows stronger for larger (S,R) values.

7. CSA codes show a similar advantage over Entangled Polynomial codes in terms of the tradeoff between encoding complexity and decoding complexity normalized by



(a) $S=30, R\leq 25$ (b) $S=300, R\leq 250$ (c) $S=3000, R\leq 2500$

Figure 6.2: Lower convex hulls of achievable (balanced upload cost, download cost) pairs (U,D) of Entangled Polynomial codes (EP codes) and *cross subspace alignment codes* (CSA codes) given (a) $(S=30, R\leq 25)$, (b) $(S=300, R\leq 250)$ and (c) $(S=3000, R\leq 2500)$.

batch size. For example, consider the balanced setting of $m=n$, $\lambda=\mu=\kappa$, and constant S/R. The encoding complexity of Entangled Polynomial codes is $\widetilde{\mathcal{O}}(\lambda^2 U\log^2 S)$, and the decoding complexity is $\widetilde{\mathcal{O}}(\lambda^2 D\log^2 R)$, where $U=U_A=U_B$ is the balanced upload cost, and D is the download cost. For CSA codes, the encoding complexity is $\widetilde{\mathcal{O}}(\lambda^2\log^2 S)$, and the decoding complexity is $\widetilde{\mathcal{O}}(\lambda^2\log^2 R)$, which corresponds to $U=D=\mathcal{O}(1)$. Thus, the communication cost advantage of CSA codes over Entangled Polynomial codes is further manifested in the improved tradeoff between encoding and decoding complexity.

8. Finally, let us place CSA codes in perspective with previous applications of cross-subspace alignment. The idea of cross-subspace alignment was introduced in the context of X-secure T-private information retrieval (XSTPIR) in Chapter 2. The goal of XSTPIR is to allow a user to retrieve, as efficiently as possible, a desired message W_θ out of K messages, $W_1, W_2, \cdots, W_K$ that are 'secret-shared' across S servers in an X-secure fashion, without revealing any information about the index θ to any group of up to T colluding servers. According to the scheme proposed in Chapter 2 the ℓ^{th} symbol of each message is stored in the $1\times K$ vector $\mathbf{W}_\ell$. The query vector $\mathbf{Q}_\theta$ is the θ^{th} column of a $K\times K$ identity matrix, so that retrieving the product $\mathbf{W}_\ell\mathbf{Q}_\theta$ retrieves the ℓ^{th}

symbol of the desired message W_θ. In order to guarantee security of data and privacy of queries, the $\mathbf{W}_\ell$ and $\mathbf{Q}_\theta$ vectors are mixed with independent noise terms. Intuitively, by replacing $\mathbf{W}_\ell$ and $\mathbf{Q}_\theta$ with matrices $\mathbf{A}$ and $\mathbf{B}$, and eliminating the corresponding noise terms if the privacy and/or security constraints are relaxed, cross-subspace alignment schemes can be used to retrieve arbitrary matrix products $\mathbf{AB}$. This intuition helps with some of the achievable schemes[9] in [51, 61]. However, in Chapter 2, the $\mathbf{W}_\ell$ vectors are not jointly encoded. Each $\mathbf{W}_\ell$ vector is separately mixed with noise. Similarly, in [51, 61] the matrices are separately mixed with noise for security, and not jointly encoded. Joint encoding of messages arises in PIR when instead of replicated storage [102, 106], coded storage is assumed [11, 108, 36, 105, 142, 116]. PIR with MDS-coded storage, X-secure data and T-private queries is studied in [54] and indeed a generalized cross-subspace alignment scheme is the key contribution of [54]. However, since there is only one query vector $\mathbf{Q}_\theta$, applications of this cross-subspace alignment scheme are useful primarily for matrix multiplications of the form $\mathbf{A}_1\mathbf{B},\mathbf{A}_2\mathbf{B},\cdots,\mathbf{A}_L\mathbf{B}$, where we have only one $\mathbf{B}$ matrix to be multiplied with each $\mathbf{A}$ matrix. This is indeed how the scheme is applied in the context of private secure distributed matrix multiplication (PSDMM) in [54]. Batch multiplications of the form $\mathbf{A}_1\mathbf{B}_1,\mathbf{A}_2\mathbf{B}_2,\cdots,\mathbf{A}_L\mathbf{B}_L$, that are studied in this chapter, present a significantly greater challenge in that joint coding is now to be applied both among $\mathbf{A}_1,\mathbf{A}_2,\cdots,\mathbf{A}_L$ and among $\mathbf{B}_1,\mathbf{B}_2,\cdots,\mathbf{B}_L$ matrices, which introduces new interference terms $\mathbf{A}_l\mathbf{B}_{l'},l\neq l'$. A central technical challenge behind this chapter is to determine if and how these terms can be aligned. The CSA codes introduced in this chapter present a solution to this challenge.

[9] Notably, batch processing is used in [51] while matrix partitioning is used in [61]. The achievable schemes in [51, 61] can be regarded as special cases of X-secure CSA codes presented in this chapter with $K_c=1$. See Appendix D.1 for details.

6.4.3 Proof of Theorem 6.1

In this section, we present the construction of CSA codes. Let $L=\ell K_c$. Before presenting the general code construction let us start with some illustrative examples.

$\ell=2, K_c=2, L=4$

Let $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, \alpha_1, \ldots, \alpha_S$ represent $(S+\ell K_c)=(S+4)$ distinct elements from $\mathbb{F}$. For all $s\in[S]$, let us define

$$\Delta_s^{1,2}=(f_{1,1}-\alpha_s)(f_{1,2}-\alpha_s), \tag{6.52}$$

$$\Delta_s^{2,2}=(f_{2,1}-\alpha_s)(f_{2,2}-\alpha_s). \tag{6.53}$$

Let us set $\mathbf{A}_{l,k}=\mathbf{A}_{K_c(l-1)+k}$ and $\mathbf{B}_{l,k}=\mathbf{B}_{K_c(l-1)+k}$ for all $l\in[2], k\in[2]$. Coded shares of matrices $\mathbf{A}$ are constructed as follows.

$$\widetilde{A}^s=(\widetilde{A}_1^s, \widetilde{A}_2^s), \tag{6.54}$$

where

$$\begin{aligned}
\widetilde{A}_1^s &= \Delta_s^{1,2}\left(\frac{1}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}+\frac{1}{f_{1,2}-\alpha_s}\mathbf{A}_{1,2}\right) && (6.55)\\
&=(f_{1,2}-\alpha_s)\mathbf{A}_{1,1}+(f_{1,1}-\alpha_s)\mathbf{A}_{1,2}, && (6.56)\\
\widetilde{A}_2^s &= \Delta_s^{2,2}\left(\frac{1}{f_{2,1}-\alpha_s}\mathbf{A}_{2,1}+\frac{1}{f_{2,2}-\alpha_s}\mathbf{A}_{2,2}\right) && (6.57)\\
&=(f_{2,2}-\alpha_s)\mathbf{A}_{2,1}+(f_{2,1}-\alpha_s)\mathbf{A}_{2,2}. && (6.58)
\end{aligned}$$

Coded shares of matrices **B** are constructed as follows.

$$\widetilde{B}^s = (\widetilde{B}_1^s, \widetilde{B}_2^s), \tag{6.59}$$

where

$$\widetilde{B}_1^s = \frac{1}{f_{1,1}-\alpha_s}\mathbf{B}_{1,1} + \frac{1}{f_{1,2}-\alpha_s}\mathbf{B}_{1,2}, \tag{6.60}$$

$$\widetilde{B}_2^s = \frac{1}{f_{2,1}-\alpha_s}\mathbf{B}_{2,1} + \frac{1}{f_{2,2}-\alpha_s}\mathbf{B}_{2,2}. \tag{6.61}$$

The answer provided by the s^{th} server to the user is constructed as follows.

$$Y_s = \widetilde{A}_1^s \widetilde{B}_1^s + \widetilde{A}_2^s \widetilde{B}_2^s. \tag{6.62}$$

To see why the $R = (\ell+1)K_c - 1 = 5$ recovery threshold holds, we rewrite Y_s as follows.

$$\begin{aligned}
Y_s &= \widetilde{A}_1^s \widetilde{B}_1^s + \widetilde{A}_2^s \widetilde{B}_2^s && (6.63)\\
&= \frac{f_{1,2}-\alpha_s}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}\mathbf{B}_{1,1} + \frac{f_{1,1}-\alpha_s}{f_{1,2}-\alpha_s}\mathbf{A}_{1,2}\mathbf{B}_{1,2} + \frac{f_{2,2}-\alpha_s}{f_{2,1}-\alpha_s}\mathbf{A}_{2,1}\mathbf{B}_{2,1} + \frac{f_{2,1}-\alpha_s}{f_{2,2}-\alpha_s}\mathbf{A}_{2,2}\mathbf{B}_{2,2}\\
&\quad + (\mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{1,1} + \mathbf{A}_{2,1}\mathbf{B}_{2,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}) && (6.64)\\
&= \frac{f_{1,1}-\alpha_s + (f_{1,2}-f_{1,1})}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}\mathbf{B}_{1,1} + \frac{f_{1,2}-\alpha_s + (f_{1,1}-f_{1,2})}{f_{1,2}-\alpha_s}\mathbf{A}_{1,2}\mathbf{B}_{1,2}\\
&\quad + \frac{f_{2,1}-\alpha_s + (f_{2,2}-f_{2,1})}{f_{2,1}-\alpha_s}\mathbf{A}_{2,1}\mathbf{B}_{2,1} + \frac{f_{2,2}-\alpha_s + (f_{2,1}-f_{2,2})}{f_{2,2}-\alpha_s}\mathbf{A}_{2,2}\mathbf{B}_{2,2}\\
&\quad + (\mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{1,1} + \mathbf{A}_{2,1}\mathbf{B}_{2,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}) && (6.65)\\
&= \frac{f_{1,2}-f_{1,1}}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}\mathbf{B}_{1,1} + \frac{f_{1,1}-f_{1,2}}{f_{1,2}-\alpha_s}\mathbf{A}_{1,2}\mathbf{B}_{1,2}\\
&\quad + \frac{f_{2,2}-f_{2,1}}{f_{2,1}-\alpha_s}\mathbf{A}_{2,1}\mathbf{B}_{2,1} + \frac{f_{2,1}-f_{2,2}}{f_{2,2}-\alpha_s}\mathbf{A}_{2,2}\mathbf{B}_{2,2}\\
&\quad + \underbrace{\begin{aligned}&(\mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{1,2} + \mathbf{A}_{2,1}\mathbf{B}_{2,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}\\ &\quad + \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{1,1} + \mathbf{A}_{2,1}\mathbf{B}_{2,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,1}).\end{aligned}}_{I_1} && (6.66)
\end{aligned}$$

Therefore, for any $R=5$ servers, whose indices are denoted as $s_1,s_2,\dots,s_5$, we can represent their answers in the following matrix form.

$$\begin{bmatrix} Y_{s_1}\\ Y_{s_2}\\ Y_{s_3}\\ Y_{s_4}\\ Y_{s_5}\end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_{1,1}-\alpha_{s_1}} & \frac{1}{f_{1,2}-\alpha_{s_1}} & \frac{1}{f_{2,1}-\alpha_{s_1}} & \frac{1}{f_{2,2}-\alpha_{s_1}} & 1\\ \frac{1}{f_{1,1}-\alpha_{s_2}} & \frac{1}{f_{1,2}-\alpha_{s_2}} & \frac{1}{f_{2,1}-\alpha_{s_2}} & \frac{1}{f_{2,2}-\alpha_{s_2}} & 1\\ \frac{1}{f_{1,1}-\alpha_{s_3}} & \frac{1}{f_{1,2}-\alpha_{s_3}} & \frac{1}{f_{2,1}-\alpha_{s_3}} & \frac{1}{f_{2,2}-\alpha_{s_3}} & 1\\ \frac{1}{f_{1,1}-\alpha_{s_4}} & \frac{1}{f_{1,2}-\alpha_{s_4}} & \frac{1}{f_{2,1}-\alpha_{s_4}} & \frac{1}{f_{2,2}-\alpha_{s_4}} & 1\\ \frac{1}{f_{1,1}-\alpha_{s_5}} & \frac{1}{f_{1,2}-\alpha_{s_5}} & \frac{1}{f_{2,1}-\alpha_{s_5}} & \frac{1}{f_{2,2}-\alpha_{s_5}} & 1 \end{bmatrix}}_{\mathbf{V}_{2,2,5}} \underbrace{\begin{bmatrix} c_{1,1} & & & & \\ & c_{1,2} & & & \\ & & c_{2,1} & & \\ & & & c_{2,2} & \\ & & & & 1 \end{bmatrix}}_{\mathbf{V}'_{2,2,5}} \otimes \mathbf{I}_\lambda \begin{bmatrix} \mathbf{A}_{1,1}\mathbf{B}_{1,1}\\ \mathbf{A}_{1,2}\mathbf{B}_{1,2}\\ \mathbf{A}_{2,1}\mathbf{B}_{2,1}\\ \mathbf{A}_{2,2}\mathbf{B}_{2,2}\\ I_1 \end{bmatrix}, \tag{6.67}$$

where $c_{1,1}=f_{1,2}-f_{1,1},\ c_{2,1}=f_{2,2}-f_{2,1},\ c_{1,2}=-c_{1,1},\ c_{2,2}=-c_{2,1}$. Since $f_{1,1},f_{1,2},f_{2,1},f_{2,2}$ are distinct elements from $\mathbb{F}$, the constants $c_{1,1},c_{1,2},c_{2,1},c_{2,2}$ take non-zero values. Guaranteed by Lemma 2.5 and the fact that Kronecker product of non-singular matrices is non-singular, the $5\lambda\times 5\lambda$ matrix $(\mathbf{V}_{2,2,5}\mathbf{V}'_{2,2,5})\otimes\mathbf{I}_\lambda$ is invertible, and the user is able to recover desired products $(\mathbf{A}_1\mathbf{B}_1,\dots,\mathbf{A}_4\mathbf{B}_4)=(\mathbf{A}_{l,k}\mathbf{B}_{l,k})_{l\in[2],k\in[2]}$ from the answers received from any $R=5$ servers. This completes the proof of the $R=5$ recovery threshold. Finally, note that the upload cost is $U_A=U_B=S/2=S/K_c$ and the download cost is $D=4/5$ because a total of $R=5$ matrix products, each of dimension $\lambda\times\mu$, are downloaded (one from each server) from which the 4 desired matrix products, also each of dimension $\lambda\times\mu$, are recovered.

$\ell=1,K_c=3,L=3$

Let $f_{1,1},f_{1,2},f_{1,3},\alpha_1,\alpha_2,\dots,\alpha_S$ represent $(S+\ell K_c)=(S+3)$ distinct elements from $\mathbb{F}$. For all $s\in[S]$, let us define

$$\Delta_s^{1,3}=(f_{1,1}-\alpha_s)(f_{1,2}-\alpha_s)(f_{1,3}-\alpha_s). \tag{6.68}$$

Shares of $\mathbf{A}$ and $\mathbf{B}$ at the s^{th} server are constructed as follows.

$$\widetilde{A}^s = \Delta_s^{1,3}\left(\frac{1}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}+\frac{1}{f_{1,2}-\alpha_s}\mathbf{A}_{1,2}+\frac{1}{f_{1,3}-\alpha_s}\mathbf{A}_{1,3}\right), \tag{6.69}$$

$$\widetilde{B}^s = \frac{1}{f_{1,1}-\alpha_s}\mathbf{B}_{1,1}+\frac{1}{f_{1,2}-\alpha_s}\mathbf{B}_{1,2}+\frac{1}{f_{1,3}-\alpha_s}\mathbf{B}_{1,3}, \tag{6.70}$$

where we set $\mathbf{A}_{1,k}=\mathbf{A}_k$ and $\mathbf{B}_{1,k}=\mathbf{B}_k$ for $k\in[3]$. The answer returned by the s^{th} server to the user is

$$Y_s = \widetilde{A}^s\widetilde{B}^s. \tag{6.71}$$

Now let us prove that the user is able to recover desired products $(\mathbf{A}_l\mathbf{B}_l)_{l\in[3]}=(\mathbf{A}_{1,k}\mathbf{B}_{1,k})_{k\in[3]}$ with recovery threshold $R=(\ell+1)K_c-1=2\times 3-1=5$. Let us rewrite Y_s as follows.

$$\begin{aligned}Y_s &= \widetilde{A}^s\widetilde{B}^s \qquad (6.72)\\ &= \frac{(f_{1,2}-\alpha_s)(f_{1,3}-\alpha_s)}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}\mathbf{B}_{1,1}+\frac{(f_{1,1}-\alpha_s)(f_{1,3}-\alpha_s)}{f_{1,2}-\alpha_s}\mathbf{A}_{1,2}\mathbf{B}_{1,2}\\ &\quad+\frac{(f_{1,1}-\alpha_s)(f_{1,2}-\alpha_s)}{f_{1,3}-\alpha_s}\mathbf{A}_{1,3}\mathbf{B}_{1,3}+(f_{1,1}-\alpha_s)(\mathbf{A}_{1,2}\mathbf{B}_{1,3}+\mathbf{A}_{1,3}\mathbf{B}_{1,2})\\ &\quad+(f_{1,2}-\alpha_s)(\mathbf{A}_{1,1}\mathbf{B}_{1,3}+\mathbf{A}_{1,3}\mathbf{B}_{1,1})+(f_{1,3}-\alpha_s)(\mathbf{A}_{1,1}\mathbf{B}_{1,2}+\mathbf{A}_{1,2}\mathbf{B}_{1,1}). \qquad (6.73)\end{aligned}$$

Next let us manipulate the first term on the RHS. By long division of polynomials (regard numerator and denominator as polynomials of α_s), we have

$$\begin{aligned}&\frac{(f_{1,2}-\alpha_s)(f_{1,3}-\alpha_s)}{f_{1,1}-\alpha_s}\mathbf{A}_{1,1}\mathbf{B}_{1,1} \qquad (6.74)\\ &=\left(-\alpha_s+(f_{1,2}+f_{1,3}-f_{1,1})+\frac{(f_{1,2}-f_{1,1})(f_{1,3}-f_{1,1})}{f_{1,1}-\alpha_s}\right)\mathbf{A}_{1,1}\mathbf{B}_{1,1}. \qquad (6.75)\end{aligned}$$

Now it is obvious that the scaling factor of $\mathbf{A}_{1,1}\mathbf{B}_{1,1}$ can be expanded into weighted sums of the terms $(f_{1,1}-\alpha_s)^{-1}$,1 and α_s. For the second and third terms in (6.73), by the long division of polynomials, we can similarly show that the second term can be expanded into weighted sums of the terms $(f_{1,2}-\alpha_s)^{-1}$,1 and α_s and that the third term can be expanded into weighted sums of the terms $(f_{1,3}-\alpha_s)^{-1}$,1 and α_s. Note that the last three terms in (6.73) can be expanded into weighted sums of the terms $1,\alpha_s$. Now, consider any $R=5$ servers, whose indices are denoted as $s_i, i \in [5]$, and we can represent their answers in the following matrix notation.

$$\begin{bmatrix} Y_{s_1} \\ Y_{s_2} \\ Y_{s_3} \\ Y_{s_4} \\ Y_{s_5} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_{1,1}-\alpha_{s_1}} & \frac{1}{f_{1,2}-\alpha_{s_1}} & \frac{1}{f_{1,3}-\alpha_{s_1}} & 1 & \alpha_1 \\ \frac{1}{f_{1,1}-\alpha_{s_2}} & \frac{1}{f_{1,2}-\alpha_{s_2}} & \frac{1}{f_{1,3}-\alpha_{s_2}} & 1 & \alpha_2 \\ \frac{1}{f_{1,1}-\alpha_{s_3}} & \frac{1}{f_{1,2}-\alpha_{s_3}} & \frac{1}{f_{1,3}-\alpha_{s_3}} & 1 & \alpha_3 \\ \frac{1}{f_{1,1}-\alpha_{s_4}} & \frac{1}{f_{1,2}-\alpha_{s_4}} & \frac{1}{f_{1,3}-\alpha_{s_4}} & 1 & \alpha_4 \\ \frac{1}{f_{1,1}-\alpha_{s_5}} & \frac{1}{f_{1,2}-\alpha_{s_5}} & \frac{1}{f_{1,3}-\alpha_{s_5}} & 1 & \alpha_5 \end{bmatrix}}_{\mathbf{V}_{1,3,5}} \underbrace{\begin{bmatrix} c_{1,1} & & & & \\ & c_{1,2} & & & \\ & & c_{1,3} & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}}_{\mathbf{V}'_{1,3,5}} \otimes \mathbf{I}_\lambda \begin{bmatrix} \mathbf{A}_{1,1}\mathbf{B}_{1,1} \\ \mathbf{A}_{1,2}\mathbf{B}_{1,2} \\ \mathbf{A}_{1,3}\mathbf{B}_{1,3} \\ * \\ * \end{bmatrix}, \tag{6.76}$$

where we have used $*$ to represent various combinations of interference symbols that can be found explicitly by expanding (6.73), since those forms are not important. We have $c_{1,1}=(f_{1,2}-f_{1,1})(f_{1,3}-f_{1,1})$, $c_{1,2}=(f_{1,1}-f_{1,2})(f_{1,3}-f_{1,2})$ and $c_{1,3}=(f_{1,1}-f_{1,3})(f_{1,2}-f_{1,3})$. Since $f_{1,1}, f_{1,2}$ and $f_{1,3}$ are distinct by definition, it follows that $c_{1,1}$, $c_{1,2}$ and $c_{1,3}$ are non-zero values. Therefore, the matrix $(\mathbf{V}_{1,3,5}\mathbf{V}'_{1,3,5})\otimes\mathbf{I}_\lambda$ is invertible according to Lemma 2.5 and the properties of Kronecker products. Thus, the user is able to recover the desired matrix products by inverting the matrix $(\mathbf{V}_{1,3,5}\mathbf{V}'_{1,3,5})\otimes\mathbf{I}_\lambda$. This completes the proof of $R=5$ recovery threshold. Similarly, we can compute the upload cost and download cost of the code as follows, $U_A=U_B=S/K_c=S/3$, and $D=5/3$, which achieves desired costs.

Arbitrary ℓ, K_c and $L = \ell K_c$

Now let us present the general code construction. Let $f_{1,1}, f_{1,2}, \cdots, f_{\ell,K_c}, \alpha_1, \alpha_2, \cdots, \alpha_S$ represent $(S+L)$ distinct elements from $\mathbb{F}$. For all $l \in [\ell], s \in [S]$, let us define

$$\Delta_s^{,K_c} = \prod_{k \in [K_c]} (f_{l,k} - \alpha_s). \tag{6.77}$$

Let us also define

$$\mathbf{A}_{l,k} = \mathbf{A}_{K_c(l-1)+k}, \tag{6.78}$$

$$\mathbf{B}_{l,k} = \mathbf{B}_{K_c(l-1)+k}, \tag{6.79}$$

for all $l \in [\ell], k \in [K_c]$. Note that by this definition, desired products can be represented as follows.

$$\begin{pmatrix} \mathbf{A}_{1,1}\mathbf{B}_{1,1} & \cdots & \mathbf{A}_{1,K_c}\mathbf{B}_{1,K_c} \\ \mathbf{A}_{2,1}\mathbf{B}_{2,1} & \cdots & \mathbf{A}_{2,K_c}\mathbf{B}_{2,K_c} \\ \vdots & \vdots & \vdots \\ \mathbf{A}_{\ell,1}\mathbf{B}_{\ell,1} & \cdots & \mathbf{A}_{\ell,K_c}\mathbf{B}_{\ell,K_c} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1\mathbf{B}_1 & \cdots & \mathbf{A}_{K_c}\mathbf{B}_{K_c} \\ \mathbf{A}_{K_c+1}\mathbf{B}_{K_c+1} & \cdots & \mathbf{A}_{2K_c}\mathbf{B}_{2K_c} \\ \vdots & \vdots & \vdots \\ \mathbf{A}_{(\ell-1)K_c+1}\mathbf{B}_{(\ell-1)K_c+1} & \cdots & \mathbf{A}_{\ell K_c}\mathbf{B}_{\ell K_c} \end{pmatrix}. \tag{6.80}$$

Now we are ready to construct the *CSA code* with arbitrary parameters (ℓ, K_c). For all $s \in [S]$, let us construct shares of matrices $\mathbf{A}$ and $\mathbf{B}$ at the s^{th} server as follows.

$$\widetilde{A}^s = (\widetilde{A}_1^s, \widetilde{A}_2^s, \ldots, \widetilde{A}_\ell^s), \tag{6.81}$$

$$\widetilde{B}^s = (\widetilde{B}_1^s, \widetilde{B}_2^s, \ldots, \widetilde{B}_\ell^s), \tag{6.82}$$

where for $l \in [\ell]$, let us set

$$\widetilde{A}_l^s = \Delta_s^{l,K_c} \sum_{k \in [K_c]} \frac{1}{f_{l,k} - \alpha_s} \mathbf{A}_{l,k}, \tag{6.83}$$

$$\widetilde{B}_l^s = \sum_{k\in[K_c]} \frac{1}{f_{l,k}-\alpha_s}\mathbf{B}_{l,k}. \tag{6.84}$$

The answer returned by the s^{th} server to the user is constructed as follows.

$$\begin{aligned} Y_s &= \sum_{l\in[\ell]} \widetilde{A}_l^s \widetilde{B}_l^s && (6.85)\\ &= \widetilde{A}_1^s \widetilde{B}_1^s + \widetilde{A}_2^s \widetilde{B}_2^s + \cdots + \widetilde{A}_\ell^s \widetilde{B}_\ell^s. && (6.86) \end{aligned}$$

Now let us see why the $R = (\ell+1)K_c - 1$ recovery threshold holds. First, let us rewrite Y_s as follows.

$$\begin{aligned} Y_s &= \widetilde{A}_1^s \widetilde{B}_1^s + \widetilde{A}_2^s \widetilde{B}_2^s + \cdots + \widetilde{A}_\ell^s \widetilde{B}_\ell^s && (6.87)\\ &= \sum_{l\in[\ell]} \Delta_s^{l,K_c} \left(\sum_{k\in[K_c]} \frac{1}{f_{l,k}-\alpha_s}\mathbf{A}_{l,k} \right) \left(\sum_{k\in[K_c]} \frac{1}{f_{l,k}-\alpha_s}\mathbf{B}_{l,k} \right) && (6.88)\\ &= \sum_{l\in[\ell]} \Delta_s^{l,K_c} \left(\sum_{k\in[K_c]} \sum_{k'\in[K_c]} \frac{\mathbf{A}_{l,k}\mathbf{B}_{l,k'}}{(f_{l,k}-\alpha_s)(f_{l,k'}-\alpha_s)} \right) && (6.89)\\ &= \sum_{l\in[\ell]} \sum_{k\in[K_c]} \frac{\prod_{k'\in[K_c]\setminus\{k\}}(f_{l,k'}-\alpha_s)}{(f_{l,k}-\alpha_s)} \mathbf{A}_{l,k}\mathbf{B}_{l,k} \\ &\qquad + \sum_{l\in[\ell]} \sum_{\substack{k,k'\in[K_c]\\ k\neq k'}} \left(\prod_{k''\in[K_c]\setminus\{k,k'\}} (f_{l,k''}-\alpha_s) \right) \mathbf{A}_{l,k}\mathbf{B}_{l,k'}, && (6.90) \end{aligned}$$

where in the last step, we split the summation into two parts depending on whether or not $k = k'$.

Let us consider the first term in (6.90). If we regard both numerator and denominator as polynomials of α_s, then by long division of polynomials, for each $l \in [\ell], k \in [K_c]$, the following term

$$\frac{\prod_{k'\in[K_c]\setminus\{k\}}(f_{l,k'}-\alpha_s)}{(f_{l,k}-\alpha_s)} \mathbf{A}_{l,k}\mathbf{B}_{l,k}, \tag{6.91}$$

can be expanded into weighted sums of the terms $(f_{l,k}-\alpha_s)^{-1}, 1, \alpha_s, \cdots, \alpha_s^{K_c-2}$, i.e., it can be rewritten as

$$\Big(c_{-1}(f_{l,k}-\alpha_s)^{-1}+c_0+c_1\alpha_s+\cdots+c_{K_c-2}\alpha_s^{K_c-2}\Big)\mathbf{A}_{l,k}\mathbf{B}_{l,k}. \tag{6.92}$$

Now note that the numerator polynomial $\prod_{k'\in[K_c]\backslash\{k\}}(f_{l,k'}-\alpha_s)$ has no root $f_{l,k}$, while $f_{l,k}$ is the only root of the denominator polynomial. Since $(f_{l,k})_{l\in[\ell],k\in[K_c]}$ are distinct elements from $\mathbb{F}$ by definition, by the polynomial remainder theorem, $c_{-1}=\prod_{k'\in[K_c]\backslash\{k\}}(f_{l,k'}-f_{l,k})\neq 0$.

Next we note that the second term in (6.90) can be expanded[10] into weighted sums of the terms $1, \alpha_s, \cdots, \alpha_s^{K_c-2}$, so in the matrix form, answers from any $R=(\ell+1)K_c-1$ servers, whose indices are denoted as $s_1, s_2, \cdots, s_R$, can be written as follows.

$$\begin{bmatrix} Y_{s_1} \\ Y_{s_2} \\ \vdots \\ Y_{s_R} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{f_{1,1}-\alpha_{s_1}} & \frac{1}{f_{1,2}-\alpha_{s_1}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_1}} & 1 & \alpha_{s_1} & \cdots & \alpha_{s_1}^{R-L-1} \\ \frac{1}{f_{1,1}-\alpha_{s_2}} & \frac{1}{f_{1,2}-\alpha_{s_2}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_2}} & 1 & \alpha_{s_2} & \cdots & \alpha_{s_2}^{R-L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_{1,1}-\alpha_{s_R}} & \frac{1}{f_{1,2}-\alpha_{s_R}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_R}} & 1 & \alpha_{s_R} & \cdots & \alpha_{s_R}^{R-L-1} \end{bmatrix}}_{\mathbf{V}_{\ell,K_c,R}}$$

$$\underbrace{\begin{bmatrix} c_{1,1} & & & & & & \\ & c_{1,2} & & & & & \\ & & \ddots & & & & \\ & & & c_{\ell,K_c} & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}}_{\mathbf{V}'_{\ell,K_c,R}} \otimes \mathbf{I}_\lambda \begin{bmatrix} \mathbf{A}_{1,1}\mathbf{B}_{1,1} \\ \mathbf{A}_{1,2}\mathbf{B}_{1,2} \\ \vdots \\ \mathbf{A}_{\ell,K_c}\mathbf{B}_{\ell,K_c} \\ * \\ \vdots \\ * \end{bmatrix}, \tag{6.93}$$

[10] When $K_c=1$, the second term in (6.90) equal zero, thus the Vandermonde terms do not appear and the matrix form representation only involves Cauchy matrices, i.e., Cauchy-Vandermonde matrices without Vandermonde part.

where we have used $*$ to represent various combinations of interference symbols that can be found explicitly by expanding (6.90), whose exact forms are irrelevant. We note that $R-L-1=(\ell+1)K_c-1-\ell K_c-1=K_c-2$. And we also note that for all $l\in[\ell]$ and $k\in[K_c]$, $c_{l,k}=\prod_{k'\in[K_c]\setminus\{k\}}(f_{l,k'}-f_{l,k})\neq 0$. Therefore, guaranteed by Lemma 2.5 and the fact that the Kronecker product of non-singular matrices is non-singular, the matrix $(\mathbf{V}_{\ell,K_c,R}\mathbf{V}'_{\ell,K_c,R})\otimes\mathbf{I}_\lambda$ is invertible. Therefore, the user is able to recover desired products $(\mathbf{A}_{l,k}\mathbf{B}_{l,k})_{l\in[\ell],k\in[K_c]}$ by inverting the matrix. This completes the proof of $R=(\ell+1)K_c-1$ recovery threshold. For the upload costs, it is easy to see that we have $U_A=U_B=(\ell S)/L=S/K_c$. The download cost is $D=R/L=((\ell+1)K_c-1)/(\ell K_c)$. The computational complexity at each server is $\mathcal{O}(\lambda\kappa\mu/K_c)$ if we assume straightforward matrix multiplication algorithms.

Finally, let us consider the encoding and decoding complexity. Recall the encoding functions (6.81), (6.82), (6.83), (6.84). Note that each of the $\widetilde{A}_\ell^s$ can be regarded as products of an $S\times K_c$ Cauchy matrix with a total of $\lambda\kappa$ column vectors of length K_c. Similarly, each of the $\widetilde{B}_\ell^s$ can be considered as products of an $S\times K_c$ Cauchy matrix by a total of $\kappa\mu$ column vectors of length K_c. Remarkably, the problem of efficiently multiplying an $S\times S$ Cauchy matrix with a column vector is known as Trummer's problem[42]. Fast algorithms exist [39, 38, 84] that solve Trummer's problem with computational complexity as low as $\widetilde{\mathcal{O}}(S\log^2 S)$, in contrast to straightforward algorithms that have computational complexity of $\mathcal{O}(S^2)$. Similarly, with fast algorithms the computational complexity of multiplying a $S\times K_c$ Cauchy matrix with a column vector is at most $\widetilde{\mathcal{O}}(S\log^2 S)$, so the encoding complexity of $\widetilde{A}^{[S]}$ and $\widetilde{B}^{[S]}$ is at most $\widetilde{\mathcal{O}}\big(\big(\lambda\kappa S\log^2 S\big)/K_c\big)$ and $\widetilde{\mathcal{O}}\big(\big(\kappa\mu S\log^2 S\big)/K_c\big)$, respectively. On the other hand, consider the decoding procedure of *CSA codes*, which can be regarded as solving a total of $\lambda\mu$ linear systems defined by an $R\times R$ coefficient matrix. Indeed, this coefficient matrix is a Cauchy-Vandermonde matrix. There is a large body of literature studying fast algorithms for solving linear systems defined by $R\times R$ Cauchy-Vandermonde matrices, and the best known computational complexity is $\widetilde{\mathcal{O}}(R\log^2 R)$, see, e.g., [35][11]. Therefore, the decoding complexity of $\widetilde{\mathcal{O}}\big((\lambda\mu R\log^2 R)/L\big)=\widetilde{\mathcal{O}}(\lambda\mu\log^2 R)$ is achievable. This completes the proof of Theorem 6.1.

6.4.4 Systematic Construction of CSA Codes

In this section, we present a systematic construction of CSA codes. Instead of uploading coded version of matrices $\mathbf{A}$ and $\mathbf{B}$ to all of the S servers, the systematic construction of CSA codes uploads uncoded constituent matrix pair $(\mathbf{A}_s,\mathbf{B}_s)$ directly to the s^{th} server for the first L servers, i.e., for all $s\in[L]$. For the remaining $S-L$ servers, coded shares are uploaded following the same construction that was presented in Section 6.4.3. We will see that the systematic construction of CSA codes works on a smaller field $\mathbb{F}$, compared to the construction presented in Section 6.4.3. The systematic construction of CSA codes requires less encoding complexity. Its decoding complexity decreases as more of the first L servers respond. In fact, if all of the first L servers respond, then no computation is required at all for decoding. The systematic construction also preserves backward compatibility to current systems that apply straightforward parallelization strategies. Formally, we have

$$\widetilde{A}^s=\mathbf{A}_s, \tag{6.94}$$

$$\widetilde{B}^s=\mathbf{B}_s \tag{6.95}$$

for all $s\in[L]$ and

$$\widetilde{A}^s=(\widetilde{A}_1^s,\widetilde{A}_2^s,\ldots,\widetilde{A}_\ell^s), \tag{6.96}$$

[11] The fast algorithm of solving Cauchy-Vandermonde type linear systems here takes inputs of only parameters of a Cauchy-Vandermonde matrix $\mathbf{V}$, i.e, $(\alpha_{s_1},\alpha_{s_2},\cdots,\alpha_{s_R},f_{1,1},f_{1,2},\cdots,f_{\ell,K_c})$ and a column vector $\mathbf{y}$, and outputs the column vector $\mathbf{x}$ such that $\mathbf{V}\mathbf{x}=\mathbf{y}$ with the computational complexity of at most $\mathcal{O}(R\log^2 R)$. Therefore, it is not necessary for the user (decoder) to store extra information beyond $\alpha_{[S]}$ and $(f_{l,k})_{l\in[\ell],k\in[K_c]}$.

$$\widetilde{B}^s = (\widetilde{B}_1^s, \widetilde{B}_2^s, \ldots, \widetilde{B}_\ell^s) \tag{6.97}$$

for all $s \in \{L+1, \cdots, S\}$, where $\widetilde{A}_{[\ell]}^s$ and $\widetilde{B}_{[\ell]}^s$ are defined in (6.83) and (6.84) respectively. Similarly, the answer returned by the s^{th} server is constructed as follows.

$$Y_s = \widetilde{A}^s \widetilde{B}^s \tag{6.98}$$

for all $s \in [L]$ and

$$Y_s = \sum_{l \in [\ell]} \widetilde{A}_l^s \widetilde{B}_l^s \tag{6.99}$$

for all $s \in \{L+1, \cdots, S\}$. Note that since coded shares are used only for $S-L$ servers, we no longer need distinct values $\alpha_1, \alpha_2, \cdots, \alpha_L$, so the field size required is only $|\mathbb{F}| \geq S$.

Now, let us prove that the recovery threshold R is not affected by the systematic construction, i.e., the desired products are still recoverable from the answers of any $R = L + K_c - 1$ servers. Denote the set of responsive servers as $\mathcal{R}$, $|\mathcal{R}| = R$. Note that if $[L] \subset \mathcal{R}$, then the desired products $\mathbf{AB}$ can be directly recovered from answers of the first L servers. On the other hand, if $[L] \cap \mathcal{R} = \emptyset$, then we can recover the desired products $\mathbf{AB}$ following the same argument that was presented in Section 6.4.3. When $[L] \cap \mathcal{R} \neq \emptyset$, denote the elements in the set $\mathcal{R} \setminus [L]$ as $(s_1, s_2, \cdots, s_{R'})$. The answers from these R' servers can be written in the following matrix form.

$$\begin{bmatrix} Y_{s_1} \\ Y_{s_2} \\ \vdots \\ Y_{s_{R'}} \end{bmatrix} = \begin{bmatrix} \frac{c_{1,1}}{f_{1,1}-\alpha_{s_1}} & \frac{c_{1,2}}{f_{1,2}-\alpha_{s_1}} & \cdots & \frac{c_{\ell,K_c}}{f_{\ell,K_c}-\alpha_{s_1}} & 1 & \alpha_{s_1} & \cdots & \alpha_{s_1}^{R-L-1} \\ \frac{c_{1,1}}{f_{1,1}-\alpha_{s_2}} & \frac{c_{1,2}}{f_{1,2}-\alpha_{s_2}} & \cdots & \frac{c_{\ell,K_c}}{f_{\ell,K_c}-\alpha_{s_2}} & 1 & \alpha_{s_2} & \cdots & \alpha_{s_2}^{R-L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{c_{1,1}}{f_{1,1}-\alpha_{s_{R'}}} & \frac{c_{1,2}}{f_{1,2}-\alpha_{s_{R'}}} & \cdots & \frac{c_{\ell,K_c}}{f_{\ell,K_c}-\alpha_{s_{R'}}} & 1 & \alpha_{s_R} & \cdots & \alpha_{s_{R'}}^{R-L-1} \end{bmatrix} \otimes \mathbf{I}_\lambda \begin{bmatrix} \mathbf{A}_{1,1}\mathbf{B}_{1,1} \\ \mathbf{A}_{1,2}\mathbf{B}_{1,2} \\ \vdots \\ \mathbf{A}_{\ell,K_c}\mathbf{B}_{\ell,K_c} \\ * \\ \vdots \\ * \end{bmatrix}, \tag{6.100}$$

Note that the dimension of the first matrix on the RHS, or the decoding matrix, is $(R' \times R)$, thus it appears to be not invertible. However, from answers of the servers in the set $\mathcal{R} \cap [L]$, we can directly recover $|\mathcal{R} \cap [L]|$ desired products. Note that the desired products appear along the dimension spanned by the Cauchy part. By subtracting these known products from the answers, we obtain the decoding matrix of dimension $(R' \times R')$, which is invertible by Lemma 2.5. This completes the proof of recovery threshold $R = L + K_c - 1$. It is easy to see that the upload and download costs are also not affected by the systematic construction. For the encoding complexity, the systematic construction requires less arithmetic operations because no computation is needed to obtain $\widetilde{A}^{[L]}$ and $\widetilde{B}^{[L]}$. For the decoding complexity, when $[L] \subset \mathcal{R}$, no computation is needed, and when $[L] \cap \mathcal{R} = \emptyset$, it follows the same argument presented in Section 6.4.3. When $[L] \cap \mathcal{R} \neq \emptyset$, the user (decoder) eliminates all products obtained from answers of the servers in the set $\mathcal{R} \cap [L]$, and then decodes the remaining products according to the fast decoding algorithm. Thus the decoding complexity is not increased.

6.5 Generalized Cross-Subspace Alignment (GCSA) Codes: Combining Batch Processing and Matrix-Partitioning

In this section, we present Generalized CSA codes (GCSA codes), which combine the batch processing of CSA codes with the matrix-partitioning approach of EP codes. Although we have shown that batch processing with CSA codes significantly improves the tradeoff between upload-download costs, evidently CSA codes require at least one matrix multiplication of dimensions $\lambda\times\kappa$ and $\kappa\times\mu$ at each server. For a computation-latency limited setting, partitioning may be necessary to reduce the computation load per server. A naive approach to combine the batch processing of CSA codes and the partitioning of EP codes is to first (separately for each $l\in[L]$) apply EP codes for each pair of matrices, $\mathbf{A}_l,\mathbf{B}_l$. Next, since only matrix multiplication is involved in obtaining the answers for EP codes, we can then apply CSA codes for these matrix multiplications. Specifically, for positive integers ℓ',K_c',S',p,m,n such that $L=\ell'K_c'$, $m\mid\lambda$, $n\mid\mu$, $p\mid\kappa$ and $S'\geq pmn+p-1$, apply EP codes of parameter p,m,n with S' servers for each matrix multiplication. This yields a total of $S'\ell'K_c'$ matrix multiplications. For these matrix multiplications, we can further apply CSA codes. Now we can see that by this construction, if we choose CSA codes parameters $\ell=\ell',K_c=K_c'S'$, we can achieve the upload cost $(U_A,U_B)=(S/(K_c'pm),S/(K_c'pn))$. It is also easy to see that for this simple combination of EP and CSA codes, the recovery threshold achieved is $R=\ell'K_c'S'+K_c'S'-1$, and the download cost is $D=(\ell'K_c'S'+K_c'S'-1)/(\ell'K_c'mn)$. However, we will see that under the same upload cost, GCSA codes can improve the recovery threshold to $R=pmn(\ell'K_c'+K_c'-1)+p-1$ and the download cost to $D=(pmn(\ell'K_c'+K_c'-1)+p-1)/(\ell'K_c'mn)$. This result is better than the naive construction because $S'\geq pmn+p-1\geq 1$. On the other hand, note that the Lagrange Coded Computation (LCC) codes in [138] can be regarded as a special case of CSA codes with parameter $\ell=1$, so the naive approach of combining LCC codes with EP codes achieves the recovery threshold of $R=2LS'-1$. With GCSA codes of parameter $\ell=1$, the recovery threshold is improved to $R=2Lpmn+p-1$.

6.5.1 GCSA Codes: Main Result

Our main result for GCSA codes appears in the following theorem.

Theorem 6.2. *For CDBMM over a field $\mathbb{F}$ with S servers, and positive integers (ℓ,K_c,p,m,n) such that $m\mid\lambda$, $n\mid\mu$, $p\mid\kappa$ and $L=\ell K_c\leq|\mathbb{F}|-S$, the GCSA codes presented in this chapter achieve*

$$\begin{aligned}
&\textit{Recovery Threshold:} && R=pmn((\ell+1)K_c-1)+p-1, && (6.101)\\
&\textit{Upload Cost for } \widetilde{A}^{[S]},\widetilde{B}^{[S]}\textit{:} && (U_A,U_B)=\left(\frac{S}{K_cpm},\frac{S}{K_cpn}\right), && (6.102)\\
&\textit{Download Cost:} && D=\frac{pmn((\ell+1)K_c-1)+p-1}{mn\ell K_c}, && (6.103)\\
&\textit{Server Computation Complexity:} && \mathcal{C}_s=\mathcal{O}\left(\frac{\lambda\kappa\mu}{K_cpmn}\right), && (6.104)\\
&\textit{Encoding Complexity for } \widetilde{A}^{[S]},\widetilde{B}^{[S]}\textit{:} && (\mathcal{C}_{eA},\mathcal{C}_{eB})=\left(\widetilde{\mathcal{O}}\left(\frac{\lambda\kappa S\log^2 S}{K_cpm}\right),\widetilde{\mathcal{O}}\left(\frac{\kappa\mu S\log^2 S}{K_cpn}\right)\right) && (6.105)\\
&\textit{Decoding Complexity:} && \mathcal{C}_d=\widetilde{\mathcal{O}}\left(\lambda\mu p\log^2 R\right). && (6.106)
\end{aligned}$$

6.5.2 Observations

1. GCSA codes generalize almost all state of art approaches for coded distributed batch matrix multiplication. Setting $m=n=p$ reduces GCSA codes to CSA codes. Further setting $\ell=1$ recovers LCC codes. Setting $\ell=K_c=1$ reduces GCSA codes to EP codes. Further setting $p=1$ recovers Polynomial codes, while setting $m=n=1$ recovers Mat-

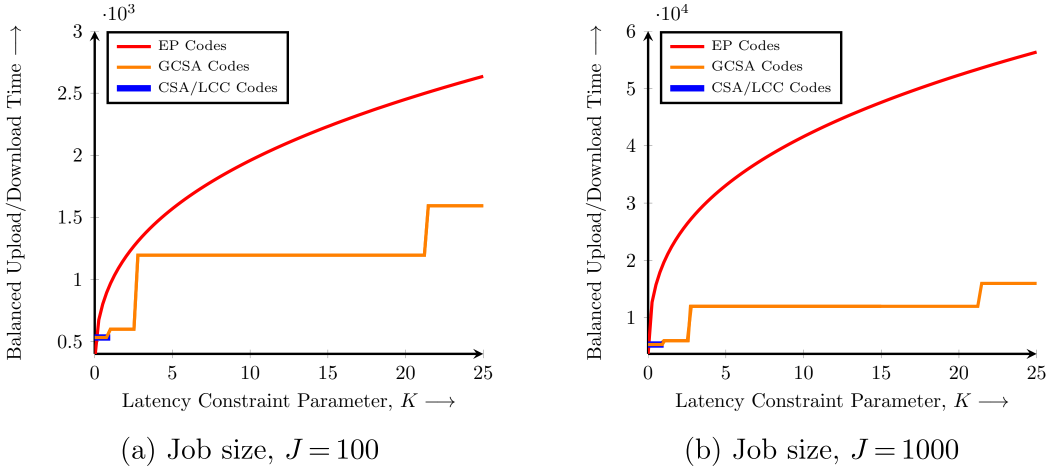
Dot codes.

2. Let us explain why GCSA codes, which include CSA codes, LCC codes and EP codes as special cases, are capable of achieving more than what each of these codes can achieve in general. Consider a finite horizon setting, where[12] the *job size* J, i.e., the number of matrices to be multiplied, is fixed. So we need to compute J matrix multiplications, $\mathbf{A}_1\mathbf{B}_1,\cdots,\mathbf{A}_J\mathbf{B}_J$, where each $\mathbf{A}_j,\mathbf{B}_j,j\in[J]$ is a $\lambda\times\lambda$ matrix. Suppose each scalar multiplication takes T_m seconds, and it takes T_c seconds to communicate one scalar over any communication channel. For simplicity let us assume that multiplying two $\lambda\times\lambda$ matrices requires λ^3T_m seconds of computation time. There is a required latency constraint for this job, such that the total computation time at each server cannot exceed λ^3T_m/K, where $K>1$ is a given parameter that determines the server latency constraint. Note that this latency constraint immediately rules out LCC codes, and even CSA codes because they need at least λ^3T_m seconds of computation time at each server which violates the given constraint. Now consider EP codes which can partition the matrices to reduce the size of computation task at each server, but need to repeat the process J times because each $\mathbf{A}_j\mathbf{B}_j$ is computed separately. EP codes need computation time $J\lambda^3T_m/(pmn)$ at each server, so they can satisfy the latency constraint by choosing $pmn\geq JK$. On the other hand, GCSA codes with, say $\ell=1$ and $K_c=J$, need computation time $\lambda^3T_m/(p''m''n'')$ at each server. So GCSA codes can satisfy the latency constraint by choosing $p''m''n''\geq K$, i.e., with less partitioning than needed for EP codes. Note that GCSA codes need less partitioning than EP codes to satisfy the same latency constraint, because they make up some of the computation time by batch processing of the J multiplications that must be carried out separately by EP codes. It turns out that this allows GCSA codes to have lower communication cost. EP codes require a total download time of $\frac{JR_{EP}\lambda^2T_c}{mn}=\frac{J(pmn+p-1)\lambda^2T_c}{mn}$,

[12] We make a distinction between batch size and job size, in that the job size is fixed as part of the problem specification while the batch size may be chosen arbitrarily by a coding scheme, e.g., to partition the job into smaller jobs.

and an upload time of $\frac{JS\lambda^2T_c}{pm}+\frac{JS\lambda^2T_c}{pn}$ seconds, where S is the number of servers utilized by the scheme. For straggler tolerance, suppose $R_{EP}/S=\eta<1$, so that the upload time is expressed as $\frac{J(pmn+p-1)\lambda^2T_c}{\eta p}(\frac{1}{m}+\frac{1}{n})$. For balanced upload and download times we need $m=n$ and $\eta pm/2\approx m^2$, so that the balanced upload/download time for EP codes is $\approx\frac{J(2m^3/\eta+2m/\eta-1)\lambda^2T_c}{m^2}$. Given the latency constraint which forces $2m^3/\eta\geq JK$, we find that the optimal balanced upload/download time for EP codes is achieved with $m\approx\left(\frac{\eta JK}{2}\right)^{1/3}$. On the other hand, now consider GCSA codes, which need total download time of $\frac{R_{GCSA}\lambda^2T_c}{m''n''}=\frac{(p''m''n''(2J-1)+p''-1)\lambda^2T_c}{m''n''}$, and total upload time of $\frac{S''\lambda^2T_c}{p''m''}+\frac{S''\lambda^2T_c}{p''n''}$. For similar straggler robustness, let $R_{GCSA}/S''=\eta$ be the same as for EP codes. For balanced costs, similarly set $m''=n''$ and $\eta p''m''/2\approx m''^2$. Thus GCSA codes achieve balanced upload/download time of $\approx\frac{((2m''^3/\eta)(2J-1)+2m''/\eta-1)\lambda^2T_c}{m''^2}$, respectively. Combined with the latency constraint which forces $2m''^3/\eta\geq K$, we find that the optimal balanced upload/download time for this particular construction of GCSA codes is achieved with $m''\approx\left(\frac{\eta K}{2}\right)^{1/3}$. For a quick comparison of approximately optimal balanced upload/download time, note that for EP codes it is lower bounded by $\frac{2Jm\lambda^2T_c}{\eta}$, and for GCSA codes it is upper bounded by $\frac{8Jm''\lambda^2T_c}{\eta}$, so EP codes need more balanced communication time by a factor of at least $\frac{m}{4m''}\approx\frac{J^{1/3}}{4}$ which can be significantly larger than 1 for large job sizes J. To complement the approximate analysis, Figure 6.3 explicitly compares the balanced upload/download time (the maximum of upload and download times) versus the job latency constraint parameter K. The values shown for EP codes are precisely lowerbounds, i.e., EP codes cannot do any better, while those for GCSA codes are strictly achievable. Thus, GCSA codes can achieve more than what can be achieved by CSA, LCC or EP codes.

3. The finite horizon, i.e., fixed job size and fixed latency constraint for each job is important in the previous discussion. If instead of the absolute value of server latency for a fixed job size, we only insisted on normalized server latency *per job*, where each job is still comprised of J matrix multiplications, then we could jointly process K jobs



(a) Job size, $J = 100$

(b) Job size, $J = 1000$

Figure 6.3: Balanced upload/download time vs the value of the latency constraint parameter K for EP codes, CSA/LCC codes and GCSA codes (normalized by $\lambda^3 T_c$). CSA/LCC codes are not feasible for $K > 1$. The values for EP codes are lower bounds while those for GCSA codes are upper bounds, showing that GCSA codes strictly outperform both batch processing (CSA/LCC) and matrix-partitioning (EP) codes.

codes with an absolute latency of $\lambda^3 T_c$ which allows LCC and CSA codes, while still achieving the latency *per job* of λ^3/K. Since batch processing approaches like LCC codes and CSA codes have already been shown to achieve better communication costs than any matrix partitioning approach, neither EP codes nor GCSA codes would be needed in that case. On the other hand, it is also worth noting that if we go to the other extreme and require a fixed server latency of less than $\lambda^3 T_c/K$ for each matrix multiplication, i.e., set $J = 1$, then it can be seen that batch processing cannot help, i.e., matrix partitioning alone is enough. In other words, if latency constraints are imposed on each matrix multiplication $(J = 1)$, then EP codes suffice, and neither LCC codes, nor CSA or GCSA codes are needed.

4. Figure 6.3 shows the advantage of GCSA codes in terms of communication cost over exclusively batch processing (LCC) and matrix-partitioning (EP) codes under absolute server latency constraints, even when GCSA codes are restricted to the choice $\ell = 1$. However, the advantage of GCSA codes over LCC and EP codes can be seen even without absolute latency constraints. This is illustrated in Figure 6.4 which only

constrains the recovery threshold R and the number of servers S. The figure shows lower convex hulls of achievable (balanced upload cost, download cost) pairs of GCSA codes for various bounds on the matrix partitioning parameters pmn, given that the number of servers $S=300$ and the overall recovery threshold $R \leq 250$. Each value of (S,R,pmn) produces an achievable region in the (U,D) plane (including all possible choices of parameters m,n,p,ℓ,K_c). What is shown in the figure is the union of these regions. The larger the value of pmn, the more the GCSA code construction shifts toward EP codes, generally with the benefit of reduced latency of computation at each server that comes with matrix partitioning. On the other hand, the smaller the value of pmn, the more the GCSA code construction shifts toward CSA codes, with the benefit of improved communication costs that come with batch processing. As noted previously, when no matrix partitioning is allowed, LCC codes can be recovered as a special case of GCSA codes by setting $\ell=1$. The figure also shows how GCSA codes are capable of improving upon LCC codes in terms of download cost by choosing $\ell>1$.

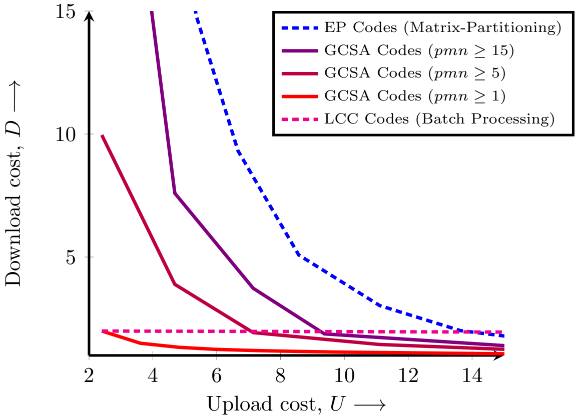


Figure 6.4: Lower convex hulls of achievable (balanced upload cost, download cost) pairs (U,D) of GCSA codes for various bounds on pmn, given that $S=300$ and the overall recovery threshold $R \leq 250$. Note that EP codes and LCC codes are also special cases of GCSA codes, obtained by setting $\ell=K_c=1$, and $\ell=m=n=p=1$, respectively. CSA codes are obtained by setting $m=n=p=1$.

5. While in this chapter we do not explore improvements that are possible by using more efficient matrix multiplication algorithms, it is worthwhile to note that stronger

constructions can indeed be built upon more efficient matrix multiplication algorithms, e.g., Strassen's algorithm. For example, it is recently shown in [137] that the recovery threshold of $2LR(p,m,n)-1$ is achievable where $R(p,m,n)$ is the bilinear complexity for multiplying two matrices of sizes $m\times p$ and $p\times n$. Note that $R(p,m,n)<pmn$. On the other hand, since GCSA codes are built upon straightforward matrix multiplication algorithms, there is a leading factor of pmn in the expression of the recovery threshold of GCSA codes. As a result, the recovery threshold achieved in [137] is order-wise better than GCSA codes for large batch size L. It is notable that for smaller batch sizes, GCSA codes based on straightforward matrix multiplication can still outperform those in [137] that are based on more sophisticated matrix multiplication algorithms. For example, it is known that $R(2,2,2)=7$. Thus if we set $\ell=1$, we have $2LR(2,2,2)-1\geq 8(2L-1)+1$ when $L=1,2,3$.

6.5.3 Proof of Theorem 6.2

Let us recall the standard result for Confluent Cauchy-Vandermonde matrices [37], reproduced here for the sake of completeness.

Lemma 6.1. *If $f_{1,1},f_{1,2},\cdots,f_{\ell,K_c},\alpha_1,\alpha_2,\cdots,\alpha_R$ are $R+L$ distinct elements of $\mathbb{F}$, with $|\mathbb{F}|\geq R+L$ and $L=\ell K_c$, then the following $R\times R$ Confluent Cauchy-Vandermonde matrix is invertible over $\mathbb{F}$.*

$$\hat{\mathbf{V}}_{\ell,K_c,R',R}\triangleq\begin{bmatrix}\frac{1}{(f_{1,1}-\alpha_1)^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_1} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_1)^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_1} & 1 & \cdots & \alpha_1^{R-R'L-1}\\ \frac{1}{(f_{1,1}-\alpha_2)^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_2} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_2)^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_2} & 1 & \cdots & \alpha_2^{R-R'L-1}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \frac{1}{(f_{1,1}-\alpha_R)^{R'}} & \cdots & \frac{1}{f_{1,1}-\alpha_R} & \cdots & \frac{1}{(f_{\ell,K_c}-\alpha_R)^{R'}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_R} & 1 & \cdots & \alpha_R^{R-R'L-1}\end{bmatrix} \tag{6.107}$$

Before presenting the generalized CSA codes construction let us start with an illustrative example.

$\ell = 1, K_c = 2, L = 2, p = 2, m = n = 1$

Let $f_{1,1}, f_{1,2}, \alpha_1, \alpha_2, \ldots, \alpha_S$ represent $(S+2)$ distinct elements from $\mathbb{F}$. For all $s \in [S]$, define,

$$\Delta_s^{1,2} = (f_{1,1} - \alpha_s)^2 (f_{1,2} - \alpha_s)^2. \tag{6.108}$$

We set $\mathbf{A}_{1,1} = \mathbf{A}_1$, $\mathbf{A}_{1,2} = \mathbf{A}_2$, $\mathbf{B}_{1,1} = \mathbf{B}_1$ and $\mathbf{B}_{1,2} = \mathbf{B}_2$. Besides, we partition each of the matrices $\mathbf{A}_{1,1}$ and $\mathbf{A}_{1,2}$ into 1×2 blocks, denoted as $\mathbf{A}_{1,1}^{1,1}$, $\mathbf{A}_{1,1}^{1,2}$ and $\mathbf{A}_{1,2}^{1,1}$, $\mathbf{A}_{1,2}^{1,2}$ respectively. Similarly, we partition each of the matrices $\mathbf{B}_{1,1}$ and $\mathbf{B}_{1,2}$ into 2×1 blocks, denoted as $\mathbf{B}_{1,1}^{1,1}$, $\mathbf{B}_{1,1}^{2,1}$ and $\mathbf{B}_{1,2}^{1,1}$, $\mathbf{B}_{1,2}^{2,1}$ respectively. Note that the desired products $\mathbf{A}_{1,1}\mathbf{B}_{1,1}, \mathbf{A}_{1,2}\mathbf{B}_{1,2}$ can be written as follows.

$$\mathbf{A}_{1,1}\mathbf{B}_{1,1} = \mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{1,1} + \mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{2,1}, \tag{6.109}$$

$$\mathbf{A}_{1,2}\mathbf{B}_{1,2} = \mathbf{A}_{1,2}^{1,1}\mathbf{B}_{1,2}^{1,1} + \mathbf{A}_{1,2}^{1,2}\mathbf{B}_{1,2}^{2,1}. \tag{6.110}$$

Shares of matrices $\mathbf{A}$ are constructed as follows.

$$\widetilde{A}^s = \Delta_s^{1,2} \left(\frac{1}{(f_{1,1} - \alpha_s)^2} \left(\mathbf{A}_{1,1}^{1,1} + (f_{1,1} - \alpha_s) \mathbf{A}_{1,1}^{1,2} \right) + \frac{1}{(f_{1,2} - \alpha_s)^2} \left(\mathbf{A}_{1,2}^{1,1} + (f_{1,2} - \alpha_s) \mathbf{A}_{1,2}^{1,2} \right) \right) \tag{6.111}$$

$$= (f_{1,2} - \alpha_s)^2 \underbrace{\left(\mathbf{A}_{1,1}^{1,1} + (f_{1,1} - \alpha_s) \mathbf{A}_{1,1}^{1,2} \right)}_{P_s^{1,1}} + (f_{1,1} - \alpha_s)^2 \underbrace{\left(\mathbf{A}_{1,2}^{1,1} + (f_{1,2} - \alpha_s) \mathbf{A}_{1,2}^{1,2} \right)}_{P_s^{1,2}}. \tag{6.112}$$

Note that now the term $P_s^{1,1}$ follows the construction of Entangled Polynomial codes of parameter $m = n = 1, p = 2$, and it is a polynomial of $(f_{1,1} - \alpha_s)$. Similarly, the term $P_s^{1,2}$ follows the construction of Entangled Polynomial codes, and it is a polynomial of $(f_{1,2} - \alpha_s)$.

Shares of matrices $\mathbf{B}$ are constructed as follows.

$$\widetilde{B}^s = \frac{1}{(f_{1,1}-\alpha_s)^2}\underbrace{\left((f_{1,1}-\alpha_s)\mathbf{B}_{1,1}^{1,1}+\mathbf{B}_{1,1}^{2,1}\right)}_{Q_s^{1,1}} + \frac{1}{(f_{1,2}-\alpha_s)^2}\underbrace{\left((f_{1,2}-\alpha_s)\mathbf{B}_{1,2}^{1,1}+\mathbf{B}_{1,2}^{2,1}\right)}_{Q_s^{1,2}}. \tag{6.113}$$

The terms $Q_s^{1,1}$ and $Q_s^{1,2}$ also follow the construction of EP codes for the given parameter values p,m,n, and they are polynomials of $(f_{1,1}-\alpha_s)$ and $(f_{1,2}-\alpha_s)$ respectively.

The answer from the s^{th} server, Y_s is constructed as $Y_s = \widetilde{A}^s\widetilde{B}^s$. To see why it is possible to recover the desired products from the answers of any $R=7$ servers, let us rewrite Y_s as follows.

$$\begin{aligned}
Y_s &= \widetilde{A}^s\widetilde{B}^s && (6.114)\\
&= \frac{(f_{1,2}-\alpha_s)^2}{(f_{1,1}-\alpha_s)^2}P_s^{1,1}Q_s^{1,1} + \frac{(f_{1,1}-\alpha_s)^2}{(f_{1,2}-\alpha_s)^2}P_s^{1,2}Q_s^{1,2} + (P_s^{1,1}Q_s^{1,2}+P_s^{1,2}Q_s^{1,1}) && (6.115)\\
&= \frac{((f_{1,1}-\alpha_s)+(f_{1,2}-f_{1,1}))^2}{(f_{1,1}-\alpha_s)^2}P_s^{1,1}Q_s^{1,1} + \frac{((f_{1,2}-\alpha_s)+(f_{1,1}-f_{1,2}))^2}{(f_{1,2}-\alpha_s)^2}P_s^{1,2}Q_s^{1,2}\\
&\quad + (P_s^{1,1}Q_s^{1,2}+P_s^{1,2}Q_s^{1,1}) && (6.116)\\
&= \left(\frac{c_{1,1,0}}{(f_{1,1}-\alpha_s)^2}+\frac{c_{1,1,1}}{f_{1,1}-\alpha_s}\right)P_s^{1,1}Q_s^{1,1} + \left(\frac{c_{1,2,0}}{(f_{1,2}-\alpha_s)^2}+\frac{c_{1,2,1}}{f_{1,2}-\alpha_s}\right)P_s^{1,2}Q_s^{1,2}\\
&\quad + (P_s^{1,1}Q_s^{1,1}+P_s^{1,2}Q_s^{1,2}+P_s^{1,1}Q_s^{1,2}+P_s^{1,2}Q_s^{1,1}), && (6.117)
\end{aligned}$$

where in the last step, we perform binomial expansion for numerator polynomials. According to the Binomial Theorem, $(c_{1,k,i})_{k\in[2],i\in\{0,1\}}$ are non-zero. Note that

$$P_s^{1,1}Q_s^{1,1} = \mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{2,1} + (f_{1,1}-\alpha_s)(\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{1,1}+\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{2,1}) + (f_{1,1}-\alpha_s)^2\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{1,1}, \tag{6.118}$$

$$P_s^{1,2}Q_s^{1,2} = \mathbf{A}_{1,2}^{1,1}\mathbf{B}_{1,2}^{2,1} + (f_{1,2}-\alpha_s)(\mathbf{A}_{1,2}^{1,1}\mathbf{B}_{1,2}^{1,1}+\mathbf{A}_{1,2}^{1,2}\mathbf{B}_{1,2}^{2,1}) + (f_{1,2}-\alpha_s)^2\mathbf{A}_{1,2}^{1,2}\mathbf{B}_{1,2}^{1,1}. \tag{6.119}$$

Therefore, we can further rewrite the first term in (6.117) as follows.

$$\frac{c_{1,1,0}\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{2,1}}{(f_{1,1}-\alpha_s)^2} + \frac{c_{1,1,1}\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{2,1}+c_{1,1,0}(\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{1,1}+\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{2,1})}{f_{1,1}-\alpha_s}$$

$$\begin{aligned}&+(c_{1,1,0}\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{1,1}+c_{1,1,1}(\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{1,1}+\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{2,1}))\\&+(f_{1,1}-\alpha_s)(c_{1,1,1}\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{1,1}).\end{aligned}\tag{6.120}$$

The second term in (6.117) can be similarly rewritten. Note that the third term in (6.117) and the last two terms in (6.120) can be expanded into weighted sums of the terms $1,\alpha_s,\alpha_s^2$, so in the matrix form, answers from any 7 servers, whose indices are denoted as $s_1,s_2,\cdots,s_7$, can be written as follows.

$$\begin{bmatrix}Y_{s_1}\\Y_{s_2}\\\vdots\\Y_{s_7}\end{bmatrix}=\underbrace{\left[\begin{array}{cc|cc|ccc}\frac{1}{(f_{1,1}-\alpha_{s_1})^2}&\frac{1}{f_{1,1}-\alpha_{s_1}}&\frac{1}{(f_{1,2}-\alpha_{s_1})^2}&\frac{1}{f_{1,2}-\alpha_{s_1}}&1&\alpha_{s_1}&\alpha_{s_1}^2\\\frac{1}{(f_{1,1}-\alpha_{s_2})^2}&\frac{1}{f_{1,1}-\alpha_{s_2}}&\frac{1}{(f_{1,2}-\alpha_{s_2})^2}&\frac{1}{f_{1,2}-\alpha_{s_2}}&1&\alpha_{s_2}&\alpha_{s_2}^2\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\\frac{1}{(f_{1,1}-\alpha_{s_7})^2}&\frac{1}{f_{1,1}-\alpha_{s_7}}&\frac{1}{(f_{1,2}-\alpha_{s_7})^2}&\frac{1}{f_{1,2}-\alpha_{s_7}}&1&\alpha_{s_7}&\alpha_{s_7}^2\end{array}\right]}_{\hat{\mathbf{V}}_{1,2,2,7}}$$

$$\underbrace{\left[\begin{array}{c|c|c}\mathbf{T}(c_{1,1,0},c_{1,1,1})&&\\\hline&\mathbf{T}(c_{1,2,0},c_{1,2,1})&\\\hline&&\mathbf{I}_3\end{array}\right]}_{\hat{\mathbf{V}}'_{1,2,2,7}}\otimes\mathbf{I}_{\lambda/m}\left[\begin{array}{c}\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{2,1}\\\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{1,1}+\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{2,1}\\\hline\mathbf{A}_{1,2}^{1,1}\mathbf{B}_{1,2}^{2,1}\\\mathbf{A}_{1,2}^{1,1}\mathbf{B}_{1,2}^{1,1}+\mathbf{A}_{1,2}^{1,2}\mathbf{B}_{1,2}^{2,1}\\\hline *\\ *\\ *\end{array}\right],\tag{6.121}$$

where we have used $*$ to represent various combinations of interference symbols that can be found explicity by exapnding (6.117), whose exact forms are irrelevant. Note that the matrix $\hat{\mathbf{V}}'_{1,2,2,7}$ is a block diagonal matrix composed with two lower triangular toeplitz matrices and an identity matrix, thus is invertible, and the matrix $\hat{\mathbf{V}}_{1,2,2,7}\hat{\mathbf{V}}'_{1,2,2,7}\otimes\mathbf{I}_{\lambda/m}$ is then invertible from Lemma 6.1 and the fact that the Kronecker product of invertible matrices is invertible. Therefore, the user is able to recover desired products, i.e., $(\mathbf{A}_{1,1}^{1,1}\mathbf{B}_{1,1}^{1,1}+\mathbf{A}_{1,1}^{1,2}\mathbf{B}_{1,1}^{2,1})$ and $(\mathbf{A}_{1,2}^{1,1}\mathbf{B}_{1,2}^{1,1}+\mathbf{A}_{1,2}^{1,2}\mathbf{B}_{1,2}^{2,1})$, from the answers of any 7 servers by inverting the matrix. This completes the proof of recovery threshold $R=7$. Finally, it is straightforward to verify that the upload cost is $U_A=S/4=S/(K_cpm)$, $U_B=S/4=S/(K_cpn)$, and the download cost is $D=7/2$, which matches Theorem 6.2.

Arbitrary (ℓ,K_c,p,m,n) and $L=\ell K_c$

Define $R'=pmn$. Let $f_{1,1}$, $f_{1,2}$, $\cdots$, f_{ℓ,K_c}, α_1, α_2, $\cdots$, α_S be $(S+L)$ distinct elements from the field $\mathbb{F}$. For all $l\in[\ell],k\in[K_c]$, we define $c_{l,k,i},i\in\{0,1,\cdots,R'(K_c-1)\}$ to be the coefficients satisfying

$$\Psi_{l,k}(\alpha)=\prod_{k'\in[K_c]\setminus\{k\}}(\alpha+(f_{l,k'}-f_{l,k}))^{R'}=\sum_{i=0}^{R'(K_c-1)}c_{l,k,i}\alpha^i, \tag{6.122}$$

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 here by its roots. Now for all $l\in[\ell],s\in[S]$, let us define

$$\Delta_s^{,K_c}=\prod_{k\in[K_c]}(f_{l,k}-\alpha_s)^{R'}. \tag{6.123}$$

Let us also split the $L=\ell K_c$ instances of $\mathbf{A}$ and $\mathbf{B}$ matrices into ℓ groups, i.e.,

$$\mathbf{A}_{l,k}=\mathbf{A}_{K_c(l-1)+k}, \tag{6.124}$$

$$\mathbf{B}_{l,k}=\mathbf{B}_{K_c(l-1)+k} \tag{6.125}$$

for all $l\in[\ell],k\in[K_c]$. Further, for each matrix $\mathbf{A}_{l,k}$, we partition it into $m\times p$ blocks, denoted as $\mathbf{A}_{l,k}^{1,1},\mathbf{A}_{l,k}^{1,2},\cdots,\mathbf{A}_{l,k}^{m,p}$. Similarly, for each matrix $\mathbf{B}_{l,k}$, we partition it into $p\times n$ blocks, denoted as $\mathbf{B}_{l,k}^{1,1},\mathbf{B}_{l,k}^{1,2},\cdots,\mathbf{B}_{l,k}^{p,n}$. Now, for all $l\in[\ell],k\in[K_c]$, let us define

$$P_s^{l,k}=\sum_{m'\in[m]}\sum_{p'\in[p]}\mathbf{A}_{l,k}^{m',p'}(f_{l,k}-\alpha_s)^{p'-1+p(m'-1)}, \tag{6.126}$$

$$Q_s^{l,k} = \sum_{p'\in[p]}\sum_{n'\in[n]} \mathbf{B}_{l,k}^{p',n'}(f_{l,k}-\alpha_s)^{p-p'+pm(n'-1)}, \tag{6.127}$$

i.e., we apply EP codes for each $\mathbf{A}_{l,k}$ and $\mathbf{B}_{l,k}$. Note that the original EP codes can be regarded as polynomials of α_s, and here for each (l,k), we construct the EP codes as polynomials of $(f_{l,k}-\alpha_s)$. Now recall that by the construction of EP codes, the product $P_s^{l,k}Q_s^{l,k}$ can be written as weighted sums of the terms $1,(f_{l,k}-\alpha_s),\cdots,(f_{l,k}-\alpha_s)^{R'+p-2}$, i.e.,

$$P_s^{l,k}Q_s^{l,k} = \sum_{i=0}^{R'+p-2} \mathbf{C}_{l,k}^{(i+1)}(f_{l,k}-\alpha_s)^i, \tag{6.128}$$

where $\mathbf{C}_{l,k}^{(1)},\mathbf{C}_{l,k}^{(2)},\cdots,\mathbf{C}_{l,k}^{(R'+p-1)}$ are various linear combinations of products of blocks of $\mathbf{A}_{l,k}$ and blocks of $\mathbf{B}_{l,k}$. In particular, the desired product $\mathbf{A}_{l,k}\mathbf{B}_{l,k}$ can be obtained from $\mathbf{C}_{l,k}^{(1)},\cdots,\mathbf{C}_{l,k}^{(R')}$. Now we are ready to formally present the construction of generalized CSA codes. For all $s\in[S]$, let us construct shares of matrices $\mathbf{A}$ and $\mathbf{B}$ at the s^{th} server as follows.

$$\widetilde{A}^s = (\widetilde{A}_1^s,\widetilde{A}_2^s,\ldots,\widetilde{A}_\ell^s), \tag{6.129}$$

$$\widetilde{B}^s = (\widetilde{B}_1^s,\widetilde{B}_2^s,\ldots,\widetilde{B}_\ell^s), \tag{6.130}$$

where for $l\in[\ell]$, let us set

$$\widetilde{A}_l^s = \Delta_s^{l,K_c}\sum_{k\in[K_c]}\frac{1}{(f_{l,k}-\alpha_s)^{R'}}P_s^{l,k}, \tag{6.131}$$

$$\widetilde{B}_l^s = \sum_{k\in[K_c]}\frac{1}{(f_{l,k}-\alpha_s)^{R'}}Q_s^{l,k}. \tag{6.132}$$

The answer returned by the s^{th} server to the user is constructed as follows.

$$Y_s = \sum_{l\in[\ell]}\widetilde{A}_l^s\widetilde{B}_l^s. \tag{6.133}$$

Now let us prove that the generalized CSA codes are $R=pmn((\ell+1)K_c-1)+p-1$ recoverable. Let us rewrite Y_s as follows.

$$
\begin{aligned}
Y_s &= \widetilde{A}_1^s\widetilde{B}_1^s+\widetilde{A}_2^s\widetilde{B}_2^s+\cdots+\widetilde{A}_\ell^s\widetilde{B}_\ell^s && (6.134)\\
&= \sum_{l\in[\ell]}\Delta_s^{l,K_c}\left(\sum_{k\in[K_c]}\frac{1}{(f_{l,k}-\alpha_s)^{R'}}P_s^{l,k}\right)\left(\sum_{k\in[K_c]}\frac{1}{(f_{l,k}-\alpha_s)^{R'}}Q_s^{l,k}\right) && (6.135)\\
&= \sum_{l\in[\ell]}\sum_{k\in[K_c]}\frac{\prod_{k'\in[K_c]\setminus\{k\}}(f_{l,k'}-\alpha_s)^{R'}}{(f_{l,k}-\alpha_s)^{R'}}P_s^{l,k}Q_s^{l,k}\\
&\quad +\sum_{l\in[\ell]}\sum_{\substack{k,k'\in[K_c]\\ k\neq k'}}\left(\prod_{k''\in[K_c]\setminus\{k,k'\}}(f_{l,k''}-\alpha_s)^{R'}\right)P_s^{l,k}Q_s^{l,k'}. && (6.136)
\end{aligned}
$$

Note that in the last step, we split the summation into two parts depending on whether or not $k=k'$.

Let us consider the first term in (6.136). For each $l\in[\ell],k\in[K_c]$, we have

$$
\begin{aligned}
&\frac{\prod_{k'\in[K_c]\setminus\{k\}}(f_{l,k'}-\alpha_s)^{R'}}{(f_{l,k}-\alpha_s)^{R'}}P_s^{l,k}Q_s^{l,k} && (6.137)\\
=&\frac{\prod_{k'\in[K_c]\setminus\{k\}}\left((f_{l,k}-\alpha_s)+(f_{l,k'}-f_{l,k})\right)^{R'}}{(f_{l,k}-\alpha_s)^{R'}}P_s^{l,k}Q_s^{l,k} && (6.138)\\
=&\frac{\Psi_{l,k}(f_{l,k}-\alpha_s)}{(f_{l,k}-\alpha_s)^{R'}}P_s^{l,k}Q_s^{l,k} && (6.139)\\
=&\left(\frac{c_{l,k,0}}{(f_{l,k}-\alpha_s)^{R'}}+\frac{c_{l,k,1}}{(f_{l,k}-\alpha_s)^{R'-1}}+\cdots+\frac{c_{l,k,R'-1}}{f_{l,k}-\alpha_s}\right)P_s^{l,k}Q_s^{l,k}\\
&\quad+\left(\sum_{i=R'}^{R'(K_c-1)}c_{l,k,i}(f_{l,k}-\alpha_s)^{i-R'}\right)P_s^{l,k}Q_s^{l,k}, && (6.140)
\end{aligned}
$$

where in (6.139), we used the definition of $\Psi_{l,k}(\cdot)$, and in the next step, we rewrite the polynomial $\Psi_{l,k}(f_{l,k}-\alpha_s)$ in terms of its coefficients. Let us consider the first term in (6.140).

$$
\left(\frac{c_{l,k,0}}{(f_{l,k}-\alpha_s)^{R'}}+\frac{c_{l,k,1}}{(f_{l,k}-\alpha_s)^{R'-1}}+\cdots+\frac{c_{l,k,R'-1}}{f_{l,k}-\alpha_s}\right)P_s^{l,k}Q_s^{l,k} \qquad (6.141)
$$

$$=\left(\frac{c_{l,k,0}}{(f_{l,k}-\alpha_s)^{R'}}+\frac{c_{l,k,1}}{(f_{l,k}-\alpha_s)^{R'-1}}+\cdots+\frac{c_{l,k,R'-1}}{f_{l,k}-\alpha_s}\right)\sum_{i=0}^{R'+p-2}\mathbf{C}_{l,k}^{(i+1)}(f_{l,k}-\alpha_s)^i \tag{6.142}$$

$$\begin{aligned}=&\sum_{i=0}^{R'-1}\frac{\sum_{i'=0}^{i}c_{l,k,i-i'}\mathbf{C}_{l,k}^{(i'+1)}}{(f_{l,k}-\alpha_s)^{R'-i}}+\sum_{i=0}^{p-2}(f_{l,k}-\alpha_s)^i\left(\sum_{i'=i+1}^{R'+i}c_{l,k,R'-i'+i}\mathbf{C}_{l,k}^{(i'+1)}\right)\\ &+\sum_{i=p-1}^{R'+p-3}(f_{l,k}-\alpha_s)^i\left(\sum_{i'=i+1}^{R'+p-2}c_{l,k,R'-i'+i}\mathbf{C}_{l,k}^{(i'+1)}\right).\end{aligned} \tag{6.143}$$

We further note that when $K_c=1$, for all $i\neq 0, c_{l,k,i}=0$, thus the second term in (6.136), the second term in (6.140) and the third term in (6.143) equal zero. The second term in (6.143) can be expanded[13] into weighted sums of the terms $1,\alpha_s,\cdots,\alpha_s^{p-2}$. Since $K_c=1$, we can equivalently write these terms as $1,\alpha_s,\cdots,\alpha_s^{R'(K_c-1)+p-2}$. On the other hand, when $K_c>1$, the second term in (6.136), the second term in (6.140), the second and the third terms in (6.143) can also be expanded into weighted sums of the terms $1,\alpha_s,\cdots,\alpha_s^{R'(K_c-1)+p-2}$. Because $R'(K_c-1)+p-2=R-R'L-1$, in the matrix form, answers from any $R=pmn((\ell+1)K_c-1)+p-1$ servers, whose indices are denoted as $s_1,s_2,\cdots,s_R$, can be written as follows.

$$\begin{bmatrix}Y_{s_1}\\ Y_{s_2}\\ \vdots\\ Y_{s_R}\end{bmatrix}=\underbrace{\left[\begin{array}{ccc|c|ccc|ccc}\frac{1}{(f_{1,1}-\alpha_{s_1})^{R'}}&\cdots&\frac{1}{f_{1,1}-\alpha_{s_1}}&\cdots&\frac{1}{(f_{\ell,K_c}-\alpha_{s_1})^{R'}}&\cdots&\frac{1}{f_{\ell,K_c}-\alpha_{s_1}}&1&\cdots&\alpha_{s_1}^{R-R'L-1}\\ \frac{1}{(f_{1,1}-\alpha_{s_2})^{R'}}&\cdots&\frac{1}{f_{1,1}-\alpha_{s_2}}&\cdots&\frac{1}{(f_{\ell,K_c}-\alpha_{s_2})^{R'}}&\cdots&\frac{1}{f_{\ell,K_c}-\alpha_{s_2}}&1&\cdots&\alpha_{s_2}^{R-R'L-1}\\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\ \frac{1}{(f_{1,1}-\alpha_{s_R})^{R'}}&\cdots&\frac{1}{f_{1,1}-\alpha_{s_R}}&\cdots&\frac{1}{(f_{\ell,K_c}-\alpha_{s_R})^{R'}}&\cdots&\frac{1}{f_{\ell,K_c}-\alpha_{s_R}}&1&\cdots&\alpha_{s_R}^{R-R'L-1}\end{array}\right]}_{\hat{\mathbf{V}}_{\ell,K_c,R',R}}$$

[13] When $K_c=p=1$, the second term in (6.143) is zero, thus the Vandermonde terms do not appear. The matrix form representation now involves only confluent Cauchy matrices, i.e., confluent Cauchy-Vandermonde matrices without Vandermonde part.

$$\underbrace{\left[\begin{array}{c|c|c|c} \mathbf{T}(c_{1,1,0},\cdots,c_{1,1,R'-1}) & & & \\ \hline & \ddots & & \\ \hline & & \mathbf{T}(c_{\ell,K_c,0},\cdots,c_{\ell,K_c,R'-1}) & \\ \hline & & & \mathbf{I}_{R-R'L} \end{array}\right]}_{\hat{\mathbf{V}}'_{\ell,K_c,R',R}} \otimes \mathbf{I}_{\lambda/m} \left[\begin{array}{c} \mathbf{C}_{1,1}^{(1)} \\ \vdots \\ \mathbf{C}_{1,1}^{(R')} \\ \hline \vdots \\ \hline \mathbf{C}_{\ell,K_c}^{(1)} \\ \vdots \\ \mathbf{C}_{\ell,K_c}^{(R')} \\ \hline * \\ \vdots \\ * \end{array}\right], \tag{6.144}$$

We have used $*$ to represent various combinations of interference symbols that can be found explicitly by expanding (6.136), whose exact forms are irrelevant. Now since $f_{1,1}, f_{1,2}, \cdots, f_{\ell,K_c}$ are distinct, for all $l \in [\ell], k \in [K_c]$, we must have

$$c_{l,k,0} = \prod_{k' \in [K_c] \setminus \{k\}} (f_{l,k'} - f_{l,k})^{R'} \tag{6.145}$$

are non-zero. Hence, the lower triangular toeplitz matrices $\mathbf{T}(c_{1,1,0}, c_{1,1,1}, \cdots, c_{1,1,R'-1}), \cdots,$ $\mathbf{T}(c_{\ell,K_c,0}, c_{\ell,K_c,1}, \cdots, c_{\ell,K_c,R'-1})$ are non-singular, and the block diagonal matrix $\hat{\mathbf{V}}'_{\ell,K_c,R',R}$ is invertible. Guaranteed by Lemma 6.1 and the fact that the Kronecker product of non-singular matrices is non-singular, the matrix $(\hat{\mathbf{V}}_{\ell,K_c,R',R} \hat{\mathbf{V}}'_{\ell,K_c,R',R}) \otimes \mathbf{I}_{\lambda/m}$ is invertible. Therefore, the user is able to recover $(\mathbf{C}_{l,k}^{(i)})_{l \in [\ell], k \in [K_c], i \in [R']}$ by inverting the matrix. And the desired products $(\mathbf{A}_l \mathbf{B}_l)_{l \in [L]}$ are recoverable from $(\mathbf{C}_{l,k}^{(i)})_{l \in [\ell], k \in [K_c], i \in [R']}$, guaranteed by the construction of Entangled Polynomial codes. This completes the proof of recovery threshold $R = pmn((\ell+1)K_c - 1) + p - 1$. It is also easy to see that the upload cost $U_A = S/(K_c pm)$ and $U_B = S/(K_c pn)$. Note that we are able to recover Lmn desired symbols from R downloaded answers, so the download cost is $D=\frac{R}{Lmn}=\frac{pmn((\ell+1)K_c-1)+p-1}{mn\ell K_c}$. Thus the desired costs are achievable. Note that the encoding procedure can be considered as products of Confluent Cauchy matrices by vectors. By fast algorithms [83], 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. Now let us consider the decoding complexity. Note that the decoding procedure involves matrix-vector multiplications of inverse of Toeplitz matrix and inverse of confluent Cauchy-Vandermonde matrix. From the inverse formula of confluent Cauchy-Vandermonde matrix presented in [132], the matrix-vector multiplication of the inverse of confluent Cauchy-Vandermonde matrix $\hat{\mathbf{V}}_{\ell,K_c,R',R}$ can be decomposed into a series of structured matrix-vector multiplications including confluent Cauchy matrix, transpose of Vandermonde matrix, Hankel matrix and Toeplitz matrix. By fast algorithms [83, 41], the complexity of decoding is at most $\widetilde{\mathcal{O}}(\lambda\mu p\log^2 R)$. With straightforward matrix multiplication algorithms, the server computation complexity is $\mathcal{C}_s=(\lambda\kappa\mu)/(K_c pmn)$. This completes the proof of Theorem 6.2.

6.6 N-CSA Codes for N-linear Coded Distributed Batch Computation (N-CDBC)

6.6.1 N-CSA Codes: Main Result

In this section, let us generalize CSA codes for N-CDBC. The generalization, called N-CSA codes, is presented in the following theorem.

Theorem 6.3. *For N-CDBC over a field $\mathbb{F}$ with S servers, and positive integers ℓ,K_c such that $L=\ell K_c\leq|\mathbb{F}|-S$, the N-CSA codes introduced in this section achieve*

$$\textit{Recovery Threshold:}\qquad R=K_c(N+\ell-1)-N+1, \tag{6.146}$$

$$\textit{Upload Cost for } \widetilde{X^{(n)}}^{[S]}, n\in[N]\textit{:}\qquad U_{X^{(n)}}=\frac{S}{K_c}, \tag{6.147}$$

$$\textit{Download Cost:} \qquad D = \frac{K_c(N+\ell-1)-N+1}{\ell K_c}, \tag{6.148}$$

$$\textit{Server Computation Complexity:} \qquad \mathcal{C}_s = \mathcal{O}(\omega/K_c), \tag{6.149}$$

$$\textit{Encoding complexity for } \widetilde{X^{(n)}}^{[S]}\textit{, } n\in[N]\textit{:} \qquad \mathcal{C}_{eX^{(n)}} = \widetilde{\mathcal{O}}\left(\frac{\dim(V_n)S\log^2 S}{K_c}\right), \tag{6.150}$$

$$\textit{Decoding complexity:} \qquad \mathcal{C}_d = \widetilde{\mathcal{O}}\left(\frac{\ell+N-1}{\ell}\dim(W)R\log^2 R\right), \tag{6.151}$$

where ω is the number of arithmetic operations required to compute the N-linear map $\Omega(\cdot)$.

6.6.2 Proof of Theorem 6.3

Now let us present the construction of N-CSA codes for N-CDBC. Let $f_{1,1}, f_{1,2}, \cdots, f_{\ell,K_c}$, $\alpha_1, \alpha_2, \cdots, \alpha_S$ represent $(S+L)$ distinct elements from $\mathbb{F}$. For all $l\in[\ell], s\in[S]$, let us define

$$\Delta_s^{l,K_c} = \prod_{k\in[K_c]}(f_{l,k}-\alpha_s). \tag{6.152}$$

For all $n\in[N], l\in[\ell], k\in[K_c]$, we define

$$x_{l,k}^{(n)} = x_{K_c(l-1)+k}^{(n)}. \tag{6.153}$$

For all $s\in[S], n\in[N]$, we construct $\widetilde{X^{(n)}}^s$ as follows.

$$\widetilde{X^{(n)}}^s = (\widetilde{X^{(n)}}_1^s, \widetilde{X^{(n)}}_2^s, \cdots, \widetilde{X^{(n)}}_\ell^s), \tag{6.154}$$

where for $l\in[\ell]$, let us set

$$\widetilde{X^{(n)}}_l^s = \Delta_s^{l,K_c}\sum_{k\in[K_c]}\frac{1}{f_{l,k}-\alpha_s}x_{l,k}^{(n)}. \tag{6.155}$$

The answer returned by the s^{th} server is constructed as follows.

$$Y_s = \sum_{l\in[\ell]} \frac{1}{\Delta_s^{,K_c}} \Omega(\widetilde{X^{(1)}}^{s}_{l}, \widetilde{X^{(2)}}^{s}_{l}, \cdots, \widetilde{X^{(N)}}^{s}_{l}). \tag{6.156}$$

To prove that the code is R-recoverable, let us rewrite Y_s as follows.

$$\begin{aligned}
Y_s &= \sum_{l\in[\ell]} \frac{1}{\Delta_s^{,K_c}} \Omega(\widetilde{X^{(1)}}^{s}_{l}, \widetilde{X^{(2)}}^{s}_{l}, \cdots, \widetilde{X^{(N)}}^{s}_{l}) && (6.157)\\
&= \sum_{l\in[\ell]} \frac{1}{\Delta_s^{,K_c}} \Omega\left(\Delta_s^{,K_c} \sum_{k\in[K_c]} \frac{1}{f_{l,k} - \alpha_s} x_{l,k}^{(1)}, \cdots, \Delta_s^{,K_c} \sum_{k\in[K_c]} \frac{1}{f_{l,k} - \alpha_s} x_{l,k}^{(N)} \right) && (6.158)\\
&= \sum_{l\in[\ell]} (\Delta_s^{,K_c})^{N-1} \left(\sum_{k_1\in[K_c]} \frac{1}{f_{l,k_1} - \alpha_s} \cdots \sum_{k_N\in[K_c]} \frac{1}{f_{l,k_N} - \alpha_s} \left(\Omega(x_{l,k_1}^{(1)}, \cdots, x_{l,k_N}^{(N)}) \right) \right) && (6.159)\\
&= \sum_{l\in[\ell]} \sum_{k\in[K_c]} \frac{\prod_{k'\in[K_c]\backslash\{k\}} (f_{l,k'} - \alpha_s)^{N-1}}{(f_{l,k} - \alpha_s)} \Omega(x_{l,k}^{(1)}, \cdots, x_{l,k}^{(N)})\\
&\quad + \sum_{l\in[\ell]} \sum_{\substack{k_1,\cdots,k_N\in[K_c],\\ \neg(k_1=\cdots=k_N)}} \left(\frac{(\Delta_s^{,K_c})^{N-1}}{(f_{l,k_1} - \alpha_s)\cdots(f_{l,k_N} - \alpha_s)} \Omega(x_{l,k_1}^{(1)}, \cdots, x_{l,k_N}^{(N)}) \right), && (6.160)
\end{aligned}$$

where in (6.160), we split the summation depending on whether or not $k_1 = k_2 = \cdots = k_N$. Following the same argument presented in Section 6.4.3, by performing long division of polynomials for the first term in (6.160), and noting that the second term in (6.160) can be expanded to weighted sums of the terms $1, \alpha_s, \alpha_s^2, \cdots, \alpha_s^{K_c(N-1)-N}$, the presented code is

$(R = K_c(N+\ell-1)-N+1)$-recoverable as long as the following matrix is non-singular.

$$\underbrace{\begin{bmatrix} \frac{1}{f_{1,1}-\alpha_{s_1}} & \frac{1}{f_{1,2}-\alpha_{s_1}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_1}} & 1 & \alpha_{s_1} & \cdots & \alpha_{s_1}^{R-L-1} \\ \frac{1}{f_{1,1}-\alpha_{s_2}} & \frac{1}{f_{1,2}-\alpha_{s_2}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_2}} & 1 & \alpha_{s_2} & \cdots & \alpha_{s_2}^{R-L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_{1,1}-\alpha_{s_R}} & \frac{1}{f_{1,2}-\alpha_{s_R}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_R}} & 1 & \alpha_{s_R} & \cdots & \alpha_{s_R}^{R-L-1} \end{bmatrix}}_{\mathbf{V}_{\ell,K_c,R}} \underbrace{\begin{bmatrix} c_{1,1} & & & & & & \\ & c_{1,2} & & & & & \\ & & \ddots & & & & \\ & & & c_{\ell,K_c} & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}}_{\mathbf{V}'_{\ell,K_c,R}}, \qquad (6.161)$$

where for all $l \in [\ell], k \in [K_c]$, $c_{l,k} = \prod_{k' \in [K_c] \setminus \{k\}} (f_{l,k'} - f_{l,k})^{N-1}$. The indices of any R responsive servers are denoted as $s_1, s_2, \cdots, s_R$. Since $f_{1,1}, f_{1,2}, \cdots, f_{l,k}$ are distinct elements from $\mathbb{F}$, $(c_{l,k})_{l\in[\ell],k\in[K_c]}$ are non-zero, and $R-L-1 = K_c(N-1)-N$, the matrix $\mathbf{V}_{\ell,K_c,R}\mathbf{V}'_{\ell,K_c,R}$ is invertible guaranteed by Lemma 2.5. This completes the proof of recovery threshold. The upload cost for $\widetilde{X^{(n)}}^{[S]}, n \in [N]$ is readily verified to be S/K_c, and the download cost is $D = R/L = \frac{K_c(N+\ell-1)-N+1}{\ell K_c}$. By fast algorithms discussed in Section 6.4.3, we can achieve the encoding/decoding complexity as presented in Theorem 6.3. The computational complexity at each server is $\mathcal{O}(\ell\omega/L) = \mathcal{O}(\omega/K_c)$, where ω is the number of arithmetic operations required to compute $\Omega(\cdot)$. This completes the proof of Theorem 6.3.

Remark 6.1. *Let us regard a multivariate polynomial of total degree N as a linear combination of various restricted evaluations of N-linear maps.*[14] *Note that the construction for $\widetilde{X^{(n)}}^{s}$ is symmetric across all $n \in [N]$. N-CSA codes can also be transformed to evaluate a multivariate polynomial at L points as follows. For each server $s \in [S]$, the answer is computed for each N-linear map according to N-CSA codes, and each server returns the user*

[14] Note that this does not alter the computation complexity of the multivariate polynomial. For any monomial of the polynomial of degree $M < N$, when it is viewed (homogenized) as a restricted N-linear map, it is viewed as a product of M variables, along with $(N-M)$ *ones*. The latter is constant, which requires no extra computation.

with the linear combination of the answers. It is easy to see that the user is able to recover the evaluation of the multivariate polynomial of total degree N at the given L points from the answers of any $R=K_c(N+\ell-1)-N+1$ servers. The LCC codes in [138], which achieve the recovery threshold $R=K_cN-N+1$, are a special case of this construction, where $\ell=1$.

Remark 6.2. *The systematic construction presented in Section 6.4.4 can be also applied directly to N-CSA codes for N-CDBC, i.e., for all $s\in[L]$, uncoded variables $(x_s^{(n)})_{n\in[N]}$ are uploaded to the s^{th} server, and coded shares are uploaded to the remaining $S-L$ servers, according to the same coding scheme. Similarly, the recovery threshold is not affected by the systematic construction.*

Remark 6.3. *The Lagrange codes presented in [138] for N-CDBC and can be considered as a special case of N-CSA codes obtained by setting the parameter $\ell=1$. Note that the download cost can be written as $D=1+\left(\frac{N-1}{\ell}\right)\left(\frac{K_c-1}{K_c}\right)$. The parameter ℓ plays an important role in improving the download cost, which may be of interest when N is large and the down-link is costly. For example, let us assume that R/S is held constant, then the order of the download cost achieved is $\mathcal{O}(1+(N-1)/\ell)$ and the order of the upload cost for $\widetilde{X^{(n)}}^{[S]}, n\in[N]$ achieved is $\mathcal{O}(\ell+(N-1))$, which offers flexible trade-off between the upload cost and download cost.*

6.7 Discussion

The main contribution of this chapter is a class of codes, based on the idea of Cross Subspace Alignment (CSA) that originated in private information retrieval (PIR) literature. These codes are shown to unify, generalize and improve upon existing algorithms for coded distributed batch matrix multiplication, N-linear batch computation, and multivariate batch polynomial evaluation, such as Polynomial, MatDot and PolyDot codes, Generalized PolyDot and Entangled Polynomial (EP) codes, and Lagrange Coded Computing (LCC). CSA codes for coded distributed batch matrix multiplication, which include LCC codes as a special case, improve significantly upon state of art matrix-partitioning approaches (EP codes) in terms of communication cost, and upon LCC codes in download-constrained settings. Generalized CSA (GCSA) codes bridge the extremes of matrix partitioning based approaches (EP codes) and batch processing approaches (CSA codes, LCC codes), and allow a tradeoff between server computation complexity, which is improved by emphasizing the matrix partitioning aspect, and communication costs, which are improved by emphasizing the batch processing aspect. N-CSA codes for N-linear batch computations and multivariate polynomial evaluations similarly generalize LCC codes, offering advantages especially in download constrained settings. As a final observation, note that LCC codes in [138] also allow settings with X-secure data and B-byzantine servers. Given that cross-subspace alignment schemes originated in PIR with X-security constraints in Chapter 2 and have also been applied to B-byzantine settings in Chapter 4, extensions of CSA codes, GCSA codes and N-CSA codes to X-secure and B-byzantine settings are relatively straightforward, as shown in Appendix D.1.

Chapter 7

X-Secure T-Private Federated Submodel Learning with Elastic Dropout Resilience

Motivated by recent interest in federated submodel learning, this chapter explores the fundamental problem of privately reading from and writing to a database comprised of K files (submodels) that are stored across N distributed servers according to an X-secure threshold secret sharing scheme. One after another, various users wish to retrieve their desired file, locally process the information and then update the file in the distributed database while keeping the identity of their desired file private from any set of up to T colluding servers. The availability of servers changes over time, so elastic dropout resilience is required. The main contribution of this chapter is an adaptive scheme, called ACSA-RW, that takes advantage of all currently available servers to reduce its communication costs, fully updates the database after each write operation even though the database is only partially accessible due to server dropouts, and ensures a memoryless operation of the network in the sense that the storage structure is preserved and future users may remain oblivious of the past history of server dropouts. The ACSA-RW construction builds upon cross-subspace alignment (CSA) codes that were originally introduced for X-secure T-private information retrieval and have been shown to be natural solutions for secure distributed matrix multiplication problems. ACSA-RW achieves the desired private read and write functionality with elastic dropout resilience, matches the best results for private-read from PIR literature, improves significantly upon available baselines for private-write, reveals a striking symmetry between upload and download costs, and exploits redundant storage dimensions to accommodate arbitrary read and write dropout servers up to certain threshold values. It also answers in the affirmative an open question by Kairouz et al. by exploiting synergistic gains from the joint design of private read and write operations.

7.1 Introduction

The rise of machine learning is marked by fundamental tradeoffs between competing concerns. Central to this chapter are 1) the need for abundant training data, 2) the need for privacy, and 3) the need for low communication cost. Federated learning [78, 80, 73, 131] is a distributed machine learning paradigm that addresses the first two concerns by allowing distributed users/clients (e.g., mobile phones) to collaboratively train a shared model that is stored in a cluster of databases/servers (cloud) while keeping their training data private. The users retrieve the current model, train the model locally with their own training data, and then aggregate the modifications as focused updates. Thus, federated learning allows utilization of abundant training data while preserving its privacy. However, this incurs higher communication cost as each update involves communication of all model parameters.

Communicating the full model may be unnecessary for large scale machine learning tasks where each user's local data is primarily relevant to a small part of the overall model. Federated *Submodel* Learning (FSL) [82] builds on this observation by partitioning the model into multiple submodels and allowing users to selectively train and update the submodels that are most relevant to their local view. This is the case, for example, in the binary relevance method for multi-label classification[90, 76], which *independently* trains a series of binary classifiers (viewed as submodels), one for each label. Given a sample to be predicted, the compound model predicts all labels for which the respective classifiers yield a positive result.

What makes federated submodel learning challenging is the privacy constraint. The identity of the submodel that is being retrieved and updated by a user must remain private. Prior works [82, 79, 1, 34, 5] that assume centralized storage of all submodels are generally able to provide relatively weaker privacy guarantees such as plausible deniability through differential privacy mechanisms that perturb the data and rely on secure inter-user peer-to-peer communication for secure aggregation. On the other hand, it is noted recently by Kim and Lee in [64] that if the servers that store the submodels are *distributed*, then stronger information theoretic guarantees such[1] as "perfect privacy" may be attainable, without the need for user-to-user communication. Indeed, in this chapter we focus on this setting of distributed servers and perfect privacy. The challenge of federated submodel learning in this setting centers around three key questions.

- **Q1 Private Read:** How can a user efficiently retrieve the desired submodel from the distributed servers without revealing which submodel is being retrieved?
- **Q2 Private Write:** How can a user efficiently update the desired submodel to the distributed servers without revealing which submodel is being updated?
- **Q3 Synergy of Private Read-Write:** Are there synergistic gains in the *joint* design of retrieval and update operations, and if so, then how to exploit these synergies?

The significance of these fundamental questions goes well beyond federated submodel learning. As recognized by [82] the private read question (Q1) by itself is equivalent to the problem

[1] By *perfect* privacy we mean that absolutely no information is leaked about the identity of a user's desired submodel to any set of colluding servers up to a target threshold.

of Private Information Retrieval (PIR) [23, 24], which has recently been studied extensively from an information theoretic perspective [102, 106, 103, 120, 117, 12, 109, 119, 111, 116, 36, 105, 54, 143, 130, 57, 7, 128, 4, 127, 112, 71, 126, 22, 104, 134, 9, 97, 123, 115, 107, 81, 21, 53, 75]. Much less is known about Q2 and Q3, i.e., the fundamental limits of private-write, and joint read-write solutions from the information theoretic perspective. Notably, Q3 has also been highlighted previously as an open problem by Kairouz et al in [80].

The problem of privately reading and writing data from a distributed memory falls under the larger umbrella of Distributed Oblivious RAM (DORAM)[74] primitives in theoretical computer science and cryptography. With a few limited exceptions (e.g., a specialized 4-server construction in [66] that allows information theoretic privacy), prior studies of DORAM generally take a cryptographic perspective, e.g., privacy is guaranteed subject to computational hardness assumptions, and the number of memory blocks is assumed to be much larger than the size of each block. In contrast, the focus of this chapter is on Q2 and Q3 under the stronger notion of information theoretic privacy. Furthermore, because our motivation comes from federated submodel learning, the size of a submodel is assumed to be significantly larger than the number of submodels (see motivating examples in [82] and Section 7.3.1). Indeed, this is a pervasive assumption in the growing body of literature on information theoretic PIR[102, 106, 103, 120, 117, 12, 109, 119, 111, 116, 36, 105, 54, 143, 130, 57, 7, 128, 4, 127, 112, 71, 126, 22, 104, 134, 9, 97, 123, 115, 107, 81, 21, 53, 75]. In a broad sense, our problem formulation in this chapter is motivated by applications of (information theoretic) PIR that also require private writes. There is no shortage of such applications, e.g., a distributed database of medical records that not only allows a physician to privately download the desired record (private read) but also to update the record with new information (private write), or a banking service that would similarly allow private reads and writes of financial records from authorized entities. Essentially, while FSL serves as our nominal application of interest based on prior works that motivated this effort, our problem formulation is broad enough to capture various distributed file systems that enable the users to read and write files without revealing the identity of the target file. The files are viewed as submodels, and the assumption that the size of the file is significantly larger than the number of the files captures the nature of file systems that are most relevant to this chapter.

Overview: We consider the federated submodel learning setting where the global model is partitioned into K submodels, and stored among N distributed servers according to an X-secure threshold secret sharing scheme, i.e., any set of up to X colluding servers can learn nothing about the stored models, while the full model can be recovered from the data stored by any $X+K_c$ servers. One at a time, users update the submodel most relevant to their local training data. The updates must be T-private, i.e., any set of up to T colluding servers must not learn anything about which submodel is being updated. The contents of the updates must be X_Δ-secure, i.e., any set of up to X_Δ colluding servers must learn nothing about the contents of the submodel updates. The size of a submodel is significantly larger than the number of submodels, which is significantly larger than 1, i.e., $L \gg K \gg 1$ where L is the size of a submodel and K is the number of submodels. Due to uncertainties of the servers' I/O states, link states, etc., an important concern in distributed systems is to allow resilience against servers that may temporarily drop out [59, 68, 114, 19, 98]. To this end, we assume that at each time $t, t \in \mathbb{N}$, a subset of servers may be unavailable. These unavailable servers are referred to as read-dropout servers or write-dropout servers depending on whether the user intends to perform the private read or the private write operation. Since the set of dropout servers changes over time, and is assumed to be known to the user, the private read and write schemes must *adapt* to the set of currently available servers. Note that this is different from the problem of stragglers in massive distributed computing applications where the set of responsive servers is not known in advance, because servers may become unavailable *during* the lengthy time interval required for their local computations. Since our focus is not on massive computing applications, the server side processing needed for private read and write is not as time-consuming. So the availabilities, which are determined in advance by the user before initiating the read or write operation, e.g., by pinging the servers, are not expected to change during the read or write operation. We do allow the server availabilities to change between the read and write operations due to the delay introduced by the intermediate processing that is needed at the user to generate his updated submodel. A somewhat surprising aspect of private write with unavailable servers is that even though the data at the unavailable servers cannot be updated, the collective storage at all servers (including the unavailable ones) must represent the updated models. The redundancy in coded storage and the X-security constraints which require that the stored information at any X servers is independent of the data, are essential in this regard.

Since the private-read problem (Q1) is essentially a form of PIR, our starting point is the X-secure T-private information retrieval scheme (XSTPIR) of Chapter 2. In particular, we build on the idea of cross-subspace alignment (CSA), and introduce a new private read-write scheme, called Adaptive CSA-RW (ACSA-RW) as an answer to Q1 and Q2. To our knowledge ACSA-RW is the first *communication-efficient* private federated submodel learning scheme that achieves information-theoretically *perfect* privacy. ACSA-RW also answers Q3 in the affirmative as it exploits query structure from the private-read operation to reduce the communication cost for the private-write operation. The evidence of synergistic gain in ACSA-RW from a joint design of submodel retrieval and submodel aggregation addresses the corresponding open problem highlighted in Section 4.4.4 of [80]. The observation that the ACSA-RW scheme takes advantage of storage redundancy for private read and private write is indicative of fundamental tradeoffs between download cost, upload cost, data security level, and storage redundancy for security and recoverability. In particular, the storage redundancy for X-security is exploited by private write, while the storage redundancy for robust recoverability is used for private read (see Theorem 7.1 and Section 7.3.1 for details). It is also remarkable that the ACSA-RW scheme requires absolutely no user-user communication, even though the server states change over time and the read-write operations are adaptive. In other words, a user is not required to be aware of the history of previous updates and the previous availability states of the servers (see Section 7.3.1 for details). The download cost and the upload cost achieved by each user is an increasing function of the number of unavailable servers at the time. When more servers are available, the download cost and the upload costs are reduced, which provides elastic dropout resilience (see Theorem 7.1). To this end, the ACSA-RW scheme uses adaptive MDS-coded answer strings. This idea originates from CSA code constructions for the problem of coded distributed batch computation in Chapter 6 (also see [55]). In terms of comparisons against available baselines, we note (see Section 7.3.1) that ACSA-RW improves significantly in both the communication efficiency and the level of privacy compared to [64]. In fact, ACSA-RW achieves asymptotically optimal download cost when $X \geq X_\Delta + T$, and is order-wise optimal in terms of the upload cost. Compared with the 4 server construction of information theoretic DORAM in [66], (where $X=1, T=1, X_\Delta=0, N=4$) ACSA-RW has better communication efficiency (the assumption of $L \gg K$ is important in this regard). For example, as the ratio L/K approaches infinity, ACSA-RW achieves total communication cost (i.e., the summation of the download cost and the upload cost, normalized by the submodel size) of 6, versus the communication cost of 8 achieved by the construction in [66].

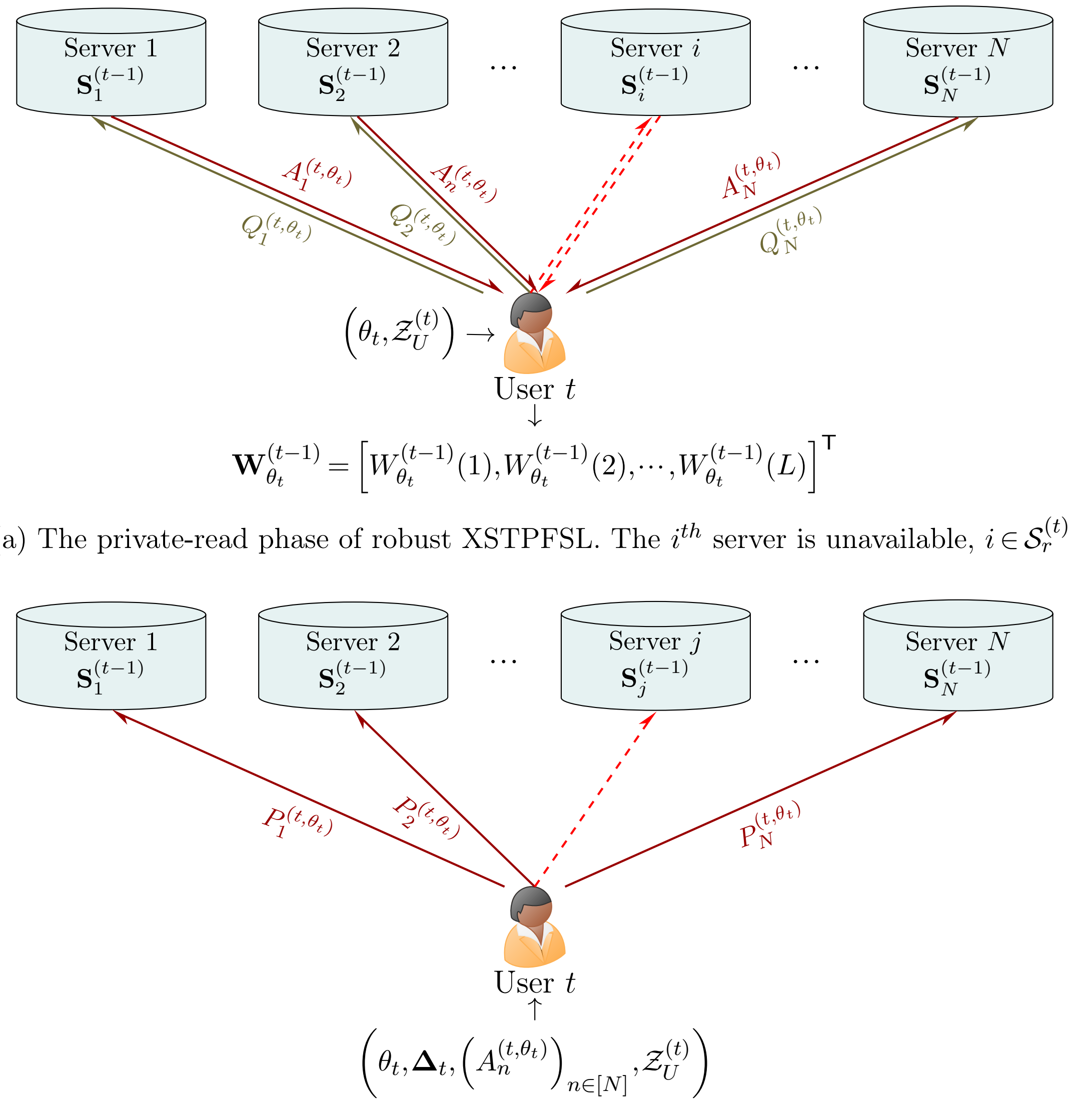
7.2 Problem Statement: Robust XSTPFSL

Consider K initial submodels $\left(\mathbf{W}_1^{(0)}, \mathbf{W}_2^{(0)}, \cdots, \mathbf{W}_K^{(0)}\right)$, each of which consists of L uniformly i.i.d.[2] random symbols from a finite field[3] $\mathbb{F}_q$. In particular, we have

$$H\left(\mathbf{W}_1^{(0)}, \mathbf{W}_2^{(0)}, \cdots, \mathbf{W}_K^{(0)}\right) = KL, \tag{7.1}$$

[2] Note that the proposed ACSA-RW scheme does not require the assumption of uniformly i.i.d. model data to be correct, secure and private. The assumption is mainly used for converse arguments and communication cost metrics. However, we note that the uniformly i.i.d. model data assumption is in fact *not* very strong because submodel learning is performed locally, and it may be possible to achieve (nearly) uniformly i.i.d. model data by exploiting entropy encoding.

[3] Note that the size of the finite field required is not too large. By our ACSA-RW scheme (see Theorem 7.1), it is sufficient to choose a finite field with $q \geq 2N$. Besides, finite field symbols are used as a representation of the submodels. Since the submodels are trained by the user *locally*, without loss of generality we can assume finite field representations.


Server 1 $\mathbf{S}_1^{(t-1)}$
Server 2 $\mathbf{S}_2^{(t-1)}$
$\cdots$
Server i $\mathbf{S}_i^{(t-1)}$
$\cdots$
Server N $\mathbf{S}_N^{(t-1)}$
$A_1^{(t,\theta_t)}$
$A_n^{(t,\theta_t)}$
$Q_1^{(t,\theta_t)}$
$Q_2^{(t,\theta_t)}$
$A_N^{(t,\theta_t)}$
$Q_N^{(t,\theta_t)}$
$\left(\theta_t, \mathcal{Z}_U^{(t)}\right) \rightarrow$
User t
$\downarrow$
$\mathbf{W}_{\theta_t}^{(t-1)} = \left[W_{\theta_t}^{(t-1)}(1), W_{\theta_t}^{(t-1)}(2), \cdots, W_{\theta_t}^{(t-1)}(L)\right]^{\mathsf{T}}$
Server 1 $\mathbf{S}_1^{(t-1)}$
Server 2 $\mathbf{S}_2^{(t-1)}$
$\cdots$
Server j $\mathbf{S}_j^{(t-1)}$
$\cdots$
Server N $\mathbf{S}_N^{(t-1)}$
$P_1^{(t,\theta_t)}$
$P_2^{(t,\theta_t)}$
$P_N^{(t,\theta_t)}$
User t
$\uparrow$
$\left(\theta_t, \boldsymbol{\Delta}_t, \left(A_n^{(t,\theta_t)}\right)_{n\in[N]}, \mathcal{Z}_U^{(t)}\right)$

(a) The private-read phase of robust XSTPFSL. The i^{th} server is unavailable, $i \in \mathcal{S}_r^{(t)}$.

(b) The private-write phase of robust XSTPFSL. The j^{th} server is unavailable, $j \in \mathcal{S}_w^{(t)}$. Note that Server i and Server j may be different servers.

Figure 7.1: The two phases of robust X-Secure T-Private Federated Submodel Learning (XSTPFSL) with arbitrary realizations of unavailable servers.

in q-ary units. Time slots are associated with users and their corresponding submodel updates, i.e., at time slot $t, t \in \mathbb{N}$, User t wishes to perform the t^{th} submodel update. At time $t, t \in \mathbb{Z}^*$, the K submodels are denoted as $\left(\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)}, \cdots, \mathbf{W}_K^{(t)}\right)$. The submodels are represented as vectors, i.e., for all $t \in \mathbb{Z}^*$, $k \in [K]$,

$$\mathbf{W}_k^{(t)} = \left[W_k^{(t)}(1), W_k^{(t)}(2), \cdots, W_k^{(t)}(L)\right]^{\mathsf{T}}. \tag{7.2}$$

The K submodels are distributively stored among the N servers. The storage at server $n, n \in [N]$ at time $t, t \in \mathbb{Z}^*$ is denoted as $\mathbf{S}_n^{(t)}$. Note that $\mathbf{S}_n^{(0)}$ represents the initial storage.

A full cycle of robust XSTPFSL is comprised of two phases — the read phase and the write phase. At the beginning of the cycle, User t privately generates the desired index θ_t, uniformly from $[K]$, and the user-side randomness $\mathcal{Z}_U^{(t)}$, which is intended to protect the user's privacy and security. In the read phase, User t wishes to retrieve the submodel $\mathbf{W}_{\theta_t}^{(t-1)}$. At all times, nothing must be revealed about any (current or past) desired indices $(\theta_1, \theta_2, \cdots, \theta_t)$ to any set of up to T colluding servers. To this end, User t generates the read-queries $\left(Q_1^{(t,\theta_t)}, Q_2^{(t,\theta_t)}, \cdots, Q_N^{(t,\theta_t)}\right)$, where $Q_n^{(t,\theta_t)}$ is intended for the n^{th} server, such that,

$$H\left(Q_n^{(t,\theta_t)} \;\middle|\; \theta_t, \mathcal{Z}_U^{(t)}\right) = 0, \qquad \forall n \in [N]. \tag{7.3}$$

Each of the currently available servers $n, n \in [N] \setminus \mathcal{S}_r^{(t)}$ is sent the query $Q_n^{(t,\theta_t)}$ by the user, and responds to the user with an answer $A_n^{(t,\theta_t)}$, such that,

$$H\left(A_n^{(t,\theta_t)} \;\middle|\; \mathbf{S}_n^{(t-1)}, Q_n^{(t,\theta_t)}\right) = 0, \qquad \forall n \in [N] \setminus \mathcal{S}_r^{(t)}. \tag{7.4}$$

From the answers returned by the servers $n, n \in [N] \setminus \mathcal{S}_r^{(t)}$, the user must be able to reconstruct the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$.

$$\text{[Correctness]}\quad H\left(\mathbf{W}_{\theta_t}^{(t-1)}\;\middle|\; \big(A_n^{(t,\theta_t)}\big)_{n\in[N]\setminus\mathcal{S}_r^{(t)}}, \big(Q_n^{(t,\theta_t)}\big)_{n\in[N]}, \theta_t\right)=0.$$

This is the end of the read phase.

Upon finishing the local submodel training, User t privately generates an increment[4] for the θ_t^{th} submodel. The increment is represented as a vector, $\mathbf{\Delta}_t=[\Delta_1^{(t)},\Delta_2^{(t)},\cdots,\Delta_L^{(t)}]^{\mathsf{T}}$, which consists of L i.i.d. uniformly distributed symbols from the finite field $\mathbb{F}_q$, i.e., $H(\mathbf{\Delta}_t)=L$ in q-ary units. The increment $\mathbf{\Delta}_t$ is intended to update the θ_t^{th} submodel $\mathbf{W}_{\theta_t}^{(t-1)}$, such that the next user who wishes to make an update, User $t+1$, is able to retrieve the submodel $\mathbf{W}_{\theta_t}^{(t)}=\mathbf{W}_{\theta_t}^{(t-1)}+\mathbf{\Delta}_t$ if $\theta_{t+1}=\theta_t$, and the submodel $\mathbf{W}_{\theta_t'}^{(t)}=\mathbf{W}_{\theta_t'}^{(t-1)}$ if $\theta_{t+1}=\theta_t'\neq\theta_t$. In other words, for all $t\in\mathbb{N}$, $k\in[K]$, the submodel $\mathbf{W}_k^{(t)}$ is defined recursively as follows.

$$\mathbf{W}_k^{(t)}=\begin{cases}\mathbf{W}_k^{(t-1)}+\mathbf{\Delta}_t & k=\theta_t,\\ \mathbf{W}_k^{(t-1)} & k\neq\theta_t.\end{cases}\tag{7.5}$$

User t initializes the write phase by generating the write-queries $\left(P_1^{(t,\theta_t)},P_2^{(t,\theta_t)},\cdots,P_N^{(t,\theta_t)}\right)$. For ease of notation, let the write-queries be nulls for the write-dropout servers, i.e., $P_n^{(t,\theta_t)}=$ for $n\in\mathcal{S}_w^{(t)}$. For all $n\in[N]$,

$$H\left(P_n^{(t,\theta_t)}\;\middle|\;\theta_t,\mathcal{Z}_U^{(t)},\big(A_n^{(t,\theta_t)}\big)_{n\in[N]},\mathbf{\Delta}_t\right)=0.\tag{7.6}$$

The user sends the write-query $P_n^{(t,\theta_t)}$ to the n^{th} server, $n\in[N]\setminus\mathcal{S}_w^{(t)}$, if the server was available in the read-phase and therefore already received the read-query. Otherwise, if the

[4] In general, the increment is the difference between the new submodel and the old submodel. We note that no generality is lost by assuming additive increments because the submodel training is performed locally.

server was not available during the read phase, then the user sends[5] both read and write queries $(Q_n^{(t,\theta_t)},P_n^{(t,\theta_t)})$. Still, any set of up to T colluding servers must learn nothing about the desired indices $(\theta_1,\theta_2,\cdots,\theta_t)$.

Upon receiving the write-queries, each of the servers $n,n\in[N]\setminus\mathcal{S}_w^{(t)}$ updates its storage based on the existing storage $\mathbf{S}_n^{(t-1)}$ and the queries for the two phases $\left(P_n^{(t,\theta_t)},Q_n^{(t,\theta_t)}\right)$, i.e.,

$$H\left(\mathbf{S}_n^{(t)}\;\middle|\;\mathbf{S}_n^{(t-1)},P_n^{(t,\theta_t)},Q_n^{(t,\theta_t)}\right)=0. \tag{7.7}$$

On the other hand, the write-dropout servers are unable to perform any storage update.

$$\mathbf{S}_n^{(t)}=\mathbf{S}_n^{(t-1)}, \qquad \forall n\in\mathcal{S}_w^{(t)}. \tag{7.8}$$

Next, let us formalize the security and privacy constraints. T-privacy guarantees that at any time, any set of up to T colluding servers learn nothing about the indices $(\theta_1,\theta_2,\cdots,\theta_t)$ from all the read and write queries and storage states.

$$\begin{aligned}&[T\text{-Privacy}]I\left((\theta_\tau)_{\tau\in[t]};\left(P_n^{(t,\theta_t)},Q_n^{(t,\theta_t)}\right)_{n\in\mathcal{T}}\;\middle|\;\left(\mathbf{S}_n^{(\tau-1)},P_n^{(\tau-1,\theta_{\tau-1})},Q_n^{(\tau-1,\theta_{\tau-1})}\right)_{\tau\in[t],n\in\mathcal{T}}\right)=0,\\&\qquad\qquad\forall\mathcal{T}\in[N],|\mathcal{T}|=T,t\in\mathbb{N},\end{aligned} \tag{7.9}$$

where for all $n\in[N]$, we define $P_n^{(0,\theta_0)}=Q_n^{(0,\theta_0)}=.$

Similarly, any set of up to X_Δ colluding servers must learn nothing about the increments $(\mathbf{\Delta}_1,\mathbf{\Delta}_2,\cdots,\mathbf{\Delta}_t)$.

$$[X_\Delta\text{-Security}]\quad I\left((\mathbf{\Delta}_\tau)_{\tau\in[t]};\left(P_n^{(t,\theta_t)}\right)_{n\in\mathcal{X}}\;\middle|\;\left(\mathbf{S}_n^{(\tau-1)},P_n^{(\tau-1,\theta_{\tau-1})},Q_n^{(\tau,\theta_\tau)}\right)_{\tau\in[t],n\in\mathcal{X}}\right)=0,$$

[5] For ease of exposition we will assume in the description of the scheme that during the read phase, the read queries are sent to *all* servers that are available during the read phase, or will become available later during the write phase, so that only the write-queries need to be sent during the write phase.

$$\forall \mathcal{X} \subset [N], |\mathcal{X}| = X_{\Delta}, t \in \mathbb{N}. \tag{7.10}$$

The storage at the N servers is formalized according to a threshold secret sharing scheme. Specifically, the storage at any set of up to X colluding servers must reveal nothing about the submodels. Formally,

$$H\left(\mathbf{S}_n^{(0)} \;\middle|\; \left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}, \mathcal{Z}_S\right) = 0, \forall n \in [N], \tag{7.11}$$

$$[X\text{-Security}] \quad I\left(\left(\mathbf{W}_k^{(t)}\right)_{k\in[K]}; \left(\mathbf{S}_n^{(t)}\right)_{n\in\mathcal{X}}\right) = 0, \forall \mathcal{X} \subset [N], |\mathcal{X}| = X, t \in \mathbb{Z}^*, \tag{7.12}$$

where for all $n \in [N]$, we define $\mathbf{S}_n^{-1} = .$ Note that $\mathcal{Z}_S$ is the private randomness used by the secret sharing scheme that implements X-secure storage across the N servers.

There is a subtle difference in the security constraint that we impose on the storage, and the previously specified security and privacy constraints on updates and queries. To appreciate this difference let us make a distinction between the notions of an internal adversary and an external adversary. We say that a set of colluding servers forms an internal adversary if those colluding servers have access to not only their current storage, but also their entire history of previous stored values and queries. Essentially the internal adversary setting represents a greater security threat because the servers themselves are dishonest and surreptitiously keep records of all their history in an attempt to learn from it. In contrast, we say that a set of colluding servers forms an external adversary if those colluding servers have access to only their current storage, but not to other historical information. Essentially, this represents an external adversary who is able to steal the current information from honest servers who do not keep records of their historical information. Clearly, an external adversary is weaker than an internal adversary. Now let us note that while the T-private queries and the X_{Δ}-secure updates are protected against *internal* adversaries, the X-secure storage is only protected against *external* adversaries. This is mainly because we will generally assume

$X > \max(X_\Delta, T)$, i.e., a higher security threshold for storage, than for updates and queries. Note that once the number of compromised servers exceeds $\max(X_\Delta, T)$, the security of updates and the privacy of queries is no longer guaranteed. In such settings the security of storage is still guaranteed, albeit in a weaker sense (against external adversaries). On the other hand if the number of compromised servers is small enough, then indeed secure storage may be guaranteed in a stronger sense, even against internal adversaries. We refer the reader to Remark 7.5 for further insight into this aspect.

The independence among various quantities of interest is specified for all $t \in \mathbb{N}$ as follows.

$$\begin{aligned}&H\left(\left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}, (\boldsymbol{\Delta}_\tau)_{\tau\in[t]}, (\theta_\tau)_{\tau\in[t]}, \left(\mathcal{Z}_U^{(\tau)}\right)_{\tau\in[t]}, \mathcal{Z}_S\right)\\ &= H\left(\left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}\right) + H\left((\boldsymbol{\Delta}_\tau)_{\tau\in[t]}\right) + H\left((\theta_\tau)_{\tau\in[t]}\right) + H\left(\left(\mathcal{Z}_U^{(\tau)}\right)_{\tau\in[t]}\right) + H(\mathcal{Z}_S).\end{aligned} \tag{7.13}$$

To evaluate the performance of a robust XSTPFSL scheme, we consider the following metrics. The first two metrics focus on communication cost. For $t \in \mathbb{N}$, the download cost D_t is the expected (over all realizations of queries) number of q-ary symbols downloaded by User t, normalized by L. The upload cost U_t is the expected number of q-ary symbols uploaded by User t, also normalized by L. The next metric focuses on storage efficiency. For $t \in \mathbb{Z}^*$, the storage efficiency is defined as the ratio of the total data content to the total storage resources consumed by a scheme, i.e., $\eta^{(t)} = \frac{KL}{\sum_{n\in[N]} H\left(S_n^{(t)}\right)}$. If $\eta^{(t)}$ takes the same value for all $t \in \mathbb{Z}^*$, we use the compact notation η instead.

7.3 Main Result: The ACSA-RW Scheme for Private Read/Write

The main contribution of this chapter is the ACSA-RW scheme, which allows private read and private write from N distributed servers according to the problem statement provided in Section 7.2. The scheme achieves storage efficiency K_c/N, i.e., it uses a total storage of $(KLN/K_c)\log_2 q$ bits across N servers in order to store the $KL\log_2 q$ bits of actual data, where $K_c \in \mathbb{N}$, and allows arbitrary read-dropouts and write-dropouts as long as the number of dropout servers is less than the corresponding threshold values. The thresholds are defined below and their relationship to redundant storage dimensions is explained in Section 7.3.1.

$$\textbf{Read-dropout threshold:} \qquad S_r^{\text{thresh}} \triangleq N-(K_c+X+T-1). \tag{7.14}$$

$$\textbf{Write-dropout threshold:} \qquad S_w^{\text{thresh}} \triangleq X-(X_\Delta+T-1). \tag{7.15}$$

It is worth emphasizing that the ACSA-RW scheme does not just tolerate dropout servers, it adapts (hence the *elastic* resilience) to the number of available servers so as to reduce its communication cost. The elasticity would be straightforward if the only concern was the private-read operation because known PIR schemes can be adapted to the number of available servers. What makes the elasticity requirement non-trivial is that the scheme must accommodate both private read and private write. The private-write requirements are particularly intriguing, almost paradoxical in that the coded storage across all N servers needs to be updated to be consistent with the new submodels, even though some of those servers (the write-dropout servers) are unavailable, so their stored information cannot be changed. Furthermore, as server states continue to change over time, future updates need no knowledge of prior dropout histories. Also of interest are the tradeoffs between storage redundancy and the resources needed for private-read and private-write functionalities. Because many of these aspects become more intuitively transparent when $L \gg K \gg 1$, the asymptotic setting is used to present the main result in Theorem 7.1. In particular, by suppressing minor terms (which can be found in the full description of the scheme), the asymptotic setting reveals an elegant symmetry between the upload and download costs. The remainder of this section is devoted to stating and then understanding the implications of Theorem 7.1. The scheme itself is presented in the form of the proof of Theorem 7.1 in Section 7.4.

Theorem 7.1. *In the limit $L/K\to\infty$, for all $t\in\mathbb{N}$ the ACSA-RW scheme achieves the following download, upload cost pair (D_t,U_t) and storage efficiency η:*

$$(D_t,U_t)=\left(\frac{N-|\mathcal{S}_r^{(t)}|}{S_r^{\text{thresh}}-|\mathcal{S}_r^{(t)}|},\ \frac{N-|\mathcal{S}_w^{(t)}|}{S_w^{\text{thresh}}-|\mathcal{S}_w^{(t)}|}\right), \qquad \eta=\frac{K_c}{N}, \tag{7.16}$$

for any $K_c\in\mathbb{N}$ such that $|\mathcal{S}_r^{(t)}|<S_r^{\text{thresh}}$, $|\mathcal{S}_w^{(t)}|<S_w^{\text{thresh}}$, and the field size $q\geq N+\max\left\{S_r^{\text{thresh}},S_w^{\text{thresh}},K_c\right\}$.

7.3.1 Observations

Storage Redundancy and Private Read/Write Thresholds

The relationship between redundant storage dimensions and the private read/write thresholds is conceptually illustrated in Figure 7.2.

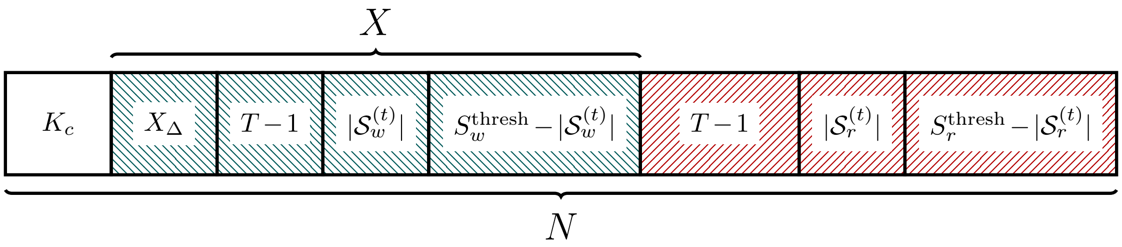

X
K_c
X_Δ
$T-1$
$|\mathcal{S}_w^{(t)}|$
$S_w^{\text{thresh}}-|\mathcal{S}_w^{(t)}|$
$T-1$
$|\mathcal{S}_r^{(t)}|$
$S_r^{\text{thresh}}-|\mathcal{S}_r^{(t)}|$
N

Figure 7.2: Conceptual partitioning of total server storage space (N dimensions) into data content (K_c dimensions), storage redundancy that is exploited by private-write (X dimensions), and storage redundancy that is exploited by private-read ($N-K_c-X$ dimensions).

The total storage utilized by the ACSA-RW scheme across N servers is represented as the overall N dimensional space in Figure 7.2. Out of this, the actual data occupies only K_c dimensions, which is why the storage efficiency of ACSA-RW is K_c/N. Because the storage must be X-secure, i.e., any set of up to X colluding servers cannot learn anything about the data, it follows that out of the N dimensions of storage space, X dimensions are occupied by information that is independent of the actual data. The storage redundancy represented by these X dimensions will be essential to enable the private-write functionality. But first let us consider the private-read operation for which we have a number of prior results on PIR as baselines for validating our intuition. From Figure 7.2 we note that outside the X dimensions of redundancy that was introduced due to data-security, the T-privacy constraint adds a storage redundancy of another $T-1$ dimensions that is shown in red. To understand this, compare the asymptotic (large K) capacity of MDS-PIR [11]: $C_{\text{MDS-PIR}}=1-K_c/N$ with the (conjectured) asymptotic capacity of MDS-TPIR [36, 105]: $C_{\text{MDS-TPIR}}=1-(K_c+T-1)/N$. To achieve a non-zero value of the asymptotic capacity, the former requires $N>K_c$, but the latter requires $N>K_c+T-1$. Equivalently, the former allows a read-dropout threshold of $N-K_c$ while the latter allows a read-dropout threshold of $N-(K_c+T-1)$. In fact, going further to the (conjectured) asymptotic capacity of MDS-XSTPIR [54], which also includes the X-security constraint, we note that a non-zero value of capacity requires $N>K_c+X+T-1$. Intuitively, we may interpret this as: the X-security constraint increases the demands on storage redundancy by X dimensions and the T-privacy constraint increases the demands on storage redundancy by another $T-1$ dimensions. This is what is represented in Figure 7.2. Aside from the K_c dimensions occupied by data, the X dimensions of redundancy added by the security constraint, and the $T-1$ dimensions of redundancy added by the T-privacy constraint, the remaining dimensions at the right end of the figure are used to accommodate read-dropouts. Indeed, this is what determines the read-dropout threshold, as we note from Figure 7.2 that $N-(X+K_c+(T-1))=S_r^{\text{thresh}}$. Now consider the private-write operation which is novel and thus lacks comparative baselines. What is remarkable is the synergistic aspect of private-write, that it does not add further redundancy beyond the

X dimensions of storage redundancy already added by the X-security constraint. Instead, it operates within these X dimensions to create further sub-partitions. Within these X dimensions, a total of $X_\Delta + T - 1$ dimensions are used to achieve X_Δ-secure updates that also preserve T-privacy, and the remaining dimensions are used to accommodate write-dropout servers, giving us the write-dropout threshold as $X - (X_\Delta + T - 1) = S_w^{\text{thresh}}$. Remarkably in Figure 7.2, the $(K_c, X_\Delta, T-1, S_w^{\text{thresh}})$ partition structure for private-write replicates at a finer level the original $(K_c, X, T-1, S_r^{\text{thresh}})$ partition structure of private-read. Also remarkable is a new constraint introduced by the private-write operation that is not encountered in prior works on PIR — the feasibility of ACSA-RW requires $X \geq T$. Whether or not this constraint is fundamental in the asymptotic setting of large K is an open problem for future work.

Optimality

Asymptotic (large K) optimality of ACSA-RW remains an open question in general. Any attempt to resolve this question runs into other prominent open problems in the information-theoretic PIR literature, such as the asymptotic capacity of MDS-TPIR [36, 105] and MDS-XSTPIR in Chapter 4 that also remain open. Nevertheless, it is worth noting that Theorem 7.1 matches or improves upon the best known results in all cases where such results are available. In particular, the private-read phase of ACSA-RW scheme recovers a universally robust X-secure T-private information retrieval scheme (see [16] and Chapter 2). When $K_c = 1, X \geq X_\Delta + T$, it achieves the asymptotic capacity; and when $K_c > 1, X \geq X_\Delta + T$, it achieves the conjectured asymptotic capacity in Chapter 4. While much less is known about optimal private-write schemes, it is clear that ACSA-RW significantly improves upon previous work as explained in Section 7.3.1. Notably, both upload and download costs are $O(1)$ in K, i.e., they do not scale with K. Thus, at the very least the costs are orderwise optimal. Another interesting point of reference is the best case scenario, where we have no dropout servers, $|\mathcal{S}_r^{(t)}| = 0, |\mathcal{S}_w^{(t)}| = 0$. The total communication cost of ACSA-RW in this case, i.e., the sum of upload and download costs, is $D_t+U_t=N\left(\frac{1}{S_r^{\text{thresh}}}+\frac{1}{S_w^{\text{thresh}}}\right)$, which is minimized when $K_c=1$, $S_r^{\text{thresh}}=S_w^{\text{thresh}}$, and $X=(N+X_\Delta-1)/2$. For large N, we have $D_t+U_t\approx 2+2=4$, thus in the best case scenario, ACSA-RW is optimal within a factor of 2. Finally, on a speculative note, perhaps the most striking aspect of Theorem 7.1 is the symmetry between upload and download costs, which (if not coincidental) bodes well for their fundamental significance and information theoretic optimality.

The Choice of Parameter K_c

The choice of the parameter K_c in ACSA-RW determines the storage efficiency of the scheme, $\eta=K_c/N$. At one extreme, we have the smallest possible value of K_c, i.e., $K_c=1$, which is the least efficient storage setting, indeed the storage efficiency is analogous to replicated storage, each server uses as much storage space as the size of all data ($KL\log_2 q$ bits). This setting yields the best (smallest) download costs. The other extreme corresponds to the maximum possible value of K_c, which is obtained as $K_c=N-(X+T)$ because the dropout thresholds cannot be smaller than 1. At this extreme, storage is the most efficient, but there is no storage redundancy left to accommodate any read dropouts, and the download cost of ACSA-RW takes its maximal value, equal to N. Remarkably, the upload cost of the private-write operation does not depend on K_c. However, K_c is significant for another reason; it determines the access complexity (see Remark 7.6) of both private read and write operations, i.e., the number of bits that are read from or written to by each available server. In particular, the access complexity of each available server in the private read or write phases is at most $(KL/K_c)\log_2 q$, so for example, increasing K_c from 1 to 2 can reduce the access complexity in half, while simultaneously doubling the storage efficiency.

Tradeoff between Upload and Download Costs

The trade-off between the upload cost and the download cost of the ACSA-RW scheme is illustrated via two examples in Figure 7.3, where we have $N=10, X_\Delta=T=1$ for the blue solid curve, and $N=10, X_\Delta=1, T=2$ for the red solid curve. For both examples, we set $K_c=1$ and assume that there are no dropout servers. The trade-off is achieved with various choices of X. For the example shown in the blue solid curve, we set $X=(2,3,4,5,6,7,8)$. For the example in the red solid curve, we set $X=(3,4,5,6,7)$. Note that the most balanced trade-off point is achieved when $X=N/2$.

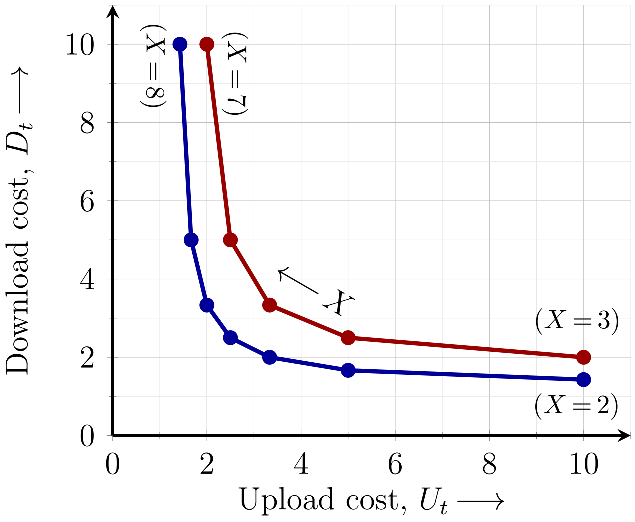


Figure 7.3: Upload, download costs pairs (U_t, D_t) of the ACSA-RW scheme in the asymptotic setting $L \gg K \gg 1$, for $N=10, X_\Delta=T=1$ (the blue curve) and $N=10, X_\Delta=1, T=2$ (the red curve), with various choices of X. Both examples assume that $K_c=1$ and there are no dropout servers.

Synergistic Gains from Joint Design of Private Read and Write

A notable aspect of the ACSA-RW scheme, which answers in the affirmative an open question raised in Section 4.4.4 of [80], is the synergistic gain from the joint design of the read phase and the write phase. While the details of the scheme are non-trivial and can be found in

Section 7.4, let us provide an intuitive explanation here by ignoring some of the details. As a simplification, let us ignore security constraints and consider the database represented by a vector $\mathbf{W}=[W_1,W_2,\cdots,W_K]^{\mathsf{T}}$ that consists of symbols from the K submodels. Suppose the user is interested in W_{θ_t} for some index $\theta_t\in[K]$ that must be kept private. Note that $W_{\theta_t}=\mathbf{W}^{\mathsf{T}}\mathbf{e}_K(\theta_t)$, where $\mathbf{e}_K(\theta_t)$ is the standard basis vector, i.e., the desired symbol is obtained as an inner product of the database vector and the basis vector $\mathbf{e}_K(\theta_t)$. To do this privately, the basis vector is treated by the user as a secret and a linear threshold secret-sharing scheme is used to generate shares that are sent to the servers. The servers return the inner products of the stored data and the secret-shared basis vector, which effectively form the secret shares of the desired inner product. Once the user collects sufficiently many secret shares, (s)he is able to retrieve the desired inner product, and therefore the desired symbol W_{θ_t}. This is the key to the private read-operation. Now during the write phase, the user wishes to update the database to the new state: $\mathbf{W}'=[W_1,\cdots,W_{\theta_t-1},W_{\theta_t}+\Delta_t,W_{\theta_t+1}\cdots,W_K]$, which can be expressed as $\mathbf{W}'=\mathbf{W}+\Delta_t\mathbf{e}_K(\theta_t)$. This can be accomplished by sending the secret-shares of $\Delta_t\mathbf{e}_K(\theta_t)$ to the N servers. The key observation here is the following: since the servers (those that were available during the read phase) have already received secret shares of $\mathbf{e}_K(\theta_t)$, the cost of sending the secret-shares of $\Delta_t\mathbf{e}_K(\theta_t)$ is significantly reduced. Essentially, it suffices to send secret shares of Δ_t which can be multiplied with the secret shares of $\mathbf{e}_K(\theta_t)$ to generate secret shares of $\Delta_t\mathbf{e}_K(\theta_t)$ at the servers. This is much more efficient because $\Delta_t\mathbf{e}_K(\theta_t)$ is a $K\times 1$ vector, while the dimension of Δ_t is 1 (scalar), and $K\gg 1$. This is the intuition behind the synergistic gains from the joint design of private read and write operations that are exploited by ACSA-RW. Note that the servers operate directly on secret shares, as in homomorphic encryption [27], and that these operations (inner products) are special cases of secure distributed matrix multiplications. Since CSA codes have been shown to be natural solutions for secure distributed matrix multiplications (see Chapter 6 and [55]), it is intuitively to be expected that CSA schemes should lead to communication-efficient solutions for the private read-write implementation as described above.

How to Fully Update a Distributed Database that is only Partially Accessible

A seemingly paradoxical aspect of the write phase of ACSA-RW is that it is able to force the distributed database across all N servers to be fully consistent with the updated data, even though the database is only partially accessible due to write-dropout servers. Let us explain the intuition behind this with a toy example[6] where we have $N=2$ servers and $X=1$ security level is required. For simplicity we have only one file/submodel, i.e., $K=1$, so there are no privacy concerns. The storage at the two servers is $S_1=W+Z$, $S_2=Z$, where W is the data (submodel) symbol, and Z is the random noise symbol used to guarantee the $X=1$ security level, i.e., the storage at each server individually reveals nothing about the data W. The storage can be expressed in the following form.

$$\begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} W \\ Z \end{bmatrix}. \tag{7.17}$$

so that $\mathbf{G}$ is the coding function, and the data can be recovered as $S_1-S_2=W$.

Now suppose the data W needs to be updated to the new value $W'=W+\Delta$. The updated storage S_1',S_2' should be such that the coding function is unchanged (still the same $\mathbf{G}$ matrix) and the updated data can similarly be recovered as $S_1'-S_2'=W'$. Remarkably this can be done even if one of the two servers drops out and is therefore inaccessible. For example, if Server 1 drops out, then we can update the storage only at Server 2 to end up with $S_1'=S_1=W+Z$, and $S_2'=Z-\Delta$, such that indeed $S_1'-S_2'=W'$ and the coding function is

[6] The toy example is not strictly a special case of the ACSA-RW scheme, because the number of servers $N=2$ is too small to guarantee any privacy. However, the example serves to demonstrate the key idea.

unchanged.

$$\begin{bmatrix} S_1' \\ S_2' \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W+\Delta \\ Z-\Delta \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W' \\ Z' \end{bmatrix}. \tag{7.18}$$

Similarly, if Server 2 drops out, then we can update the storage only at Server 1 to end up with $S_1' = W+\Delta+Z$, and $S_2' = S_2 = Z$, such that we still have $S_1' - S_2' = W'$ and the coding function is unchanged.

$$\begin{bmatrix} S_1' \\ S_2' \end{bmatrix} = \begin{bmatrix} S_1' \\ S_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W+\Delta \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W' \\ Z \end{bmatrix}. \tag{7.19}$$

This, intuitively, is how the paradox is resolved, and a distributed database is fully updated even when it is only partially accessible. Note that the realization of the "noise" symbol is different for various realizations of the dropout servers, $Z' \neq Z$.

While the toy example conveys a key idea, the generalization of this idea to the ACSA-RW scheme is rather non-trivial. Let us shed some light on this generalization, which is made possible by the construction of an "*ACSA null-shaper*", see Definition 7.8. Specifically, by carefully placing nulls of the CSA code polynomial in the update equation, the storage of the write-dropout servers in $\mathcal{S}_w^{(t)}$ is left unmodified. It is important to point out that the storage structure (i.e., ACSA storage, see Definition 7.3) is preserved, just as the coding function is left unchanged in the toy example above. Let us demonstrate the idea with a minimal example where $X=2, T=1, X_\Delta=0, N=4$. Let us define the following functions.

$$\mathbf{S}(\alpha) = \mathbf{W} + \alpha\mathbf{Z}_1 + \alpha^2\mathbf{Z}_2, \tag{7.20}$$

$$\mathbf{Q}(\alpha) = \mathbf{e}_K(\theta_t) + \alpha\mathbf{Z}', \tag{7.21}$$

where $\mathbf{W} = [W_1, W_2, \cdots, W_K]$ is a $K \times 1$ vector of the K data (submodel) symbols, $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}'$

are uniformly and independently distributed noise vectors that are used to protect data security and the user's privacy, respectively. Let $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ be 4 distinct non-zero elements from a finite field $\mathbb{F}_q$. The storage at the 4 servers is $\mathbf{S}(\alpha_1)$, $\mathbf{S}(\alpha_2)$, $\mathbf{S}(\alpha_3)$, $\mathbf{S}(\alpha_4)$, respectively. Similarly, the read-queries for the 4 servers are $\mathbf{Q}(\alpha_1)$, $\mathbf{Q}(\alpha_2)$, $\mathbf{Q}(\alpha_3)$, $\mathbf{Q}(\alpha_4)$, respectively. We note that the storage vectors and the query vectors can be viewed as secret sharings of $\mathbf{W}$ and $\mathbf{e}_K(\theta_t)$ vectors with threshold of $X=2$ and $T=1$, respectively. Now let us assume that $\mathcal{S}_w^{(t)}=\{1\}$ for some $t\in\mathbb{N}$, i.e., Server 1 drops out. Let us define the function $\Omega(\alpha)=(\alpha_1-\alpha)/\alpha_1$, which is referred to as *ACSA null-shaper*, and consider the following update equation.

$$\mathbf{S}'(\alpha)=\mathbf{S}(\alpha)+\Omega(\alpha)\Delta_t\mathbf{Q}(\alpha). \tag{7.22}$$

Inspecting the second term on the RHS, we note that

$$\begin{aligned}
\Omega(\alpha)\Delta_t\mathbf{Q}(\alpha)&=\frac{1}{\alpha_1}(\alpha_1-\alpha)\Delta_t(\mathbf{e}_K(\theta_t)+\alpha\mathbf{Z}') && (7.23)\\
&=\frac{1}{\alpha_1}\Delta_t\big(\alpha_1\mathbf{e}_K(\theta_t)+\alpha(\alpha_1\mathbf{Z}'-\mathbf{e}_K(\theta_t))-\alpha^2\mathbf{Z}'\big) && (7.24)\\
&=\Delta_t\big(\mathbf{e}_K(\theta_t)+\alpha(\mathbf{Z}'-\alpha_1^{-1}\mathbf{e}_K(\theta_t))-\alpha^2\alpha_1^{-1}\mathbf{Z}'\big), && (7.25)\\
&=\Delta_t\Big(\mathbf{e}_K(\theta_t)+\alpha\dot{\mathbf{I}}_1-\alpha^2\dot{\mathbf{I}}_2\Big), && (7.26)
\end{aligned}$$

where $\dot{\mathbf{I}}_1=\mathbf{Z}'-\alpha_1^{-1}\mathbf{e}_K(\theta_t)$ and $\dot{\mathbf{I}}_2=\alpha_1^{-1}\mathbf{Z}'$. Evidently, by the update equation, the user is able to update the symbol of the θ_t^{th} message with the increment Δ_t, while maintaining the storage as a secret sharing of threshold 2, i.e.,

$$\mathbf{S}'(\alpha)=(\mathbf{W}+\Delta_t\mathbf{e}_k(\theta_t))+\alpha(\mathbf{Z}_1+\Delta_t\dot{\mathbf{I}}_1)+\alpha^2(\mathbf{Z}_2+\Delta_t\dot{\mathbf{I}}_2). \tag{7.27}$$

However, by the definition of ACSA null-shaper, we have $\Omega(\alpha_1)=0$. Thus $\mathbf{S}'(\alpha_1)=\mathbf{S}(\alpha_1)$, and we do not have to update the storage at the Server 1. In other words, $\dot{\mathbf{I}}_1$ and $\dot{\mathbf{I}}_2$ are *artificially correlated interference symbols* such that the codeword $\Omega(\alpha_1)\mathbf{Q}(\alpha_1)$ is zero, and accordingly, the storage of Server 1 is left unmodified. Note that ACSA null-shaper does not affect the storage structure because $X=2>T=1$. The idea illustrated in this minimal example indeed generalizes to the full ACSA-RW scheme, see Section 7.4 for details.

Comparison with [64]

Let us compare our ACSA-RW solution with that in [64]. The setting in [64] corresponds to $X=0,T=1,X_\Delta=0$, and $|\mathcal{S}_w^{(t)}|=|\mathcal{S}_r^{(t)}|=0$ for all $t\in\mathbb{N}$. Note that our ACSA-RW scheme for $X=1,T=1,X_\Delta=0$ and $K_c=1$ applies to the setting of [64] ($X=1$ security automatically satisfies $X=0$ security). To make the comparison more transparent, let us briefly review the construction in [64], where at any time $t,t\in\mathbb{Z}^*$, each of the N servers stores the K submodels in the following coded form.

$$\mathbf{W}_k^{(t-1)}+z_k^{(t)}\mathbf{\Delta}_t,\forall k\in[K]. \tag{7.28}$$

For all $t\in\mathbb{N}$, $\left(z_k^{(t)}\right)_{k\in[K]}$ are distinct random scalars generated by User t and we set $z_{\theta_t}^{(t)}=1$. For completeness we define $\mathbf{\Delta}_0=\mathbf{0},z_k^{(0)}=0,\mathbf{W}_k^{(-1)}=\mathbf{W}_k^{(0)},\forall k\in[K]$. In addition, the N servers store the random scalars $\left(z_k^{(t)}\right)_{k\in[K]}$, as well as the increment $\mathbf{\Delta}_t$ according to a secret sharing scheme of threshold 1. In the retrieval phase, User t retrieves the coded desired submodel $\mathbf{W}_{\theta_t}^{(t-2)}+z_{\theta_t}^{(t-1)}\mathbf{\Delta}_{t-1}$ privately according to a capacity-achieving replicated storage based PIR scheme, e.g., [24]. Besides, User t also downloads the secret shared random scalars $\left(z_k^{(t-1)}\right)_{k\in[K]}$ and the increment $\mathbf{\Delta}_{t-1}$ to correctly recover the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$. In the update phase, User t uploads to each of the N servers the following update vectors $\mathbf{P}_k^{(t)}$ for all $k\in[K]$.

$$\mathbf{P}_k^{(t)}=\begin{cases} z_k^{(t)}\mathbf{\Delta}_t-z_k^{(t-1)}\mathbf{\Delta}_{t-1}, & k\neq\theta_{t-1}, \\ z_k^{(t)}\mathbf{\Delta}_t, & k=\theta_{t-1}. \end{cases} \tag{7.29}$$

Also, User t uploads the secret shared random scalars $\left(z_k^{(t)}\right)_{k\in[K]}$ and the increment $\mathbf{\Delta}_t$ to the N servers. To perform an update, each of the N servers updates all of the submodels $k\in[K]$ according to the following equation.

$$\left(\mathbf{W}_k^{(t-2)}+z_k^{(t-1)}\mathbf{\Delta}_{t-1}\right)+\mathbf{P}_k^{(t)}=\mathbf{W}_k^{(t-1)}+z_k^{(t)}\mathbf{\Delta}_t. \tag{7.30}$$

Perhaps the most significant difference between our ACSA-RW scheme and the construction in [64] is that the latter does not guarantee the privacy of successive updates, i.e., by monitoring the storage at multiple time slots, the servers are eventually able to learn about the submodel indices from past updates[7]. On the other hand, our construction guarantees information theoretic privacy for an unlimited number of updates, without extra storage overhead. Furthermore, we note that the normalized download cost achieved by the construction in [64] cannot be less than 2, whereas ACSA-RW achieves download cost of less than 2 with large enough N. For the asymptotic setting $L/K\to\infty$, the upload cost[8] achieved by [64] is at least $2N+1$, while ACSA-RW achieves the upload cost of at most N. The lower bound of upload cost of XSTPFSL is characterized in [64] as NK. However, our construction of ACSA-RW shows that it is possible to do better.[9] In particular, for the asymptotic setting $K\to\infty, L/K\to\infty$, the upload cost of less than N is achievable by the ACSA-RW scheme.

[7] This is because the update vectors (7.29) for the K submodels at any time t can be viewed as $\mathcal{P}_t=\text{span}\{\mathbf{\Delta}_t,\mathbf{\Delta}_{t-1}\}$. For any two consecutive time slots t and $t+1$, it is possible to determine $\text{span}\{\mathbf{\Delta}_t\}=\text{span}\{\mathbf{\Delta}_t,\mathbf{\Delta}_{t-1}\}\cap\text{span}\{\mathbf{\Delta}_{t+1},\mathbf{\Delta}_t\}=\mathcal{P}_t\cap\mathcal{P}_{t+1}$. Due to the fact that for User t, the update vector for the θ_{t-1}^{th} submodel only lies in $\text{span}\{\mathbf{\Delta}_t\}$, any curious server is able to obtain information about θ_{t-1} from $\mathcal{P}_t$ if $\mathbf{\Delta}_t$ is linearly independent of $\mathbf{\Delta}_{t-1}$.

[8] In [64] the achieved upload cost is $NLK+L+K$. However, in terms of average upload compression, e.g., by entropy encoding, allows lower upload cost. For example, in the asymptotic setting $L/K\to\infty$, the upload cost of $(2N+1+1/(N-1))$ may be achievable.

[9] It is assumed in [64] that for the desired submodel, the user uploads L symbols for the update, while for other submodels, the user should also upload $(K-1)L$ symbols to guarantee the privacy. However, it turns out that the uploaded symbols for the update of the desired submodel and the symbols for the purpose of guaranteeing the privacy do not have to be independent.

Comparison with [66]

Let us also briefly review the 4-server information-theoretic DORAM construction in [66] to see how our ACSA-RW scheme improves upon it. Note that the setting considered in [66] is a special case of our problem where $X=1, T=1, X_\Delta=0, K_c=1$ and $|\mathcal{S}_w^{(t)}|=|\mathcal{S}_r^{(t)}|=0$ for all $t\in\mathbb{N}$. First, we note that in the asymptotic setting $L/K\to\infty$, the upload cost achieved by the ACSA-RW is the same as that in [66]. Therefore, for this comparison we focus on the read phase and the download cost. Specifically, the information-theoretic DORAM construction in [66] partitions the four servers into two groups, each of which consists of 2 servers. For the retrieval phase, the first group emulates a 2-server PIR, storing the K submodels secured with additive random noise, i.e., $\mathbf{W}+\mathbf{Z}$. The second group emulates another 2-server PIR storing the random noise $\mathbf{Z}$. To retrieve the desired submodel privately, the user exploits a PIR scheme to retrieve the desired secured submodel, as well as the corresponding random noise. Therefore, with capacity-achieving PIR schemes, the download cost is 4 (for large K). On the other hand, our ACSA-RW scheme avoids the partitioning of the servers and improves the download cost by jointly exploiting all 4 servers. Remarkably, with the idea of cross-subspace alignment, out of the 4 downloaded symbols, the interference symbols align within 2 dimensions, leaving 2 dimensions interference-free for the desired symbols, and consequently, the asymptotically optimal download cost of 2 is achievable. Lastly, the ACSA-RW scheme also generalizes efficiently to arbitrary numbers of servers.

On the Assumption $L\gg K\gg 1$ for FSL

Finally, let us briefly explore the practical relevance of the asymptotic limits $K\to\infty, L/K\to\infty$, with an example. Suppose we have $N=6$ distributed servers, we require security and privacy levels of $X_\Delta=T=1, X=3$. Let us set $K_c=1$, and we operate over $\mathbb{F}_8$. Consider an e-commerce recommendation application similar to what is studied in [82], where a global model with a total of 3,500,000 symbols (from $\mathbb{F}_8$) is partitioned into $K=50$ submodels. Each of the submodels is comprised of $L=70{,}000$ symbols. Note that $L \gg K \gg 1$. Let us assume that there are no dropout servers for some $t \in \mathbb{N}$. Now according to Lemma 7.2, the normalized upload cost achieved by the ACSA-RW scheme is $U_t = \frac{6\times 35000+6\times 2\times 50}{70{,}000} \approx 3.00857$. On the other hand, the normalized download cost achieved is $D_t = 6/2 = 3$. Evidently, the asymptotic limits $K \to \infty, L/K \to \infty$ are fairly accurate for this non-asymptotic setting. For this particular example, the upload cost is increased by only 0.29% compared to the asymptotic limit. On the other hand, the download cost is increased by only $1.4 \times 10^{-22}\%$ compared to the lower bound (evaluated for $K=50$) from Theorem 2.1.

7.4 Proof of Theorem 7.1

For all $t \in \mathbb{N}$, we require that the number of read and write dropout servers is less than the corresponding threshold values, $0 \leq |\mathcal{S}_r^{(t)}| < S_r^{\text{thresh}}$ and $0 \leq |\mathcal{S}_w^{(t)}| < S_w^{\text{thresh}}$. Since the read and write dropout thresholds cannot be less than 1, we require that $X \geq X_\Delta + T$, and $N \geq K_c + X + T$ for a positive integer K_c. Let us define $J = \xi \cdot \text{lcm}\big([S_r^{\text{thresh}}] \cup [S_w^{\text{thresh}}]\big)$ and we set $L = JK_c$, where ξ is a positive[10] integer. In other words, $i \mid J$ for all $i \in [S_r^{\text{thresh}}] \cup [S_w^{\text{thresh}}]$. For ease of reference, let us define $R_r^{(t)} \triangleq S_r^{\text{thresh}} - |\mathcal{S}_r^{(t)}|$, $\#_r^{(t)} \triangleq J/R_r^{(t)}$, $R_w^{(t)} \triangleq S_w^{\text{thresh}} - |\mathcal{S}_w^{(t)}|$ and $\#_w^{(t)} \triangleq J/R_w^{(t)}$ for all $t \in \mathbb{N}$. Note that it is guaranteed by the choice of J that $\#_r^{(t)}, \#_w^{(t)}$ are positive integers for all $t \in \mathbb{N}$. Indeed in the ACSA-RW scheme, private read and write operations can be viewed as operations that consist of $K_c\#_r^{(t)}$ and $K_c\#_w^{(t)}$ sub-operations, and in each sub-operation, $R_r^{(t)}$ and $R_w^{(t)}$ symbols of the desired submodel are retrieved and updated, respectively. The choice of J guarantees that the number of sub-operations is always an integer, regardless of $|\mathcal{S}_r^{(t)}|$ and $|\mathcal{S}_w^{(t)}|$. The parameter ξ guarantees that L is still a free parameter so that $L/K \to \infty$ is well-defined. In other words, L can be any

[10] The purpose of ξ is primarily to allow the scheme to scale to larger values of L, one could assume $\xi = 1$ for simplicity.

multiple of $K_c \cdot \mathrm{lcm}\big([S_r^{\text{thresh}}] \cup [S_w^{\text{thresh}}]\big)$. Let us define $\mu \triangleq \max(S_r^{\text{thresh}}, S_w^{\text{thresh}})$. We will need a total of $N+\max(\mu,K_c)$ distinct elements from the finite field $\mathbb{F}_q, q \geq N+\max(\mu,K_c)$, denoted as $(\alpha_1,\alpha_2,\cdots,\alpha_N)$, $(\widetilde{f}_1,\widetilde{f}_2,\cdots,\widetilde{f}_{\max(\mu,K_c)})$. Let us define the set $\overline{\mathcal{S}}_w^{(t)} = [N] \setminus \mathcal{S}_w^{(t)}$, and the set $\overline{\mathcal{S}}_r^{(t)} = [N] \setminus \mathcal{S}_r^{(t)}$. For all $t \in \mathbb{Z}^*, j \in [J], k \in [K], i \in [K_c]$, let us define

$$W_k^{(t)}(j,i) = W_k^{(t)}(i+K_c(j-1)), \tag{7.31}$$

i.e., the $L = JK_c$ symbols of each of the K messages are reshaped into a $J \times K_c$ matrix. Similarly, for all $t \in \mathbb{N}, j \in [J], i \in [K_c]$, we define

$$\Delta_{j,i}^{(t)} = \Delta_t(i+K_c(j-1)). \tag{7.32}$$

For all $t \in \mathbb{Z}^*, j \in [J], i \in [K_c]$, let us define the following vectors.

$$\dot{\mathbf{W}}_{j,i}^{(t)} = \Big[W_1^{(t)}(j,i), W_2^{(t)}(j,i), \cdots, W_K^{(t)}(j,i)\Big]^{\mathsf{T}}. \tag{7.33}$$

Further, let us set $\mathcal{Z}_s = \Big\{\dot{\mathbf{Z}}_{j,x}^{(0)}\Big\}_{j\in[J], x\in[X]}$, where $\dot{\mathbf{Z}}_{j,x}^{(0)}$ are i.i.d. uniform column vectors from $\mathbb{F}_q^K$, $\forall j \in [J], x \in [X]$. For all $t \in \mathbb{N}, j \in [J], x \in [X]$, let $\dot{\mathbf{Z}}_{j,x}^{(t)}$ be $K \times 1$ column vectors from the finite field $\mathbb{F}_q$. For all $t \in \mathbb{N}, u \in [\mu], i \in [K_c], s \in [T]$, let $\widetilde{\mathbf{Z}}_{u,i,s}^{(t)}$ be i.i.d. uniform column vectors from $\mathbb{F}_q^K$, and let us set $\mathcal{Z}_U^{(t)} = \Big\{\widetilde{\mathbf{Z}}_{u,i,s}^{(t)}\Big\}_{u\in[\mu], i\in[K_c], s\in[T]} \cup \Big\{\dddot{Z}_{\ell,i,x}^{(t)}\Big\}_{\ell\in[\#_w^{(t)}], i\in[K_c], x\in[X_\Delta]}$, where for all $t \in \mathbb{N}, \ell \in [\#_w^{(t)}], i \in [K_c], x \in [X_\Delta]$, $\dddot{Z}_{\ell,i,x}^{(t)}$ are i.i.d. uniform scalars from the finite field $\mathbb{F}_q$.

The ACSA-RW scheme (Definition 7.9) is built upon the elements introduced in Definitions 7.1–7.8.

Definition 7.1. (Pole Assignment) *If $\mu \geq K_c$, let us define the $\mu \times K_c$ matrix $\mathbf{F}$ to be* *the first K_c columns of the following $\mu \times \mu$ matrix.*

$$\begin{bmatrix} \widetilde{f}_1 & \widetilde{f}_\mu & \dots & \widetilde{f}_3 & \widetilde{f}_2 \\ \widetilde{f}_2 & \widetilde{f}_1 & \widetilde{f}_\mu & & \widetilde{f}_3 \\ \vdots & \widetilde{f}_2 & \widetilde{f}_1 & \ddots & \vdots \\ \widetilde{f}_{\mu-1} & & \ddots & \ddots & \widetilde{f}_\mu \\ \widetilde{f}_\mu & \widetilde{f}_{\mu-1} & \dots & \widetilde{f}_2 & \widetilde{f}_1 \end{bmatrix}. \tag{7.34}$$

On the other hand, if $\mu < K_c$, let us define the $\mu \times K_c$ matrix $\mathbf{F}$ to be the first μ rows of the following $K_c \times K_c$ matrix.

$$\begin{bmatrix} \widetilde{f}_1 & \widetilde{f}_2 & \dots & \widetilde{f}_{K_c-1} & \widetilde{f}_{K_c} \\ \widetilde{f}_{K_c} & \widetilde{f}_1 & \widetilde{f}_2 & & \widetilde{f}_{K_c-1} \\ \vdots & \widetilde{f}_{K_c} & \widetilde{f}_1 & \ddots & \vdots \\ \widetilde{f}_3 & & \ddots & \ddots & \widetilde{f}_2 \\ \widetilde{f}_2 & \widetilde{f}_3 & \dots & \widetilde{f}_{K_c} & \widetilde{f}_1 \end{bmatrix}. \tag{7.35}$$

Let $(f_{j,i})_{j\in[J],i\in[K_c]}$ be a total of JK_c elements from the finite field $\mathbb{F}_q$, which are defined as follows.

$$\begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,K_c} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,K_c} \\ \vdots & \vdots & \vdots & \vdots \\ f_{J,1} & f_{J,2} & \cdots & f_{J,K_c} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{F} \\ \vdots \\ \mathbf{F} \end{bmatrix}. \tag{7.36}$$

According to the definition, we have the following two propositions immediately.

Proposition 7.1. *For all $i \in [K_c], m, n \in [J]$ such that $m \leq n$, $|[m:n]| \leq \mu$, the constants $(f_{j,i})_{j\in[m:n]}$ are distinct.*

Proposition 7.2. *For all $j \in [J]$, the constants $(f_{j,i})_{i\in[K_c]}$ are distinct.*

Definition 7.2. (Noise Assignment) *For all $t\in\mathbb{N},i\in[K_c],s\in[T]$, let us define*

$$\begin{aligned}&\left(\ddot{\mathbf{Z}}_{1,i,s}^{(t)},\ddot{\mathbf{Z}}_{2,i,s}^{(t)},\cdots,\ddot{\mathbf{Z}}_{J,i,s}^{(t)}\right)\\&=\left(\widetilde{\mathbf{Z}}_{1,i,s}^{(t)},\widetilde{\mathbf{Z}}_{2,i,s}^{(t)},\cdots,\widetilde{\mathbf{Z}}_{\mu,i,s}^{(t)},\widetilde{\mathbf{Z}}_{1,i,s}^{(t)},\widetilde{\mathbf{Z}}_{2,i,s}^{(t)},\cdots,\widetilde{\mathbf{Z}}_{\mu,i,s}^{(t)},\cdots,\widetilde{\mathbf{Z}}_{1,i,s}^{(t)},\widetilde{\mathbf{Z}}_{2,i,s}^{(t)},\cdots,\widetilde{\mathbf{Z}}_{\mu,i,s}^{(t)}\right),\end{aligned}\tag{7.37}$$

i.e., $\left(\widetilde{\mathbf{Z}}_{1,i,s}^{(t)},\widetilde{\mathbf{Z}}_{2,i,s}^{(t)},\cdots,\widetilde{\mathbf{Z}}_{\mu,i,s}^{(t)}\right)$ are assigned in cyclic order to $\left(\ddot{\mathbf{Z}}_{1,i,s}^{(t)},\ddot{\mathbf{Z}}_{2,i,s}^{(t)},\cdots,\ddot{\mathbf{Z}}_{J,i,s}^{(t)}\right)$.

Definition 7.3. (ACSA Storage) *For any $t\in\mathbb{Z}^*$, the storage at the N servers is said to form an ACSA storage if for all $n\in[N]$, $\mathbf{S}_n^{(t)}$ has the following form.*

$$\mathbf{S}_n^{(t)}=\begin{bmatrix}\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{1,i}}\dot{\mathbf{W}}_{1,i}^{(t)}+\sum_{x\in[X]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{1,x}^{(t)}\\\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{2,i}}\dot{\mathbf{W}}_{2,i}^{(t)}+\sum_{x\in[X]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{2,x}^{(t)}\\\vdots\\\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{J,i}}\dot{\mathbf{W}}_{J,i}^{(t)}+\sum_{x\in[X]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{J,x}^{(t)}\end{bmatrix}.\tag{7.38}$$

Note that X i.i.d. uniform random noise terms are MDS coded to guarantee the X-security.

Definition 7.4. (ACSA Query) *For any $t\in\mathbb{N}$, the read-queries $\left(Q_n^{(t,\theta_t)}\right)_{n\in[N]}$ by User t for the N servers are said to form an ACSA query if for all $n\in[N]$, we have*

$$Q_n^{(t,\theta_t)}=\left(\mathbf{Q}_{n,1}^{(t,\theta_t)},\mathbf{Q}_{n,2}^{(t,\theta_t)},\cdots,\mathbf{Q}_{n,K_c}^{(t,\theta_t)}\right),\tag{7.39}$$

where for all $i\in[K_c]$

$$\mathbf{Q}_{n,i}^{(t,\theta_t)}=\begin{bmatrix}\mathbf{e}_K(\theta_t)+(\alpha_n-f_{1,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{1,i,s}^{(t)}\\\mathbf{e}_K(\theta_t)+(\alpha_n-f_{2,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{2,i,s}^{(t)}\\\vdots\\\mathbf{e}_K(\theta_t)+(\alpha_n-f_{J,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{J,i,s}^{(t)}\end{bmatrix}.\tag{7.40}$$

Note that for all $t\in\mathbb{N},n\in[N]$, $H\left(Q_n^{(t,\theta_t)}\right)=\mu K_cK$ in q-ary units, because according to Definition 7.1 and Definition 7.2, for all $i\in[K_c]$, $\mathbf{Q}_{n,i}^{(t,\theta_t)}$ is uniquely determined by its first μK

entries (the rest are replicas). Also note that T i.i.d. uniform random noise terms are MDS coded to guarantee the T-privacy.

Remark 7.1. *Indeed, the fact that $\mathbf{Q}_{n,i}^{(t,\theta_t)}$ is uniquely determined by its first μK entries is jointly guaranteed by the cyclic structures of pole assignment and noise assignment, i.e., Definition 7.1 and Definition 7.2. These cyclic structures are important in terms of minimizing the entropy of ACSA queries, so that it does not scale with the number of symbols L.*

Definition 7.5. (ACSA Increment) *For any $t \in \mathbb{N}$, the write-queries $\left(P_n^{(t,\theta_t)}\right)_{n\in[N]}$ by User t for the N servers are said to form an ACSA increment if for all $n \in [N]$, we have*

$$P_n^{(t,\theta_t)} = \left(\mathbf{P}_{n,1}^{(t,\theta_t)}, \mathbf{P}_{n,2}^{(t,\theta_t)}, \cdots, \mathbf{P}_{n,K_c}^{(t,\theta_t)}\right), \tag{7.41}$$

where for all $i \in [K_c]$,

$$\mathbf{P}_{n,i}^{(t,\theta_t)} = \operatorname{diag}\left(\widetilde{\Delta}_{1,i}^{(t)}\mathbf{I}_{KR_w^{(t)}}, \widetilde{\Delta}_{2,i}^{(t)}\mathbf{I}_{KR_w^{(t)}}, \cdots, \widetilde{\Delta}_{\#_w^{(t)},i}^{(t)}\mathbf{I}_{KR_w^{(t)}}\right), \tag{7.42}$$

and for all $\ell \in [\#_w^{(t)}]$,

$$\widetilde{\Delta}_{\ell,i}^{(t)} = \sum_{j\in[(\ell-1)R_w^{(t)}+1:\ell R_w^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} \Delta_{j,i}^{(t)} + \sum_{x\in[X_\Delta]} \alpha_n^{x-1} \dddot{Z}_{\ell,i,x}^{(t)}. \tag{7.43}$$

Note that for all $t \in \mathbb{N}, n \in [N]$, $H\left(P_n^{(t,\theta_t)}\right) = \#_w^{(t)} K_c$ in q-ary units. This is because for all $i \in [K_c]$, $\mathbf{P}_{n,i}^{(t,\theta_t)}$ is uniquely determined by $\left(\widetilde{\Delta}_{\ell,i}^{(t)}\right)_{\ell\in[\#_w^{(t)}]}$. Also note that X_Δ MDS coded i.i.d. uniform noise terms are used to guarantee the X_Δ-security.

Definition 7.6. (ACSA Packer) *For all $t \in \mathbb{N}, n \in [N]$, the ACSA Packer is defined as follows.*

$$\Xi_n^{(t)} = \left(\mathbf{\Xi}_{n,\ell,i}^{(t)}\right)_{\ell\in[\#_r^{(t)}], i\in[K_c]}, \tag{7.44}$$

where for all $\ell \in [\#_r^{(t)}]$, $i \in [K_c]$,

$$\boldsymbol{\Xi}^{(t)}_{n,\ell,i} = \operatorname{diag}\left(\frac{\prod_{i'\in[K_c]\setminus\{i\}}(\alpha_n - f_{1,i'})}{\prod_{i'\in[K_c]\setminus\{i\}}(f_{1,i}-f_{1,i'})}\mathbf{I}_K,\cdots,\frac{\prod_{i'\in[K_c]\setminus\{i\}}(\alpha_n - f_{J,i'})}{\prod_{i'\in[K_c]\setminus\{i\}}(f_{J,i}-f_{J,i'})}\mathbf{I}_K\right) \times \operatorname{diag}(\underbrace{0\mathbf{I}_K,\cdots,0\mathbf{I}_K}_{R_r^{(t)}(\ell-1)\ 0\mathbf{I}_K\text{'s}}, \underbrace{1\mathbf{I}_K,\cdots,1\mathbf{I}_K}_{R_r^{(t)}\ 1\mathbf{I}_K\text{'s}}, \underbrace{0\mathbf{I}_K,\cdots,0\mathbf{I}_K}_{(J-\ell R_r^{(t)})\ 0\mathbf{I}_K\text{'s}}). \tag{7.45}$$

Note that the ACSA packer is a **constant** *since* $|\mathcal{S}_r^{(t)}|$ *is globally known.*

Definition 7.7. (ACSA Unpacker) *For all* $t \in \mathbb{N}, n \in [N]$, *the ACSA Unpacker is defined as follows.*

$$\Upsilon^{(t)}_n = \left(\boldsymbol{\Upsilon}^{(t)}_{n,1}, \boldsymbol{\Upsilon}^{(t)}_{n,2}, \cdots, \boldsymbol{\Upsilon}^{(t)}_{n,K_c}\right), \tag{7.46}$$

where for all $i \in [K_c]$,

$$\boldsymbol{\Upsilon}^{(t)}_{n,i} = \operatorname{diag}\left(\left(\frac{\prod_{j\in\mathcal{F}_1^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_1^{(t)}}(f_{1,i}-f_{j,i})}\right)\mathbf{I}_K,\cdots,\left(\frac{\prod_{j\in\mathcal{F}_J^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_J^{(t)}}(f_{J,i}-f_{j,i})}\right)\mathbf{I}_K\right), \tag{7.47}$$

where for all $j \in [J]$, *we define* $\mathcal{F}_j = \left[\left(\left\lceil j/R_w^{(t)}\right\rceil - 1\right)R_w^{(t)} + 1 : \left\lceil j/R_w^{(t)}\right\rceil R_w^{(t)}\right] \setminus \{j\}$. *Note that the ACSA unpacker is a* **constant** *since* $|\mathcal{S}_w^{(t)}|$ *is globally known.*

Remark 7.2. *As noted at the beginning of the proof, private read and write operations consists of* $K_c\#_r^{(t)}$ *and* $K_c\#_w^{(t)}$ *sub-operations, and for each sub-operation,* $R_r^{(t)}$ *and* $R_w^{(t)}$ *symbols of the desired submodel are retrieved and updated respectively. This is respectively made possible by the constructions of ACSA Packer and Unpacker. For private read, by ACSA Packer, in each sub-operation* $R_r^{(t)}$ *symbols of the desired submodel are "packed" into a Cauchy-Vandermonde structured answer string for best download efficiency, where Cauchy terms carry desired symbols and interference symbols are aligned along Vandermonde terms. Recoverability of the desired symbols is guaranteed by the invertibility of Cauchy-Vandermonde matrices (see, e.g., [37]). On the other hand, ACSA Increment (Definition 7.5) can be viewed* *as a bunch of Cauchy-Vandermonde structured codewords, and in each codeword, a total of $R_w^{(t)}$ symbols are "packed" together for best upload efficiency. To correctly perform private write, each of the codewords is "unpacked" by the ACSA Unpacker to produce $R_w^{(t)}$ codewords that individually carry different increment symbols to preserve the ACSA Storage structure (Definition 7.3).*

Definition 7.8. (ACSA Null-shaper) *For all $t\in\mathbb{N}, n\in[N]$, the ACSA null-shaper is defined as follows.*

$$\Omega^{(t)}=\left(\mathbf{\Omega}_{n,1}^{(t)},\mathbf{\Omega}_{n,2}^{(t)},\cdots,\mathbf{\Omega}_{n,K_c}^{(t)}\right), \tag{7.48}$$

where for all $i\in[K_c]$,

$$\mathbf{\Omega}_{n,i}^{(t)}=\operatorname{diag}\left(\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{1,i}-\alpha_m)}\right)\mathbf{I}_K,\cdots,\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{J,i}-\alpha_m)}\right)\mathbf{I}_K\right). \tag{7.49}$$

Note that for all $n\in\mathcal{S}_w^{(t)}$, we have $\mathbf{\Omega}_n=\mathbf{0}$. Besides, the ACSA null-shaper is a **constant** *since $\mathcal{S}_w^{(t)}$ is globally known.*

Definition 7.9. (ACSA-RW Scheme) *The initial storage at the N servers is the ACSA storage at time 0, i.e., $\left(\mathbf{S}_n^{(0)}\right)_{n\in[N]}$. At time $t, t\in\mathbb{N}$, in the read phase, the user uploads the ACSA query for the N servers and retrieves the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ from the answers returned by the servers $n, n\in\overline{\mathcal{S}}_r^{(t)}$, which are constructed as follows.*

$$A_n^{(t,\theta_t)}=\left(\left(\mathbf{S}_n^{(t-1)}\right)^{\mathsf{T}}\mathbf{\Xi}_{n,\ell,i}^{(t)}\mathbf{Q}_{n,i}^{(t,\theta_t)}\right)_{\ell\in[\#_r^{(t)}],i\in[K_c]}, n\in\overline{\mathcal{S}}_r^{(t)}. \tag{7.50}$$

In the update phase, the user uploads the ACSA increment for the servers $n, n\in\overline{\mathcal{S}}_w^{(t)}$, and each of the servers updates its storage according to the following equation.

$$\mathbf{S}_n^{(t)}=\mathbf{S}_n^{(t-1)}+\sum_{i\in[K_c]}\mathbf{\Omega}_{n,i}^{(t)}\mathbf{\Upsilon}_{n,i}^{(t)}\mathbf{P}_{n,i}^{(t,\theta_t)}\mathbf{Q}_{n,i}^{(t,\theta_t)}, n\in\overline{\mathcal{S}}_w^{(t)}. \tag{7.51}$$

Now let us prove the correctness, privacy and security of the ACSA-RW scheme. To proceed, we need the following lemmas.

Lemma 7.1. *At any time $t, t \in \mathbb{N}$, User t retrieves the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ from the answers returned by the servers $n, n \in \overline{\mathcal{S}}_r^{(t)}$ according to the ACSA-RW scheme while guaranteeing T-privacy.*

Proof. Let us consider the answers returned by the servers $n, n \in \overline{\mathcal{S}}_r^{(t)}$ in (7.50). Note that for all $\ell \in [\#_r^{(t)}]$, $i \in [K_c]$, we have

$$\begin{aligned}
&\left(\mathbf{S}_n^{(t-1)}\right)^{\mathsf{T}} \mathbf{\Xi}_{n,\ell,i}^{(t)} \mathbf{Q}_{n,i}^{(t,\theta_t)} \\
&= \sum_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \left(\sum_{i' \in [K_c]} \frac{1}{\alpha_n - f_{j,i'}} \dot{\mathbf{W}}_{j,i'}^{(t-1)} + \sum_{x \in [X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{j,x}^{(t-1)} \right)^{\mathsf{T}} \\
&\qquad \left(\frac{\prod_{i' \in [K_c] \setminus \{i\}} (\alpha_n - f_{j,i'})}{\prod_{i' \in [K_c] \setminus \{i\}} (f_{j,i} - f_{j,i'})} \right) \left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{j,i}) \sum_{s \in [T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{j,i,s}^{(t)} \right)
\end{aligned} \tag{7.52}$$

$$\begin{aligned}
&= \sum_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \left(\sum_{i' \in [K_c]} \frac{1}{\alpha_n - f_{j,i'}} \dot{\mathbf{W}}_{j,i'}^{(t-1)} + \sum_{x \in [X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{j,x}^{(t-1)} \right)^{\mathsf{T}} && (7.53) \\
&\qquad \left(\left(\frac{\prod_{i' \in [K_c] \setminus \{i\}} (\alpha_n - f_{j,i'})}{\prod_{i' \in [K_c] \setminus \{i\}} (f_{j,i} - f_{j,i'})} \right) \mathbf{e}_K(\theta_t) + \frac{\prod_{i' \in [K_c]} (\alpha_n - f_{j,i})}{\prod_{i' \in [K_c] \setminus \{i\}} (f_{j,i} - f_{j,i'})} \sum_{s \in [T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{j,i,s}^{(t)} \right) && (7.54) \\
&= \sum_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \left(\frac{1}{\alpha_n - f_{j,i}} \dot{\mathbf{W}}_{j,i}^{(t-1)} \mathbf{e}_K(\theta_t) + \sum_{m \in [X+T+K_c-1]} \alpha_n^{m-1} \dot{I}_{j,i,m}^{(t)} \right) && (7.55) \\
&= \sum_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} W_{\theta_t}^{(t-1)}(j,i) + \sum_{m \in [X+T+K_c-1]} \alpha_n^{m-1} \ddot{I}_{r,i,m}^{(t)}. && (7.56)
\end{aligned}$$

For all $j \in [J], m \in [X+T+K_c-1]$, $\dot{I}_{j,i,m}^{(t)}$ are various linear combinations of inner products of $\left(\dot{\mathbf{W}}_{j,i}^{(t-1)}\right)_{i \in [K_c]}, \mathbf{e}_K(\theta_t), \left(\dot{\mathbf{Z}}_{j,x}^{(t-1)}\right)_{x \in [X]}$ and $\left(\ddot{\mathbf{Z}}_{j,i,s}^{(t)}\right)_{i \in [K_c], s \in [T]}$, whose exact forms are irrelevant. Besides, $\ddot{I}_{\ell,i,m}^{(t)} = \sum_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \dot{I}_{j,i,m}^{(t)}, \forall \ell \in [\#_r^{(t)}], i \in [K_c]$. Note that for all

$j \in [J],\ i \in [K_c],\ \prod_{i' \in [K_c]\setminus\{i\}}(f_{j,i} - f_{j,i'})$ is the remainder of the polynomial division (with respect to α_n) $\left(\prod_{i' \in [K_c]\setminus\{i\}}(\alpha_n - f_{j,i'})\right)/(\alpha_n - f_{j,i})$, which is for normalization. The existence of multiplicative inverse of $\prod_{i' \in [K_c]\setminus\{i\}}(f_{j,i} - f_{j,i'})$ is guaranteed by Proposition 7.2, i.e., $\prod_{i' \in [K_c]\setminus\{i\}}(f_{j,i} - f_{j,i'}) \neq 0$. Therefore, due to the fact that the desired symbols are carried by the Cauchy terms (i.e., the first term in (7.56)) and the interference symbols (i.e., the undesired symbols, second term in (7.56)) are aligned along the Vandermonde terms, the desired symbols $\left(W_{\theta_t}^{(t-1)}(j,i)\right)_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]}$ of the submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ are resolvable by inverting the following Cauchy-Vandermonde matrix.

$$\mathbf{C} = \begin{bmatrix} \frac{1}{f_{(\ell-1)R_r^{(t)}+1,i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(1)}} & \cdots & \frac{1}{f_{\ell R_r^{(t)},i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(1)}} & 1 & \cdots & \alpha_{\overline{\mathcal{S}}_r^{(t)}(1)}^{X+T-1} \\ \frac{1}{f_{(\ell-1)R_r^{(t)}+1,i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(2)}} & \cdots & \frac{1}{f_{\ell R_r^{(t)},i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(2)}} & 1 & \cdots & \alpha_{\overline{\mathcal{S}}_r^{(t)}(2)}^{X+T-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_{(\ell-1)R_r^{(t)}+1,i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(|\overline{\mathcal{S}}_r^{(t)}|)}} & \cdots & \frac{1}{f_{\ell R_r^{(t)},i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(|\overline{\mathcal{S}}_r^{(t)}|)}} & 1 & \cdots & \alpha_{\overline{\mathcal{S}}_r^{(t)}(|\overline{\mathcal{S}}_r^{(t)}|)}^{X+T-1} \end{bmatrix}. \tag{7.57}$$

It is remarkable that the non-singularity of the matrix $\mathbf{C}$ follows from Lemma 2.5, as well as the determinant of Cauchy-Vandermonde matrix (see, e.g., [37]) and the fact that according to Proposition 7.1, the constants $\left((f_{j,i})_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]}, (\alpha_n)_{n \in \overline{\mathcal{S}}_r^{(t)}}\right)$ are distinct for all $\ell \in [\#_r^{(t)}]$, regardless of the realizations of $\overline{\mathcal{S}}_r^{(t)}$ and $\overline{\mathcal{S}}_w^{(t)}$. Recall that $\overline{\mathcal{S}}_r^{(t)}(1)$, $\overline{\mathcal{S}}_r^{(t)}(2)$, etc., refer to distinct elements of $\overline{\mathcal{S}}_r^{(t)}$ (arranged in ascending order). Therefore, the user is able to reconstruct the desired submodel due to the fact that $\mathbf{W}_{\theta_t}^{(t-1)} = \left(\left(W_{\theta_t}^{(t-1)}(j,i)\right)_{j \in [(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]}\right)_{\ell \in [\#_r^{(t)}], i \in [K_c]}$. To see why the T-privacy holds, we note that by the construction of the query $\left(Q_n^{(t,\theta_t)}\right)_{n \in [N]}$, the vector $\mathbf{e}_K(\theta_t)$, which carries the information of desired index θ_t, is protected by the MDS(N,T) coded uniform i.i.d. random noise vectors. Thus the queries for the N servers form a secret sharing of threshold T, and are independent of the queries and the increments of all prior users $\tau, \tau \in [t-1]$. Thus T-privacy is guaranteed. To calculate the download cost, we note that a total of $L = JK_c$ symbols of the desired submodel are retrieved out of the $K_c \#_r^{(t)} |\overline{\mathcal{S}}_r^{(t)}|$ downloaded symbols.

Therefore we have $D_t = \left(K_c \#_r^{(t)} |\overline{\mathcal{S}}_r^{(t)}|\right)/(JK_c) = \left(\#_r^{(t)}\left(N - |\mathcal{S}_r^{(t)}|\right)\right)/J = \left(N - |\mathcal{S}_r^{(t)}|\right)/R_r^{(t)} = \left(N - |\mathcal{S}_r^{(t)}|\right)/\left(S_r^{\text{thresh}} - |\mathcal{S}_r^{(t)}|\right)$. This completes the proof of Lemma 7.1. □

Lemma 7.2. *At any time $t, t \in \mathbb{N}$, User t correctly updates the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ and achieves ACSA storage by uploading the ACSA increments to the servers $n, n \in \overline{\mathcal{S}}_w^{(t)}$ and exploiting the update equation* (7.51) *according to the ACSA-RW scheme while guaranteeing T-privacy and X_Δ-security.*

Proof. Let us first inspect the second term on the RHS of (7.51). Note that for any $t \in \mathbb{N}$, for all $n \in [N], i \in [K_c]$, we can write

$$\begin{aligned}
&\mathbf{\Omega}_{n,i}^{(t)} \mathbf{\Upsilon}_{n,i}^{(t)} \mathbf{P}_{n,i}^{(t,\theta_t)} \mathbf{Q}_{n,i}^{(t,\theta_t)} \\
&= \left[\left(\mathbf{\Gamma}_{n,1,i}^{(t)}\right)^{\mathsf{T}}, \left(\mathbf{\Gamma}_{n,2,i}^{(t)}\right)^{\mathsf{T}}, \cdots, \left(\mathbf{\Gamma}_{n,\#_w^{(t)},i}^{(t)}\right)^{\mathsf{T}}\right]^{\mathsf{T}},
\end{aligned} \tag{7.58}$$

where for all $\ell \in [\#_w^{(t)}]$, we define $\phi_\ell = (\ell - 1) R_w^{(t)}$, and

$$\begin{aligned}
&\mathbf{\Gamma}_{n,\ell,i}^{(t)} \\
&= \begin{bmatrix}
\dfrac{\left(\frac{\prod_{j \in \mathcal{F}_{\phi_\ell+1}^{(t)}} (\alpha_n - f_{j,i})}{\prod_{j \in \mathcal{F}_{\phi_\ell+1}^{(t)}} (f_{\phi_\ell+1,i} - f_{j,i})}\right) \left(\sum_{j \in [\phi_\ell+1:\phi_\ell+R_w^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} \Delta_{j,i}^{(t)} + \sum_{x \in [X_\Delta]} \alpha_n^{x-1} \dddot{Z}_{\ell,i,x}^{(t)}\right)}{\left(\frac{\prod_{m \in \mathcal{S}_w^{(t)}} (\alpha_n - \alpha_m)}{\prod_{m \in \mathcal{S}_w^{(t)}} (f_{\phi_\ell+1,i} - \alpha_m)}\right) \left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+1,i}) \sum_{s \in [T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{\phi_\ell+1,i,s}^{(t)}\right)} \\
\dfrac{\left(\frac{\prod_{j \in \mathcal{F}_{\phi_\ell+2}^{(t)}} (\alpha_n - f_{j,i})}{\prod_{j \in \mathcal{F}_{\phi_\ell+2}^{(t)}} (f_{\phi_\ell+2,i} - f_{j,i})}\right) \left(\sum_{j \in [\phi_\ell+1:\phi_\ell+R_w^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} \Delta_{j,i}^{(t)} + \sum_{x \in [X_\Delta]} \alpha_n^{x-1} \dddot{Z}_{\ell,i,x}^{(t)}\right)}{\left(\frac{\prod_{m \in \mathcal{S}_w^{(t)}} (\alpha_n - \alpha_m)}{\prod_{m \in \mathcal{S}_w^{(t)}} (f_{\phi_\ell+2,i} - \alpha_m)}\right) \left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+2,i}) \sum_{s \in [T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{\phi_\ell+2,i,s}^{(t)}\right)} \\
\hline
\vdots \\
\hline
\dfrac{\left(\frac{\prod_{j \in \mathcal{F}_{\phi_\ell+R_w^{(t)}}^{(t)}} (\alpha_n - f_{j,i})}{\prod_{j \in \mathcal{F}_{\phi_\ell+R_w^{(t)}}^{(t)}} (f_{\phi_\ell+R_w^{(t)},i} - f_{j,i})}\right) \left(\sum_{j \in [\phi_\ell+1:\phi_\ell+R_w^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} \Delta_{j,i}^{(t)} + \sum_{x \in [X_\Delta]} \alpha_n^{x-1} \dddot{Z}_{\ell,i,x}^{(t)}\right)}{\left(\frac{\prod_{m \in \mathcal{S}_w^{(t)}} (\alpha_n - \alpha_m)}{\prod_{m \in \mathcal{S}_w^{(t)}} (f_{\phi_\ell+R_w^{(t)},i} - \alpha_m)}\right) \left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+R_w^{(t)},i}) \sum_{s \in [T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{\phi_\ell+R_w^{(t)},i,s}^{(t)}\right)}
\end{bmatrix}
\end{aligned} \tag{7.59}$$

$$
= \left[\begin{array}{c}
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+1,i}-\alpha_m)}\right)\left(\frac{1}{\alpha_n-f_{\phi_\ell+1,i}}\Delta^{(t)}_{\phi_\ell+1,i}+\sum_{m\in[X_\Delta+R_w^{(t)}-1]}\alpha_n^{m-1}\dddot{I}^{(t)}_{\phi_\ell+1,i,m}\right)\\
\left(\mathbf{e}_K(\theta_t)+(\alpha_n-f_{\phi_\ell+1,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}^{(t)}_{\phi_\ell+1,i,s}\right)\\
\hline
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+2,i}-\alpha_m)}\right)\left(\frac{1}{\alpha_n-f_{\phi_\ell+2,i}}\Delta^{(t)}_{\phi_\ell+2,i}+\sum_{m\in[X_\Delta+R_w^{(t)}-1]}\alpha_n^{m-1}\dddot{I}^{(t)}_{\phi_\ell+2,i,m}\right)\\
\left(\mathbf{e}_K(\theta_t)+(\alpha_n-f_{\phi_\ell+2,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}^{(t)}_{\phi_\ell+2,i,s}\right)\\
\hline
\vdots\\
\hline
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+R_w^{(t)},i}-\alpha_m)}\right)\left(\frac{1}{\alpha_n-f_{\phi_\ell+R_w^{(t)},i}}\Delta^{(t)}_{\phi_\ell+R_w^{(t)},i}+\sum_{m\in[X_\Delta+R_w^{(t)}-1]}\alpha_n^{m-1}\dddot{I}^{(t)}_{\phi_\ell+R_w^{(t)},i,m}\right)\\
\left(\mathbf{e}_K(\theta_t)+(\alpha_n-f_{\phi_\ell+R_w^{(t)},m})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}^{(t)}_{\phi_\ell+R_w^{(t)},i,s}\right)
\end{array}\right]
\tag{7.60}
$$

$$
= \left[\begin{array}{c}
\left(\frac{1}{\alpha_n-f_{\phi_\ell+1,i}}\Delta^{(t)}_{\phi_\ell+1,i}+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}\alpha_n^{m-1}\ddddot{I}^{(t)}_{\phi_\ell+1,i,m}\right)\\
\left(\mathbf{e}_K(\theta_t)+(\alpha_n-f_{\phi_\ell+1,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}^{(t)}_{\phi_\ell+1,i,s}\right)\\
\hline
\left(\frac{1}{\alpha_n-f_{\phi_\ell+2,i}}\Delta^{(t)}_{\phi_\ell+2,i}+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}\alpha_n^{m-1}\ddddot{I}^{(t)}_{\phi_\ell+2,i,m}\right)\\
\left(\mathbf{e}_K(\theta_t)+(\alpha_n-f_{\phi_\ell+2,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}^{(t)}_{\phi_\ell+2,i,s}\right)\\
\hline
\vdots\\
\hline
\left(\frac{1}{\alpha_n-f_{\phi_\ell+R_w^{(t)},i}}\Delta^{(t)}_{\phi_\ell+R_w^{(t)},i}+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}\alpha_n^{m-1}\ddddot{I}^{(t)}_{\phi_\ell+R_w^{(t)},i,m}\right)\\
\left(\mathbf{e}_K(\theta_t)+(\alpha_n-f_{\phi_\ell+R_w^{(t)},i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}^{(t)}_{\phi_\ell+R_w^{(t)},i,s}\right)
\end{array}\right]
\tag{7.61}
$$

$$
= \left[\begin{array}{c}
\left(\frac{1}{\alpha_n-f_{\phi_\ell+1,i}}\Delta^{(t)}_{\phi_\ell+1,i}\mathbf{e}_K(\theta_t)+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}^{(t)}_{\phi_\ell+1,i,m}\right)\\
\hline
\left(\frac{1}{\alpha_n-f_{\phi_\ell+2,i}}\Delta^{(t)}_{\phi_\ell+2,i}\mathbf{e}_K(\theta_t)+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}^{(t)}_{\phi_\ell+2,i,m}\right)\\
\hline
\vdots\\
\hline
\left(\frac{1}{\alpha_n-f_{\phi_\ell+R_w^{(t)},i}}\Delta^{(t)}_{\phi_\ell+R_w^{(t)},i}\mathbf{e}_K(\theta_t)+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}^{(t)}_{\phi_\ell+R_w^{(t)},i,m}\right)
\end{array}\right],
\tag{7.62}
$$

where for all $v\in[R_w^{(t)}]$, $(\dddot{I}_{\phi_\ell+v,m})_{m\in[X_\Delta+R_w^{(t)}-1]}$, $(\ddddot{I}_{\phi_\ell+v,m})_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}$ and $\left(\dot{\mathbf{I}}_{\phi_\ell+v,m}\right)_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}$ are various interference symbols, whose exact forms are not important. Note that in (7.60), we multiply the first two terms in each row of (7.59). It can be justified from the fact that the constants $(f_{j,i})_{j\in[\phi_\ell+1:\phi_\ell+R_w^{(t)}]}$ are distinct according to Proposition 7.1. Besides, for all $v\in[R_w^{(t)}]$, according to the definition of $\mathcal{F}^{(t)}_{\phi_\ell+v}$, we can equivalently write $\mathcal{F}^{(t)}_{\phi_\ell+v}=[\phi_\ell+1:\phi_\ell+R_w^{(t)}]\setminus\{\phi_\ell+v\}$, thus $\left|\mathcal{F}^{(t)}_{\phi_\ell+v}\right|=R_w^{(t)}-1$ and $(\phi_\ell+v)\notin\mathcal{F}_{\phi_\ell+v}$. Note that the denominator in the first term in each row of (7.59) is for normalization, which is the remainder of the polynomial division (with respect to α_n) $\left(\prod_{j\in\mathcal{F}^{(t)}_{\phi_\ell+v}}(\alpha_n-f_{j,i})\right)/(\alpha_n-f_{\phi_\ell+v,i})$. In (7.61), we multiply the first two terms in each row of (7.60), and it can be justified by noting that the denominator in the first term in each row of (7.60) is the remainder of the polynomial division (with respect to α_n) $\left(\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)\right)/(\alpha_n-f_{\phi_\ell+v,i})$. And finally in (7.62), we multiply the two terms in each row of (7.61). Therefore, for all $n\in[N],i\in[K_c]$, the term $\mathbf{\Omega}^{(t)}_{n,i}\mathbf{\Upsilon}^{(t)}_{n,i}\mathbf{P}^{(t,\theta_t)}_{n,i}\mathbf{Q}^{(t,\theta_t)}_{n,i}$ can be written as follows.

$$\begin{aligned}&\mathbf{\Omega}^{(t)}_{n,i}\mathbf{\Upsilon}^{(t)}_{n,i}\mathbf{P}^{(t,\theta_t)}_{n,i}\mathbf{Q}^{(t,\theta_t)}_{n,i}\\&=\left[\begin{array}{c}\frac{1}{\alpha_n-f_{1,i}}\Delta^{(t)}_{1,i}\mathbf{e}_K(\theta_t)+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}^{(t)}_{1,i,m}\\\hline\frac{1}{\alpha_n-f_{2,i}}\Delta^{(t)}_{2,i}\mathbf{e}_K(\theta_t)+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}^{(t)}_{2,i,m}\\\hline\vdots\\\hline\frac{1}{\alpha_n-f_{J,i}}\Delta^{(t)}_{J,i}\mathbf{e}_K(\theta_t)+\sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}^{(t)}_{J,i,m}\end{array}\right].\end{aligned}\tag{7.63}$$

Note that $X=X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1$, the update equation (7.51) is thus correct because

$$\begin{aligned}&\mathbf{S}_n^{(t-1)}+\sum_{i\in[K_c]}\mathbf{\Omega}^{(t)}_{n,i}\mathbf{\Upsilon}^{(t)}_{n,i}\mathbf{P}^{(t,\theta_t)}_{n,i}\mathbf{Q}^{(t,\theta_t)}_{n,i}\\&=\left[\begin{array}{c}\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{1,i}}\dot{\mathbf{W}}^{(t-1)}_{1,i}+\sum_{x\in[X]}\alpha_n^{x-1}\dot{\mathbf{Z}}^{(t-1)}_{1,x}\\\hline\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{2,i}}\dot{\mathbf{W}}^{(t-1)}_{2,i}+\sum_{x\in[X]}\alpha_n^{x-1}\dot{\mathbf{Z}}^{(t-1)}_{2,x}\\\hline\vdots\\\hline\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{J,i}}\dot{\mathbf{W}}^{(t-1)}_{J,i}+\sum_{x\in[X]}\alpha_n^{x-1}\dot{\mathbf{Z}}^{(t-1)}_{J,x}\end{array}\right]\\&\quad+\left[\begin{array}{c}\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{1,i}}\Delta^{(t)}_{1,i}\mathbf{e}_K(\theta_t)+\sum_{x\in[X]}\alpha_n^{x-1}\ddot{\mathbf{I}}^{(t)}_{1,x}\\\hline\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{2,i}}\Delta^{(t)}_{2,i}\mathbf{e}_K(\theta_t)+\sum_{x\in[X]}\alpha_n^{x-1}\ddot{\mathbf{I}}^{(t)}_{2,x}\\\hline\vdots\\\hline\sum_{i\in[K_c]}\frac{1}{\alpha_n-f_{J,i}}\Delta^{(t)}_{J,i}\mathbf{e}_K(\theta_t)+\sum_{x\in[X]}\alpha_n^{x-1}\ddot{\mathbf{I}}^{(t)}_{J,x}\end{array}\right]\end{aligned}\tag{7.64}$$

$$=\mathbf{S}_n^{(t)}, \tag{7.65}$$

where for all $t\in\mathbb{N},j\in[J],x\in[X]$, $\dot{\mathbf{Z}}_{j,x}^{(t)}=\dot{\mathbf{Z}}_{j,x}^{(t-1)}+\ddot{\mathbf{I}}_{j,x}^{(t)}$, and $\ddot{\mathbf{I}}_{j,x}^{(t)}=\sum_{i\in[K_c]}\dot{\mathbf{I}}_{j,i,x}^{(t)}$. Note that for all $n\in\mathcal{S}_w^{(t)}$, $i\in[K_c]$, we have $\mathbf{\Omega}_{n,i}^{(t)}=\mathbf{0}$. Therefore, for all $n\in\mathcal{S}_w^{(t)}$, it holds that $\mathbf{S}_n^{(t)}=\mathbf{S}_n^{(t-1)}$. In other words, the update equation (7.51) correctly updates the desired submodel and achieves the ACSA storage at time $t+1$ by updating the storage of server $n,n\in\overline{\mathcal{S}}_w^{(t)}$. The proof of T-privacy follows from that in Lemma 7.1, so we do not repeat it here. The proof of X_Δ-security follows from the fact that by the definition of ACSA increment $\left(P_n^{(t,\theta_t)}\right)_{n\in[N]}$, the symbols of the increment $\mathbf{\Delta}_t$ are protected by the MDS(N,X_Δ) coded uniform i.i.d. random noise symbols. Thus the ACSA increment for the N servers form a secret sharing of threshold X_Δ, and it is independent of the write-queries by the users $\tau,\tau\in[t-1]$ and the read-queries by the users $\tau,\tau\in[t]$. Finally let us calculate the upload cost. The upload cost consists of two parts, i.e., the upload cost of the ACSA query and the upload cost of the ACSA increment. Note that to upload the ACSA query, a total of $\left|[N]\setminus\mathcal{S}_r^{(t)}\setminus\mathcal{S}_w^{(t)}\right|\mu KK_c$ q-ary symbols must be uploaded. On the other hand, to upload the ACSA increment, we need to upload a total of $\left(N-|\mathcal{S}_w^{(t)}|\right)\#_w^{(t)}K_c$ q-ary symbols. Therefore, in the limit as $L/K\to\infty$, the normalized upload cost is

$$U_t=\frac{K_c\left(N-|\mathcal{S}_w^{(t)}|\right)\#_w^{(t)}+\left|[N]\setminus\mathcal{S}_r^{(t)}\setminus\mathcal{S}_w^{(t)}\right|\mu KK_c}{L} \tag{7.66}$$

$$\overset{L/K\to\infty}{=}\frac{N-|\mathcal{S}_w^{(t)}|}{R_w^{(t)}} \tag{7.67}$$

$$=\frac{N-|\mathcal{S}_w^{(t)}|}{S_w^{\text{thresh}}-|\mathcal{S}_w^{(t)}|}. \tag{7.68}$$

This completes the proof of Lemma 7.2. □

The ACSA-RW scheme satisfies the correctness, T-privacy, and X_Δ-security constraints for each update $t,t\in\mathbb{N}$ because of Lemma 7.1 and Lemma 7.2, which hold for all $t,t\in\mathbb{N}$. Now let us see why the X-security constraint is satisfied. By the definition of the ACSA storage (i.e., Definition 7.3) at any time $t,t\in\mathbb{N}$, the symbols of the K submodels are protected by the MDS(N,X) coded i.i.d. uniform random noise symbols. In other words, it forms a secret sharing of threshold X, thus X-security is guaranteed.

Remark 7.3. (Byzantine Tolerance) *It is remarkable that the answers returned by the servers in the private read phase can be viewed as codewords of an MDS code (generated by the Cauchy-Vandermonde matrix, see, e.g., Chapter 4 and [54]). Therefore, with additional* $2B$ *redundant answers from the servers, we can correct up to* B *erroneous answers.*

Remark 7.4. (Symmetric Security) *If common randomness is allowed among the servers, so-called* **symmetric security** *can be achieved[20], i.e., the user will learn nothing about the global model beyond the desired submodel. Note that this does not affect the communication cost of the ACSA-RW scheme.*

Remark 7.5. (External Adversaries versus Internal Adversaries) *Recall that the* X*-security constraint only requires protection against an external adversary who can access the current storage but not the past history at any* X*-servers. However, a closer look at the ACSA-RW scheme reveals that if the number of compromised servers is no more than* $\min(X_\Delta,T)$*, then even an* internal *adversary, i.e., an adversary who has access to the entire history of all previous stored values and queries seen by the compromised servers, can still learn nothing about the stored submodels.*

Remark 7.6. (Access Complexity) *We define the access complexity as the number of elements over the finite field* $\mathbb{F}_q$ *that must be accessed/updated during the private read and private write phases. We note that at any time* $t,t\in\mathbb{N}$*, the access complexity of each of the responsive servers in the private read and write phases is at most* KL/K_c*. Hence with greater* K_c*, it is possible to reduce the access complexity.*

Remark 7.7. (Encoding and Decoding Complexity) *Let us consider the complexity of the encoding and decoding algorithms of our construction. It is worth noting that the computations for producing the ACSA storage, ACSA query and ACSA increment can be* *regarded as multiplications of (scaled) Cauchy-Vandermonde matrices with various vectors. The computation for recovering the desired submodel by the user from the answers of the N servers can be viewed as solving linear systems defined by Cauchy-Vandermonde matrices. Cauchy-Vandermonde matrices are an important class of structured matrices, for which* "superfast" *algorithms have been studied extensively [85, 35]. Therefore, by these superfast algorithms, the complexity of producing the ACSA storage, ACSA query and ACSA increment is at most $\widetilde{\mathcal{O}}((LKN\log^2 N)/K_c)$,$\widetilde{\mathcal{O}}(\mu KNK_c\log^2 N)$ and $\widetilde{\mathcal{O}}(U_tL\log^2 N)$, respectively. On the other hand, the complexity of decoding the desired submodel from the answers of the servers is at most $\widetilde{\mathcal{O}}(D_tL\log^2 N)$. It is obvious that the encoding/decoding algorithms have a complexity that is almost linear in their output/input sizes.*

7.4.1 Example

Let us consider an illustrative example to make the construction of the ACSA-RW scheme and the proof of Theorem 7.1 more accessible. In particular, for this example, let us set $N=8,X=4,T=1,X_\Delta=1$ and $K_c=1$ (note that $L=J$ in this case). We follow the notations used in the proof of Theorem 7.1, and since $K_c=1$, we are able to omit an index in some of the subscripts, which corresponds to $i\in[K_c]$. For example, for all $l\in[L]$, $f_{l,1}$ is abbreviated as f_l, etc. We have $S_r^{\text{thresh}}=3,S_w^{\text{thresh}}=3$. Let ξ be a positive integer, and we set $L=\xi\cdot\text{lcm}\left([S_r^{\text{thresh}}]\cup[S_w^{\text{thresh}}]\right)=6\xi$, i.e., each of the K submodels consists of $L=6\xi$ symbols from a finite field $\mathbb{F}_q,q\geq N+\mu=11$. Let $\alpha_1,\alpha_2,\cdots,\alpha_8,\widetilde{f}_1,\widetilde{f}_2,\widetilde{f}_3$ be a total of 11 distinct elements from the finite field $\mathbb{F}_q$. For $(f_l)_{l\in[L]}$, let us define

$$(f_1,f_2,\cdots,f_L)=(\widetilde{f}_1,\widetilde{f}_2,\widetilde{f}_3,\widetilde{f}_1,\widetilde{f}_2,\widetilde{f}_3,\cdots,\widetilde{f}_1,\widetilde{f}_2,\widetilde{f}_3). \tag{7.69}$$

Let $\left(\dot{\mathbf{Z}}_{l,x}^{(0)}\right)_{l\in[L],x\in[4]}$ be uniformly i.i.d. column vectors from $\mathbb{F}_q^K$. The initial ACSA storage at the $N=8$ servers is defined as follows.

$$\mathbf{S}_n^{(0)}=\begin{bmatrix}\frac{1}{\alpha_n-f_1}\dot{\mathbf{W}}_1^{(0)}+\sum_{x\in[4]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{1,x}^{(0)}\\ \frac{1}{\alpha_n-f_2}\dot{\mathbf{W}}_2^{(0)}+\sum_{x\in[4]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{2,x}^{(0)}\\ \vdots\\ \frac{1}{\alpha_n-f_L}\dot{\mathbf{W}}_L^{(0)}+\sum_{x\in[4]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{L,x}^{(0)}\end{bmatrix},\forall n\in[8]. \tag{7.70}$$

Now let us assume that User 1 experiences $|\mathcal{S}_r^{(1)}|=1,|\mathcal{S}_w^{(1)}|=2$, i.e., there is 1 dropout server in the read phase and 2 dropout servers in the write phase. Note that the two sets can be arbitrarily realized. We have $R_r^{(1)}=2,R_w^{(1)}=1,\#_r^{(1)}=3\xi,\#_w^{(1)}=6\xi$. The ACSA query sent by User 1 to the n^{th} server, $n\in[N]$ is defined as follows.

$$\mathbf{Q}_n^{(1,\theta_1)}=\begin{bmatrix}\mathbf{e}_K(\theta_1)+(\alpha_n-f_1)\ddot{\mathbf{Z}}_{1,1}^{(1)}\\ \mathbf{e}_K(\theta_1)+(\alpha_n-f_2)\ddot{\mathbf{Z}}_{2,1}^{(1)}\\ \vdots\\ \mathbf{e}_K(\theta_1)+(\alpha_n-f_L)\ddot{\mathbf{Z}}_{L,1}^{(1)}\end{bmatrix}. \tag{7.71}$$

where

$$\left(\ddot{\mathbf{Z}}_{1,1}^{(1)},\ddot{\mathbf{Z}}_{2,1}^{(1)},\cdots,\ddot{\mathbf{Z}}_{L,1}^{(1)}\right)=\left(\widetilde{\mathbf{Z}}_{1,1}^{(1)},\widetilde{\mathbf{Z}}_{2,1}^{(1)},\widetilde{\mathbf{Z}}_{3,1}^{(1)},\widetilde{\mathbf{Z}}_{1,1}^{(1)},\widetilde{\mathbf{Z}}_{2,1}^{(1)},\widetilde{\mathbf{Z}}_{3,1}^{(1)},\cdots,\widetilde{\mathbf{Z}}_{1,1}^{(1)},\widetilde{\mathbf{Z}}_{2,1}^{(1)},\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right), \tag{7.72}$$

and $\left(\widetilde{\mathbf{Z}}_{l,1}^{(1)}\right)_{l\in[3]}$ are i.i.d. uniform column vectors from $\mathbb{F}_q^K$. Evidently, for all $n\in[8]$, $\mathbf{Q}_n^{(1,\theta_1)}$ is uniquely determined by its first $3K$ rows. Upon receiving the queries, servers $n,n\in\overline{\mathcal{S}}_r^{(1)}$ respond to User 1 with answers $A_n^{(1,\theta_1)}$ constructed as follows.

$$A_n^{(1,\theta_1)}=\left(\mathbf{A}_{n,\ell}^{(1,\theta_1)}\right)_{\ell\in[3\xi]},n\in\overline{\mathcal{S}}_r^{(1)}, \tag{7.73}$$

where for all $n\in\overline{\mathcal{S}}_r^{(1)},\ell\in[3\xi]$,

$$\begin{aligned}
\mathbf{A}_{n,\ell}^{(1,\theta_1)} &= \left(\mathbf{S}_n^{(0)}\right)^{\mathsf{T}}\mathbf{\Xi}_{n,\ell}^{(1)}\mathbf{Q}_n^{(1,\theta_1)} && (7.74)\\
&= \sum_{l\in[2\ell-1:2\ell]}\left(\frac{1}{\alpha_n-f_l}\dot{\mathbf{W}}_l^{(0)}+\sum_{x\in[4]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{l,x}^{(0)}\right)^{\mathsf{T}}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_l)\ddot{\mathbf{Z}}_{l,1}^{(1)}\right) && (7.75)\\
&= \frac{W_{\theta_1}^{(0)}(2\ell-1)}{\alpha_n-f_{2\ell-1}}+\frac{W_{\theta_1}^{(0)}(2\ell)}{\alpha_n-f_{2\ell}}+\sum_{i\in[5]}\alpha_n^{i-1}\ddot{I}_{\ell,i}^{(1)}, && (7.76)
\end{aligned}$$

where $\left(\ddot{I}_{\ell,i}\right)_{\ell\in[3\xi],i\in[5]}$ are various interference symbols. Thus for all $\ell\in[3\xi]$, User 1 recovers the desired symbols $W_{\theta_1}^{(0)}(2\ell-1)$ and $W_{\theta_1}^{(0)}(2\ell)$ by inverting the following matrix.

$$\mathbf{C}=\begin{bmatrix}
\frac{1}{f_{2\ell-1}-\alpha_{\overline{\mathcal{S}}_r^{(1)}(1)}} & \frac{1}{f_{2\ell}-\alpha_{\overline{\mathcal{S}}_r^{(1)}(1)}} & 1 & \cdots & \alpha^4_{\overline{\mathcal{S}}_r^{(1)}(1)}\\
\frac{1}{f_{2\ell-1}-\alpha_{\overline{\mathcal{S}}_r^{(1)}(2)}} & \frac{1}{f_{2\ell}-\alpha_{\overline{\mathcal{S}}_r^{(1)}(2)}} & 1 & \cdots & \alpha^4_{\overline{\mathcal{S}}_r^{(1)}(2)}\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
\frac{1}{f_{2\ell-1}-\alpha_{\overline{\mathcal{S}}_r^{(1)}(7)}} & \frac{1}{f_{2\ell}-\alpha_{\overline{\mathcal{S}}_r^{(1)}(7)}} & 1 & \cdots & \alpha^4_{\overline{\mathcal{S}}_r^{(1)}(7)}
\end{bmatrix}. \qquad (7.77)$$

Again, its invertibility is guaranteed by Lemma 2.5, as well as the determinant of Cauchy-Vandermonde matrices [37] and the fact that the constants $(f_{2\ell-1},f_{2\ell},\alpha_1,\alpha_2,\cdots,\alpha_8)$ are distinct for all $\ell\in[3\xi]$. Recall that $\overline{\mathcal{S}}_r^{(1)}(1)$, $\overline{\mathcal{S}}_r^{(1)}(2)$, etc., refer to distinct elements of $\overline{\mathcal{S}}_r^{(1)}$, i.e., available servers during the first read phase, arranged in ascending order. Therefore, the user is able to reconstruct the desired submodel $\mathbf{W}_{\theta_1}^{(0)}$.

To update the desired submodel with the increment $\mathbf{\Delta}_1$, User 1 constructs the ACSA increment $\mathbf{P}_n^{(1,\theta_1)}$ as follows.

$$\mathbf{P}_n^{(1,\theta_1)}=\operatorname{diag}\left(\widetilde{\Delta}_1^{(1)}\mathbf{I}_K,\widetilde{\Delta}_2^{(1)}\mathbf{I}_K,\cdots,\widetilde{\Delta}_L^{(1)}\mathbf{I}_K\right), \qquad (7.78)$$

where for all $\ell \in [L]$,

$$\widetilde{\Delta}_{\ell}^{(1)} = \frac{1}{\alpha_n - f_\ell}\Delta_{\ell}^{(1)} + \dddot{Z}_{\ell,1}^{(1)}. \tag{7.79}$$

Note that the ACSA unpacker at time $t=1$ is $\boldsymbol{\Upsilon}_n^{(1)} = \mathbf{I}_{KL}$ for all $n \in [N]$, therefore,

$$\begin{aligned}
&\boldsymbol{\Omega}_n^{(1)}\boldsymbol{\Upsilon}_n^{(1)}\mathbf{P}_n^{(1,\theta_1)}\mathbf{Q}_n^{(1,\theta_1)} \\
&= \left[\begin{array}{c}
\left(\frac{\prod_{i\in\mathcal{S}_w^{(1)}}(\alpha_n-\alpha_i)}{\prod_{i\in\mathcal{S}_w^{(1)}}(f_1-\alpha_i)}\right)\left(\frac{1}{\alpha_n-f_1}\Delta_1^{(1)} + \dddot{Z}_{1,1}^{(1)}\right)\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_1)\ddot{\mathbf{Z}}_{1,s}^{(1)}\right) \\ \hline
\left(\frac{\prod_{i\in\mathcal{S}_w^{(1)}}(\alpha_n-\alpha_i)}{\prod_{i\in\mathcal{S}_w^{(1)}}(f_2-\alpha_i)}\right)\left(\frac{1}{\alpha_n-f_2}\Delta_2^{(1)} + \dddot{Z}_{2,1}^{(1)}\right)\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_2)\ddot{\mathbf{Z}}_{2,s}^{(1)}\right) \\ \hline
\vdots \\ \hline
\left(\frac{\prod_{i\in\mathcal{S}_w^{(1)}}(\alpha_n-\alpha_i)}{\prod_{i\in\mathcal{S}_w^{(1)}}(f_L-\alpha_i)}\right)\left(\frac{1}{\alpha_n-f_L}\Delta_L^{(1)} + \dddot{Z}_{L,1}^{(1)}\right)\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_L)\ddot{\mathbf{Z}}_{L,s}^{(1)}\right)
\end{array}\right] && (7.80) \\
&= \left[\begin{array}{c}
\frac{1}{\alpha_n-f_1}\Delta_1^{(1)}\mathbf{e}_K(\theta_t) + \sum_{i\in[4]}\alpha_n^{i-1}\dot{\mathbf{I}}_{1,i}^{(1)} \\ \hline
\frac{1}{\alpha_n-f_2}\Delta_2^{(1)}\mathbf{e}_K(\theta_t) + \sum_{i\in[4]}\alpha_n^{i-1}\dot{\mathbf{I}}_{2,i}^{(1)} \\ \hline
\vdots \\ \hline
\frac{1}{\alpha_n-f_L}\Delta_L^{(1)}\mathbf{e}_K(\theta_t) + \sum_{i\in[4]}\alpha_n^{i-1}\dot{\mathbf{I}}_{L,i}^{(1)}
\end{array}\right]. && (7.81)
\end{aligned}$$

Thus by the update equation (7.51), the ACSA scheme updates the storage at time 0, i.e., it holds that $\mathbf{S}_n^{(0)} + \boldsymbol{\Omega}_n^{(1)}\boldsymbol{\Upsilon}_n^{(1)}\mathbf{P}_n^{(1,\theta_1)}\mathbf{Q}_1^{(1,\theta_1)} = \mathbf{S}_n^{(1)}$. Besides, it is guaranteed that $\mathbf{S}_n^{(1)} = \mathbf{S}_n^{(0)}$ for all $n \in \mathcal{S}_w^{(1)}$. This is the end of the full cycle of ACSA-RW at time 1. We note that in the limit as $L/K \to \infty$, we have $D_t = 7/2 = 3.5$ and $U_t = 6$.

Now let us assume that at time $t=2$, i.e., for the second cycle with a second user, we have $|\mathcal{S}_r^{(1)}| = 2, |\mathcal{S}_w^{(1)}| = 1$. Therefore, $R_r^{(2)} = 1, R_w^{(2)} = 2, \#_r^{(2)} = 6\xi, \#_w^{(2)} = 3\xi$. Upon receiving the queries, the servers $n, n \in \overline{\mathcal{S}}_r^{(2)}$ respond to the user with the answers as follows.

$$A_n^{(2,\theta_2)} = \left(\mathbf{A}_{n,\ell}^{(2,\theta_2)}\right)_{\ell\in[L]}, n \in \overline{\mathcal{S}}_r^{(2)}, \tag{7.82}$$

where for all $\ell \in [L]$, we have

$$\mathbf{A}_{n,\ell}^{(2,\theta_2)} = \frac{W_{\theta_2}^{(1)}(\ell)}{\alpha_n - f_\ell} + \sum_{i\in[5]} \alpha_n^{i-1} \ddot{I}_{\ell,i}^{(2)}. \tag{7.83}$$

Thus, User 2 recovers the desired submodel by inverting the matrix $\mathbf{C}$ as defined in (7.57). To update the submodel with the increment $\boldsymbol{\Delta}_2$, User 2 constructs the ACSA increment $\mathbf{P}_n^{(2,\theta_2)}$ as follows.

$$\mathbf{P}_n^{(2,\theta_2)} = \operatorname{diag}\left(\widetilde{\Delta}_1^{(2)}\mathbf{I}_{2K}, \widetilde{\Delta}_2^{(2)}\mathbf{I}_{2K}, \cdots, \widetilde{\Delta}_{3\xi}^{(2)}\mathbf{I}_{2K}\right), \tag{7.84}$$

where for all $\ell \in [3\xi]$,

$$\widetilde{\Delta}_\ell^{(2)} = \frac{1}{\alpha_n - f_{2\ell-1}}\Delta_{2\ell-1}^{(2)} + \frac{1}{\alpha_n - f_{2\ell}}\Delta_{2\ell}^{(2)} + \dddot{Z}_{\ell,1}^{(2)}. \tag{7.85}$$

Recall that the ACSA unpacker at time $t = 2$ is defined as follows.

$$\begin{aligned}
\boldsymbol{\Upsilon}_n^{(2)} &= \operatorname{diag}\left(\left(\frac{\prod_{i\in\mathcal{F}_1^{(2)}}(\alpha_n - f_i)}{\prod_{i\in\mathcal{F}_1^{(2)}}(f_1 - f_i)}\right)\mathbf{I}_K, \cdots, \left(\frac{\prod_{i\in\mathcal{F}_L^{(2)}}(\alpha_n - f_i)}{\prod_{i\in\mathcal{F}_L^{(2)}}(f_L - f_i)}\right)\mathbf{I}_K\right) && (7.86)\\
&= \operatorname{diag}\left(\left(\frac{\alpha_n - f_2}{f_1 - f_2}\right)\mathbf{I}_K, \left(\frac{\alpha_n - f_1}{f_2 - f_1}\right)\mathbf{I}_K, \left(\frac{\alpha_n - f_4}{f_3 - f_4}\right)\mathbf{I}_K, \left(\frac{\alpha_n - f_3}{f_4 - f_3}\right)\mathbf{I}_K, \cdots\right.\\
&\qquad\qquad \left. \cdots, \left(\frac{\alpha_n - f_L}{f_{L-1} - f_L}\right)\mathbf{I}_K, \left(\frac{\alpha_n - f_{L-1}}{f_L - f_{L-1}}\right)\mathbf{I}_K\right). && (7.87)
\end{aligned}$$

Therefore, for all $n \in [N]$, we can write

$$\begin{aligned}
&\boldsymbol{\Upsilon}_n^{(2)}\mathbf{P}_n^{(2,\theta_2)}\\
&= \operatorname{diag}\left(\left(\frac{1}{\alpha_n - f_1}\Delta_1^{(2)} + \sum_{i\in[2]}\alpha_n^{i-1}\dddot{I}_{1,i}^{(2)}\right)\mathbf{I}_K, \cdots, \left(\frac{1}{\alpha_n - f_L}\Delta_1^{(2)} + \sum_{i\in[2]}\alpha_n^{i-1}\dddot{I}_{L,i}^{(2)}\right)\mathbf{I}_K\right).
\end{aligned} \tag{7.88}$$

Accordingly, the second term in the RHS of the update equation (7.51) can be written as follows.

$$\begin{aligned}&\mathbf{\Omega}_n^{(2)}\mathbf{\Upsilon}_n^{(2)}\mathbf{P}_n^{(2,\theta_2)}\mathbf{Q}_n^{(2,\theta_2)}\\ &=\left[\begin{array}{c}\frac{1}{\alpha_n-f_1}\Delta_1^{(2)}\mathbf{e}_K(\theta_t)+\sum_{i\in[4]}\alpha_n^{i-1}\dot{\mathbf{I}}_{1,i}^{(2)}\\ \hline \frac{1}{\alpha_n-f_2}\Delta_2^{(2)}\mathbf{e}_K(\theta_t)+\sum_{i\in[4]}\alpha_n^{i-1}\dot{\mathbf{I}}_{2,i}^{(2)}\\ \hline \vdots\\ \hline \frac{1}{\alpha_n-f_L}\Delta_L^{(2)}\mathbf{e}_K(\theta_t)+\sum_{i\in[4]}\alpha_n^{i-1}\dot{\mathbf{I}}_{L,i}^{(2)}\end{array}\right],\end{aligned}\tag{7.89}$$

which guarantees the correctness of the update. It is remarkable that by the construction of ACSA null-shaper, for all $n\in\mathcal{S}_w^{(2)}$, it is guaranteed that $\mathbf{\Omega}_n^{(2)}=\mathbf{0}$, thus we have $\mathbf{S}_n^{(2)}=\mathbf{S}_n^{(1)}$ for all $n\in\mathcal{S}_w^{(2)}$. Therefore, the write dropout constraint is satisfied. It is easy to verify that the privacy and security constraints are satisfied, and in the limit as $L/K\to\infty$, the normalized upload cost is $U_2=7/2=3.5$, the normalized download cost is $D_2=6$. We can see that when the number of dropout servers is decreased, the communication efficiency is accordingly improved (e.g., $U_2<U_1,D_2>D_1$).

7.5 Discussion

Inspired by the recent interest in X-secure T-private federated submodel learning, we explored the fundamental problem of privately reading from and writing to a distributed and secure database. By interpreting the private read and write operations as secure matrix multiplications (between query vectors and stored data), and recognizing that CSA codes are natural solutions to such problems, we constructed a novel Adaptive CSA-RW scheme. ACSA-RW achieves synergistic gains from the joint design of private read and write operations because the same one hot vector representation of the desired message index needs to be secret shared for both the private read and write operations. In addition to allowing private read and write, ACSA-RW also provides elastic resilience against server dropouts, up to thresholds that are determined by the number of redundant storage dimensions. Surprisingly, ACSA-RW is able to fully update the distributed database even though the database is only partially accessible due to write-dropout servers. This is accomplished by exploiting the redundancy that is already required for secure storage. The scheme allows a memoryless operation of the database in the sense that the storage structure is preserved and users may remain oblivious of the prior history of server dropouts.

Chapter 8

Conclusion

In this dissertation, we introduced the idea of cross-subspace alignment and its applications to the problem of X-secure T-private information retrieval, X-secure T-private information retrieval with graph-based replicated storage, X-secure T-private information retrieval with MDS coded storage, secure distributed matrix multiplication, coded distributed batch computation and X-secure T-private federated submodel learning. The idea of CSA is shown to be the essential ingredient of the construction of optimal/asymptotic optimal/state-of-art approaches that minimize the download and/or communication cost of these problems. To conclude the dissertation, we present several promising directions for future work.

In terms of Chapter 2, we note that as indicated by various open problems identified in Chapter 2, XSTPIR is a fertile research avenue for future work. In particular, the capacity characterization for arbitrary K could reveal fundamentally new schemes for PIR. Especially intriguing would be the role that field size might play in such a result. Capacity of Sym-XSTPIR is another promising open problem. XSTPIR with constraints on the amount of storage per server, multi-message retrieval are other open problems that merit investigation.

For GXSTPIR studied in Chapter 3, generalizations of the private computation scheme presented in Section 3.4.3 represent an interesting problem for future work, especially because such private computation schemes are needed for GXSTPIR, as evident from the achievability proof of Theorem 3.3. Asymptotic capacity for GPIR with arbitrary graph based storage when each message is replicated 4 times is the next step for the direction initiated by Theorem 3.3. The relationship between GXSTPIR and index coding, through the connecting thread of min-rank problems that arise in both contexts is another promising research avenue. Finally, the tightness of the converse bound in Theorem 3.2 remains an interesting question. Given that the bound is tight in all cases for which the asymptotic capacity is settled so far, it is tempting to conjecture that the converse bound is tight in general. Settling this conjecture is perhaps the most important immediate objective for future work on the asymptotic capacity of GXSTPIR.

As for Chapter 4, the immediate challenge for future work is to settle the asymptotic capacity conjectures for MDS-TPIR, and also of MDS-XSTPIR, either in the affirmative by finding tight converse bounds or in the negative by finding better asymptotic achievable schemes. Beyond this, settling down the conjecture of asymptotic capacity of U-B-MDS-XSTPIR with unresponsive and Byzantine servers also merits investigation.

In Chapter 5, we studied the capacity of SDMM. In terms of future work, open problems that merit immediate attention include the many cases of SDMM where the capacity remains open. For example, the capacity of the basic $\text{SDMM}_{(\mathbf{AB},\phi)}$ setting, previously believed to be solved in [61] is shown to be still open in general, including the important case of square matrices $L = K = M > 1$ with sufficiently many servers $N > X$. From the case $L = K = M = 1$ that is already solved in this chapter, it seems that the generalization could require expanding the scope of constructions based on non-trivial monomorphic transformations, which presents an interesting research avenue. In terms of the connection to PIR, this chapter highlights the importance of finding the capacity characterizations for MM-XSTPIR, as well as MM-XSTPC. Evidently solutions to these PIR problems would not only add to the growing literature on PIR that already includes many successful capacity characterizations [102, 106, 103, 11, 57, 107, 81], but also have a ripple effect on important problems that are intimately connected to PIR. Similar to PIR, the models of SDMM could also be further enriched to include privacy of retrieved information, coded storage [11, 36, 105], storage size and repair constraints [113, 4, 127] and generalized forms of side-information [58, 112, 22]. Thus, just like PIR, SDMM offers a fertile research landscape for discovering new coding structures and converse arguments.

In respect of the problem of CDBC studied in Chapter 6, an interesting direction for future work is the possibility of task partitioning (similar to matrix partitioning) for N-CSA codes to reduce the computation cost per server in settings where latency constraints prevent any server from fully computing the N-linear map, or the multivariate polynomial evaluation by itself.

Finally, for the problem of XSTPFSL considered in Chapter 7, as indicated in Section 7.3.1, the converse argument of the asymptotic capacity of XSTPFSL is still open in general. Another promising direction for future work is to explore applications of the idea of write dropout resilience to multi-version coding [124].

Bibliography

[1] M. Abadi, A. Chu, I. Goodfellow, H. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. *Proceedings of CCS*, 2016.

[2] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 195–203. ACM, 1987.

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

[4] M. A. Attia, D. Kumar, and R. Tandon. The capacity of private information retrieval from uncoded storage constrained databases. *IEEE Transactions on Information Theory*, 66(11):6617–6634, 2020.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[6] 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)*, pages 1993–1997. IEEE, 2018.

[7] K. Banawan, B. Arasli, Y.-P. Wei, and S. Ulukus. The capacity of private information retrieval from heterogeneous uncoded caching databases. *IEEE Transactions on Information Theory*, 66(6):3407–3416, 2020.

[8] K. Banawan and S. Ulukus. The capacity of private information retrieval from byzantine and colluding databases. *IEEE Transactions on Information Theory*, 65(2):1206–1219, 2018.

[9] K. Banawan and S. Ulukus. Multi-message private information retrieval: Capacity results and near-optimal schemes. *IEEE Transactions on Information Theory*, 64(10):6842–6862, 2018.

[10] K. Banawan and S. Ulukus. Multi-message private information retrieval: Capacity results and near-optimal schemes. *IEEE Transactions on Information Theory*, 2018.

[11] K. Banawan and S. Ulukus. The Capacity of Private Information Retrieval from Coded Databases. *IEEE Transactions on Information Theory*, 64(3):1945–1956, 2018.

[12] K. Banawan and S. Ulukus. The capacity of private information retrieval from byzantine and colluding databases. *IEEE Transactions on Information Theory*, 65(2):1206–1219, Feb 2019.

[13] K. Banawan and S. Ulukus. Private information retrieval from non-replicated databases. *arXiv preprint arXiv:1901.00004*, 2019.

[14] K. Banawan and S. Ulukus. Private information retrieval through wiretap channel ii: Privacy meets security. *IEEE Transactions on Information Theory*, 66(7):4129–4149, 2020.

[15] Y. Birk and T. Kol. Informed-source coding-on-demand (ISCOD) over broadcast channels. In *Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM'98*, volume 3, pages 1257–1264, 1998.

[16] R. Bitar and S. E. Rouayheb. Staircase-pir: Universally robust private information retrieval. *arXiv preprint arXiv:1806.08825*, 2018.

[17] W. Chang and R. Tandon. On the upload versus download cost for secure and private matrix multiplication. *arXiv preprint arXiv:1906.10684*, 2019.

[18] W.-T. Chang and R. Tandon. On the capacity of secure distributed matrix multiplication. *arXiv preprint arXiv:1806.00469*, 2018.

[19] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pages 903–912. PMLR, 2018.

[20] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar. Gcsa codes with noise alignment for secure coded multi-party batch matrix multiplication. *IEEE Journal on Selected Areas in Information Theory*, 2(1):306–316, 2021.

[21] Z. Chen, Z. Wang, and S. A. Jafar. The asymptotic capacity of private search. *IEEE Transactions on Information Theory*, 66(8):4709–4721, 2020.

[22] Z. Chen, Z. Wang, and S. A. Jafar. The capacity of t-private information retrieval with private side information. *IEEE Transactions on Information Theory*, 66(8):4761–4773, 2020.

[23] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[24] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private Information Retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[25] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251, 1990.

[26] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 2006.

[27] D. H. Duong, P. K. Mishra, and M. Yasuda. Efficient secure matrix multiplication over lwe-based homomorphic encryption. *Tatra Mountains Mathematical Publications*, 67(1):69–83, 2016.

[28] 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:1811.1075*, Nov. 2018.

[29] 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*, pages 2100–2108, 2016.

[30] S. Dutta, V. Cadambe, and P. Grover. Coded convolution for parallel and distributed computing within a deadline. *arXiv preprint arXiv:1705.03875*, 2017.

[31] 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*, 66(1):278–301, 2019.

[32] R. G. D'Oliveira, S. El Rouayheb, and D. Karpuk. Gasp codes for secure distributed matrix multiplication. *IEEE Transactions on Information Theory*, 66(7):4038–4050, 2020.

[33] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 1. Elsevier, 1977.

[34] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. *Proceedings of FOCS*, 2018.

[35] T. Finck, G. Heinig, and K. Rost. An inversion formula and fast algorithms for cauchy-vandermonde matrices. *Linear algebra and its applications*, 183:179–191, 1993.

[36] R. Freij-Hollanti, O. Gnilke, C. Hollanti, and D. Karpuk. Private Information Retrieval from Coded Databases with Colluding Servers. *SIAM Journal on Applied Algebra and Geometry*, 1(1):647–664, 2017.

[37] M. Gasca, J. Martinez, and G. Mühlbach. Computation of Rational Interpolants with Prescribed Poles. *Journal of Computational and Applied Mathematics*, 26(3):297–309, 1989.

[38] A. Gerasoulis. A fast algorithm for the multiplication of generalized hilbert matrices with vectors. *Mathematics of Computation*, 50(181):179–188, 1988.

[39] A. Gerasoulis, M. D. Grigoriadis, and L. Sun. A fast algorithm for trummer's problem. *SIAM journal on Scientific and Statistical Computing*, 8(1):s135–s138, 1987.

[40] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160. ACM, 1998.

[41] I. Gohberg and V. Olshevsky. Fast Algorithms with Preprocessing for matrix-Vector multiplication Problems. *Journal of Complexity*, 10(4):411–427, 1994.

[42] G. Golub. Trummer's problem. *SIGACT news*, 17(2):12, 1985.

[43] P. Gopalan, C.Huang, H. Simitci, and S. Yekhanin. On the Locality of Codeword Symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, Nov. 2012.

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

[45] C. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM Journal on Computing*, 18(4):658–669, 1989.

[46] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 262–271. ACM, 2004.

[47] S. A. Jafar. Blind Interference Alignment. *IEEE Journal of Selected Topics in Signal Processing*, 6(3):216–227, June 2012.

[48] T. Jahani-Nezhad and M. A. Maddah-Ali. Codedsketch: A coding scheme for distributed computation of approximated matrix multiplication. *IEEE Transactions on Information Theory*, pages 1–1, 2021.

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

[50] Z. Jia. On the capacity of weakly-private information retrieval. Master's thesis, UC Irvine, 2019.

[51] Z. Jia and S. Jafar. On the Capacity of Secure Distributed Matrix Multiplication. *ArXiv:1908.06957*, 2019.

[52] Z. Jia and S. A. Jafar. x-secure t-private federated submodel learning with elastic dropout resilience. *arXiv preprint arXiv:2010.01059v2*, 2020.

[53] Z. Jia and S. A. Jafar. On the asymptotic capacity of x-secure t-private information retrieval with graph-based replicated storage. *IEEE Transactions on Information Theory*, 66(10):6280–6296, 2020.

[54] Z. Jia and S. A. Jafar. X-secure t-private information retrieval from mds coded storage with byzantine and unresponsive servers. *IEEE Transactions on Information Theory*, 66(12):7427–7438, 2020.

[55] Z. Jia and S. A. Jafar. Cross subspace alignment codes for coded distributed batch computation. *IEEE Transactions on Information Theory*, pages 1–1, 2021.

[56] Z. Jia, H. Sun, and S. Jafar. The capacity of private information retrieval with disjoint colluding sets. In *IEEE GLOBECOM*, 2017.

[57] Z. Jia, H. Sun, and S. A. Jafar. Cross subspace alignment and the asymptotic capacity of x-secure t-private information retrieval. *IEEE Transactions on Information Theory*, 65(9):5783–5798, 2019.

[58] S. Kadhe, B. Garcia, A. Heidarzadeh, S. El Rouayheb, and A. Sprintson. Private information retrieval with side information. *IEEE Transactions on Information Theory*, 66(4):2032–2043, 2019.

[59] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[60] J. Kakar, S. Ebadifar, and A. Sezgin. Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation. *arXiv preprint arXiv:1810.13006*, 2018.

[61] J. Kakar, S. Ebadifar, and A. Sezgin. On the Capacity and Straggler-Robustness of Distributed Secure Matrix Multiplication. *IEEE Access*, 7:45783–45799, 2019.

[62] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 80–86. ACM, 2000.

[63] M. Kim and J. Lee. Private secure coded computation. *IEEE Communications Letters*, pages 1–1, 2019, doi: 10.1109/LCOMM.2019.2934436.

[64] M. Kim and J. Lee. Information-theoretic privacy in federated submodel learning. *arXiv preprint arXiv:2008.07656*, 2020.

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

[66] E. Kushilevitz and T. Mour. Sub-logarithmic distributed oblivious ram with small block size. In *IACR International Workshop on Public Key Cryptography*, pages 3–33. Springer, 2019.

[67] F. Le Gall. Powers of tensors and fast matrix multiplication. *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, 2014.

[68] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2017.

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

[70] S. Li and M. Gastpar. Single-server multi-message private information retrieval with side information. *arXiv preprint arXiv:1808.05797*, 2018.

[71] S. Li and M. Gastpar. Single-server multi-user private information retrieval with side information. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1954–1958. IEEE, 2018.

[72] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr. Coding for distributed fog computing. *IEEE Communications Magazine*, 55(4):34–40, 2017.

[73] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[74] S. Lu and R. Ostrovsky. Distributed oblivious ram for secure two-party computation. In *Theory of Cryptography Conference*, pages 377–396. Springer, 2013.

[75] Y. Lu, Z. Jia, and S. A. Jafar. Double blind t-private information retrieval. *IEEE Journal on Selected Areas in Information Theory*, 2(1):428–440, 2021.

[76] O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313, 2012.

[77] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi. Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–40, 2019.

[78] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[79] H. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. *Proceedings of ICLR*, 2018.

[80] H. B. McMahan et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1), 2021.

[81] M. Mirmohseni and M. A. Maddah-Ali. Private function retrieval. *arXiv preprint arXiv:1711.04677*, 2017.

[82] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[83] 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*, 280:31–46, 2001.

[84] V. Pan, M. A. Tabanjeh, Z. Chen, E. Landowne, and A. Sadikou. New transformations of cauchy matrices and trummer's problem. *Computers & Mathematics with Applications*, 35(12):1–5, 1998.

[85] V. Y. Pan. *Structured matrices and polynomials: unified superfast algorithms*. Springer Science & Business Media, 2012.

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

[87] M. O. Rabin. How to exchange secrets with oblivious transfer. In *Technical Report TR-81, Aiken Computation Laboratory, Harvard University*, 1981.

[88] N. Raviv and D. A. Karpuk. Private polynomial computation from lagrange encoding. *IEEE Transactions on Information Forensics and Security*, 15:553–563, 2019.

[89] N. Raviv, I. Tamo, and E. Yaakobi. Private information retrieval in graph-based replication systems. *IEEE Transactions on Information Theory*, 66(6):3590–3602, 2019.

[90] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011.

[91] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr. Coded computation over heterogeneous clusters. *IEEE Transactions on Information Theory*, 2019.

[92] A. Schönhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7(3-4):281–292, 1971.

[93] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer, 2003. ISBN: 978-3-540-44389-6, ISSN: 0937-5511.

[94] A. Severinson, A. G. i Amat, and E. Rosnes. Block-diagonal and lt codes for distributed computing with straggling servers. *IEEE Transactions on Communications*, 67(3):1739–1753, 2018.

[95] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[96] 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)*, pages 1113–1120. IEEE, 2018.

[97] M. J. Siavoshani, S. P. Shariatpanahi, and M. A. Maddah-Ali. Private information retrieval for a multi-message scenario with private side information. *IEEE Transactions on Communications*, pages 1–1, 2021.

[98] J.-y. Sohn, D.-J. Han, B. Choi, and J. Moon. Election coding for distributed learning: Protecting signsgd against byzantine attacks. *Advances in Neural Information Processing Systems*, 33, 2020.

[99] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[100] 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)*, pages 1271–1278. IEEE, 2017.

[101] H. Sun and S. A. Jafar. Blind interference alignment for private information retrieval. *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 560–564, 2016.

[102] H. Sun and S. A. Jafar. The Capacity of Private Information Retrieval. *IEEE Transactions on Information Theory*, 63(7):4075–4088, July 2017.

[103] H. Sun and S. A. Jafar. The capacity of symmetric private information retrieval. *IEEE Transactions on Information Theory*, 65(1):322–329, 2018.

[104] H. Sun and S. A. Jafar. Multiround Private Information Retrieval: Capacity and Storage Overhead. *IEEE Transactions on Information Theory*, 64(8):5743–5754, August 2018.

[105] H. Sun and S. A. Jafar. Private Information Retrieval from MDS Coded Data with Colluding Servers: Settling a Conjecture by Freij-Hollanti et al. *IEEE Transactions on Information Theory*, 64(2):1000–1022, February 2018.

[106] H. Sun and S. A. Jafar. The Capacity of Robust Private Information Retrieval with Colluding Databases. *IEEE Transactions on Information Theory*, 64(4):2361–2370, April 2018.

[107] H. Sun and S. A. Jafar. The capacity of private computation. *IEEE Transactions on Information Theory*, 65(6):3880–3897, June 2019.

[108] R. Tajeddine, O. W. Gnilke, and S. El Rouayheb. Private Information Retrieval from MDS Coded Data in Distributed Storage Systems. *IEEE Transactions on Information Theory*, 2018.

[109] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, and C. Hollanti. Private information retrieval from coded storage systems with colluding, byzantine, and unresponsive servers. *IEEE Transactions on Information Theory*, 65(6):3898–3906, June 2019.

[110] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, C. Hollanti, and S. El Rouayheb. Private information retrieval schemes for coded data with arbitrary collusion patterns. *IEEE International Symposium on Information Theory (ISIT)*, pages 1908–1912, 2017.

[111] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, C. Hollanti, and S. E. Rouayheb. Private Information Retrieval Schemes for Coded Data with Arbitrary Collusion Patterns. *arXiv preprint arXiv:1701.07636*, 2017.

[112] R. Tandon. The capacity of cache aided private information retrieval. *arXiv preprint arXiv:1706.07035*, 2017.

[113] R. Tandon, S. Amuru, T. C. Clancy, and R. M. Buehrer. Toward optimal secure distributed storage systems with exact repair. *IEEE Transactions on Information Theory*, 62(6):3477–3492, 2016.

[114] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *International Conference on Machine Learning*, pages 3368–3376, 2017.

[115] C. Tian, H. Sun, and J. Chen. Capacity-achieving private information retrieval codes with optimal message size and upload cost. *IEEE Transactions on Information Theory*, 65(11):7613–7627, 2019.

[116] Q. Wang and M. Skoglund. Linear symmetric private information retrieval for MDS coded distributed storage with colluding servers. *arXiv preprint arXiv:1708.05673*, 2017.

[117] Q. Wang and M. Skoglund. Secure private information retrieval from colluding databases with eavesdroppers. *arXiv preprint arXiv:1710.01190*, 2017.

[118] Q. Wang and M. Skoglund. On pir and symmetric pir from colluding databases with adversaries and eavesdroppers. *IEEE Transactions on Information Theory*, 65(5):3183–3197, 2019.

[119] Q. Wang, H. Sun, and M. Skoglund. The ϵ-error capacity of symmetric pir with byzantine adversaries. *arXiv preprint arXiv:1809.03988*, 2018.

[120] Q. Wang, H. Sun, and M. Skoglund. The capacity of private information retrieval with eavesdroppers. *IEEE Transactions on Information Theory*, 65(5):3198–3214, 2019.

[121] S. Wang, J. Liu, and N. Shroff. Coded sparse matrix multiplication. In *International Conference on Machine Learning*, pages 5152–5160. PMLR, 2018.

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

[123] Z. Wang, K. Banawan, and S. Ulukus. Private set intersection: A multi-message symmetric private information retrieval perspective. *arXiv preprint arXiv:1912.13501*, 2019.

[124] Z. Wang and V. R. Cadambe. Multi-version coding—an information-theoretic perspective of consistent distributed storage. *IEEE Transactions on Information Theory*, 64(6):4540–4561, 2017.

[125] W. Waterhouse. How often do determinants over finite fields vanish? *Discrete Mathematics*, (65):103–104, 1987.

[126] Y. Wei, K. Banawan, and S. Ulukus. The capacity of private information retrieval with partially known private side information. *IEEE Transactions on Information Theory*, 65(12):8222–8231, 2019.

[127] Y. Wei, K. Banawan, and S. Ulukus. Fundamental limits of cache-aided private information retrieval with unknown and uncoded prefetching. *IEEE Transactions on Information Theory*, 65(5):3215–3232, May 2019.

[128] Y.-P. Wei, B. Arasli, K. Banawan, and S. Ulukus. The capacity of private information retrieval from decentralized uncoded caching databases. *Information*, 10(12):372, 2019.

[129] H. Yang and J. Lee. Secure Distributed Computing with Straggling Servers Using Polynomial Codes. *IEEE Transactions on Information Forensics and Security*, 14(1):141–150, Jan. 2019.

[130] H. Yang, W. Shin, and J. Lee. Private information retrieval for secure distributed storage systems. *IEEE Transactions on Information Forensics and Security*, 13(12):2953–2964, December 2018.

[131] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2), 2019.

[132] Z.-H. Yang and Y.-J. Hu. Displacement Structure Approach to Cauchy and Cauchy–Vandermonde Matrices: Inversion Formulas and Fast Algorithms. *Journal of Computational and Applied Mathematics*, 138(2):259–272, 2002.

[133] A. C. Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[134] X. Yao, N. Liu, and W. Kang. The capacity of multi-round private information retrieval from byzantine databases. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2124–2128. IEEE, 2019.

[135] X. Yao, N. Liu, and W. Kang. The capacity of multi-round private information retrieval from byzantine databases. *arXiv preprint arXiv:1901.06907*, 2019.

[136] S. Yekhanin. *Locally Decodable Codes and Private Information Retrieval Schemes.* PhD thesis, Massachusetts Institute of Technology, 2007.

[137] Q. Yu and A. S. Avestimehr. Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication. *IEEE Transactions on Communications*, 69(1):59–72, 2021.

[138] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr. Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1215–1225. PMLR, 2019.

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

[140] 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.

[141] 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*, 66(3):1920–1933, 2020.

[142] Y. Zhang and G. Ge. Private information retrieval from MDS coded databases with colluding servers under several variant models. *arXiv preprint arXiv:1705.03186*, 2017.

[143] R. Zhou, C. Tian, H. Sun, and T. Liu. Capacity-achieving private information retrieval codes from mds-coded databases with minimum message size. *IEEE Transactions on Information Theory*, 66(8):4904–4916, 2020.

Appendix A

Appendix of Chapter 2

A.1 Proof of Lemma 2.7

Let $\mathbf{J}_k$ denote the $k \times k$ anti-diagonal identity matrix, and let $\mathbf{0}_{k_1 \times k_2}$ denote the $k_1 \times k_2$ matrix where all elements are equal to 0 (when $k_1 = k_2$, this notation is further simplified to $\mathbf{0}_{k_1}$). Define

$$\mathbf{I}'_k = \begin{bmatrix} \mathbf{I}_k & \mathbf{0}_{k \times 1} \\ \mathbf{0}_{1 \times k} & 0 \end{bmatrix}. \tag{A.1}$$

Choose $\mathbf{B}$ as follows.

$$\mathbf{B} = \begin{cases} \begin{bmatrix} \mathbf{I}_{\frac{K}{2}} & \mathbf{J}_{\frac{K}{2}} \\ \mathbf{J}_{\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}} \end{bmatrix}, & \text{if } K \text{ is even,} \\ \\ \left[\begin{array}{c|c} \mathbf{J}_{\frac{K+1}{2}} + \mathbf{I}'_{\frac{K-1}{2}} + \mathbf{I}_{\frac{K+1}{2}} & \begin{matrix} \mathbf{J}_{\frac{K-1}{2}} \\ \mathbf{0}_{1\times\frac{K-1}{2}} \end{matrix} \\ \hline \begin{matrix} \mathbf{J}_{\frac{K-1}{2}} & \mathbf{0}_{\frac{K-1}{2}\times 1} \end{matrix} & \mathbf{0}_{\frac{K-1}{2}} \end{array}\right], & \text{if } K \text{ is odd.} \end{cases} \tag{A.2}$$

For example,

$$\text{when } K=4: \quad \mathbf{B} = \begin{bmatrix} 1\,0\,0\,1 \\ 0\,1\,1\,0 \\ 0\,1\,0\,0 \\ 1\,0\,0\,0 \end{bmatrix}, \qquad \text{and when } K=5: \quad \mathbf{B} = \begin{bmatrix} 0\,0\,1\,0\,1 \\ 0\,1\,0\,1\,0 \\ 1\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix}. \tag{A.3}$$

Let us show that $\mathbf{B}$ and $\mathbf{I}_K + \mathbf{B}$ are both invertible.

First, consider $\mathbf{B}$. Regardless of whether K is even or odd, $\mathbf{B}$ is an upper anti-triangular matrix where all anti-diagonal elements are 1 so that $\det(\mathbf{B}) = 1$ and $\mathbf{B}$ has full rank.

Next, consider $\mathbf{I}_K + \mathbf{B}$.

$$\text{When } K \text{ is even: } \mathbf{I}_K + \mathbf{B} = \begin{bmatrix} \mathbf{I}_{\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}} \end{bmatrix} + \begin{bmatrix} \mathbf{I}_{\frac{K}{2}} & \mathbf{J}_{\frac{K}{2}} \\ \mathbf{J}_{\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\frac{K}{2}} & \mathbf{J}_{\frac{K}{2}} \\ \mathbf{J}_{\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}} \end{bmatrix} \tag{A.4}$$

$$\Rightarrow \quad \det(\mathbf{I}_K + \mathbf{B}) = 1. \tag{A.5}$$

$$\text{When } K \text{ is odd: } \mathbf{I}_K+\mathbf{B}=\left[\begin{array}{c|c} \mathbf{I}_{\frac{K+1}{2}} & \mathbf{0}_{\frac{K+1}{2}\times\frac{K-1}{2}} \\ \hline \mathbf{0}_{\frac{K-1}{2}\times\frac{K+1}{2}} & \mathbf{I}_{\frac{K-1}{2}} \end{array}\right]+\left[\begin{array}{c|c} \mathbf{J}_{\frac{K+1}{2}}+\mathbf{I}'_{\frac{K-1}{2}}+\mathbf{I}_{\frac{K+1}{2}} & \begin{array}{c}\mathbf{J}_{\frac{K-1}{2}} \\ \mathbf{0}_{1\times\frac{K-1}{2}}\end{array} \\ \hline \mathbf{J}_{\frac{K-1}{2}} \quad \mathbf{0}_{\frac{K-1}{2}\times 1} & \mathbf{0}_{\frac{K-1}{2}} \end{array}\right] \tag{A.6}$$

$$=\left[\begin{array}{c|c} \mathbf{J}_{\frac{K+1}{2}}+\mathbf{I}'_{\frac{K-1}{2}} & \begin{array}{c}\mathbf{J}_{\frac{K-1}{2}} \\ \mathbf{0}_{1\times\frac{K-1}{2}}\end{array} \\ \hline \mathbf{J}_{\frac{K-1}{2}} \quad \mathbf{0}_{\frac{K-1}{2}\times 1} & \mathbf{I}_{\frac{K-1}{2}} \end{array}\right] \tag{A.7}$$

$$\Rightarrow \det(\mathbf{I}_K+\mathbf{B})=\det\left(\mathbf{J}_{\frac{K+1}{2}}+\mathbf{I}'_{\frac{K-1}{2}}+\begin{bmatrix}\mathbf{J}_{\frac{K-1}{2}} \\ \mathbf{0}_{1\times\frac{K-1}{2}}\end{bmatrix}\mathbf{I}^{-1}_{\frac{K-1}{2}}\begin{bmatrix}\mathbf{J}_{\frac{K-1}{2}} & \mathbf{0}_{\frac{K-1}{2}\times 1}\end{bmatrix}\right) \tag{A.8}$$

$$=\det\left(\mathbf{J}_{\frac{K+1}{2}}+\mathbf{I}'_{\frac{K-1}{2}}+\mathbf{I}'_{\frac{K-1}{2}}\right)=\det\left(\mathbf{J}_{\frac{K+1}{2}}\right)=1 \tag{A.9}$$

where (A.8) follows from the following formula on the determinant of a block matrix that is made up of matrices $\mathbf{A}$,$\mathbf{B}$,$\mathbf{C}$,$\mathbf{D}$ with proper dimensions and $\mathbf{D}$ is invertible.

$$\det\begin{pmatrix}\mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{pmatrix}=\det(\mathbf{D})\det\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right). \tag{A.10}$$

The proof is thus complete. □

A.2 Proof of Corollaries 2.1, 2.2, 2.3, 2.4

The proof of Corollary 2.1 is trivial because imposing the symmetric security constraint cannot increase capacity.

A.2.1 Proof of Corollary 2.2

To prove Corollary 2.2 we provide a scheme as follows. Each message $W_k, k \in [1:K]$, consists of $L=1$ symbol from some finite field $\mathbb{F}_q$. Let $Z_{x,k,m}, x \in [1:X], k \in [1:K], m \in [1:K]$ be independent uniform noise symbols from $\mathbb{F}_q$. The subscript, m, in $Z_{x,k,m}$ is interpreted modulo K, i.e., $Z_{x,k,m} = Z_{x,k,m+K}$. The storage at each server is specified as,

$$S_n = \{Z_{n,k,m}, k \in [1:K], m \in [1:K]\}, \qquad n \in [1:X], \tag{A.11}$$

$$S_n = \left\{ W_k + \sum_{x=1}^{X} Z_{x,k,m}, k \in [1:K], m \in [1:K] \right\}, \qquad n = N. \tag{A.12}$$

The queries from each server are specified as,

$$Q_n^{[\theta]} : \text{Ask for } \{Z_{n,k,m_o}, k \in [1:K]\}, \qquad n \in [1:X], \tag{A.13}$$

$$Q_n^{[\theta]} : \text{Ask for } \{W_k + \sum_{x=1}^{X} Z_{x,k,m_o-\theta+k}, k \in [1:K]\}, \qquad n = N, \tag{A.14}$$

where m_o is chosen privately and uniformly randomly by the user from $[1:K]$. Thus, in order to retrieve 1 desired message symbol, the user downloads a total of KN symbols from all servers. The scheme is X-secure because each message symbol is protected by independent uniform noise terms. It is correct because for $k=\theta$ the download from Server N, contains the symbol $W_\theta + \sum_{x=1}^{X} Z_{x,\theta,m_o}$ and the downloads from the first X servers include all the noise terms Z_{x,θ,m_o}. The scheme is private because m_o is chosen uniformly and privately by the user. It satisfies symmetric security because all the undesired message symbols $W_k, k \neq \theta$, contained in the answers are protected by noise terms $Z_{x,k,m_o-\theta+k}$ and these noise terms are independent of the noise terms downloaded from servers $n \in [1:X]$ because $m_o - \theta + k \neq m_o$ when $k \neq \theta$. The rate achieved is $\frac{1}{KN}$, which is the capacity for this setting. $\square$

Note that in the Sym-XSPIR scheme described above, each server stores K^2 symbols, when the total data is only $KL = K$ symbols. Thus, this Sym-XSPIR scheme takes advantage of unconstrained storage when K is large, more so than the XSTPIR schemes which store no more than KL symbols at each server.

A.2.2 Proof of Corollary 2.3

To prove Corollary 2.3, we show that the scheme presented in Section 2.6 automatically guarantees symmetric security when $T=1$. Define

$$\mathbf{W}_i^c = \{\mathbf{W}_l, l \in [1:L], l \neq i\} \tag{A.15}$$

$$\mathbf{Z}_{ij}^c = \{\mathbf{Z}_{lx}, l \in [1:L], x \in [1:X], (l,x) \neq (i,j)\}. \tag{A.16}$$

We need to prove that beyond the information that the user must have, i.e., $W_\theta, Q_{[1:N]}^{[\theta]}, \theta$, he cannot learn anything about the messages $\mathbf{W}_{[1:L]}$ from the answers $A_{[1:N]}^{[\theta]}$.

$$I\left(\mathbf{W}_{[1:L]}; A_{[1:N]}^{[\theta]} \mid W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) = \sum_{l \in [1:L]} I\left(\mathbf{W}_l; A_{[1:N]}^{[\theta]} \mid \mathbf{W}_{[1:l-1]}, W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) \tag{A.17}$$

$$\leq \sum_{l \in [1:L]} I\left(\mathbf{W}_l; A_{[1:N]}^{[\theta]} \mid \mathbf{W}_l^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) \tag{A.18}$$

$$\leq \sum_{l \in [1:L]} I\left(\mathbf{W}_l; A_{[1:N]}^{[\theta]} \mid \mathbf{Z}_{l1}^c, \mathbf{W}_l^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) \tag{A.19}$$

where we repeatedly used the fact that $I(A;B \mid C) \leq I(A;B \mid C,D)$ if $I(A;D \mid C)=0$ and the facts that

$$I(\mathbf{W}_l; \mathbf{W}_{[l+1:L]} \mid W_{[1:l-1]}, W_\theta, Q_{[1:N]}^{[\theta]}, \theta) = 0 \tag{A.20}$$

$$I\left(\mathbf{W}_l; \mathbf{Z}_{l1}^c \mid \mathbf{W}_l^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) = 0 \tag{A.21}$$

that follow from the indepenence of messages, queries, and the noise terms, by construction of the scheme in Section 2.6. To prove Corollary 2.3 it suffices to show that each of the terms in the summation is zero. Without loss of generality, let us consider $l=1$. Because of the conditioning on $\mathbf{Z}_{11}^c, \mathbf{W}_1^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta$, we can subtract the contributions from these terms, whose values are fixed, from $A_n^{[\theta]}$, leaving us with only

$$A'^{[\theta]}_n = (\mathbf{W}_1 + (f_1 - \alpha_n)\mathbf{Z}_{11}) \left(\frac{\Delta_n}{f_1 - \alpha_n}\right) (\mathbf{Q}_\theta + (f_1 - \alpha_n)\mathbf{Z}_1') \tag{A.22}$$

$$= \left(\frac{\Delta_n}{f_1 - \alpha_n}\right) \mathbf{W}_1\mathbf{Q}_\theta + \Delta_n(\mathbf{W}_1\mathbf{Z}_1' + \mathbf{Z}_{11}\mathbf{Q}_\theta) + \Delta_n(f_1 - \alpha_n)\mathbf{Z}_{11}\mathbf{Z}_1' \tag{A.23}$$

$$= \left(\frac{\Delta_n}{f_1 - \alpha_n}\right) W_{\theta 1} + \Delta_n(\mathbf{W}_1\mathbf{Z}_1' + \mathbf{Z}_{11}(\theta)) + \Delta_n(f_1 - \alpha_n)\mathbf{Z}_{11}\mathbf{Z}_1' \tag{A.24}$$

where $\mathbf{Z}_{11}(i)$ is the i^{th} element of the vector $\mathbf{Z}_{11}$. Note that $W_{\theta 1}$ is also a constant because of the conditioning on W_θ. Given $\mathbf{Z}_{11}^c, \mathbf{W}_1^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta$, the random variable $A_{[1:N]}^{[\theta]}$ is an invertible function of $A'^{[\theta]}_{[1:N]}$.

$$I\left(\mathbf{W}_1; A_{[1:N]}^{[\theta]} \mid \mathbf{Z}_{11}^c, \mathbf{W}_1^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) \tag{A.25}$$

$$= I\left(\mathbf{W}_1; A'^{[\theta]}_{[1:N]} \mid \mathbf{Z}_{11}^c, \mathbf{W}_1^c, W_\theta, Q_{[1:N]}^{[\theta]}, \theta\right) \tag{A.26}$$

$$= I\left(\mathbf{W}_1; A'^{[\theta]}_{[1:N]} \mid W_{\theta 1}, Q_{[1:N]}^{[\theta]}, \theta\right) \tag{A.27}$$

$$\leq I(\mathbf{W}_1; \mathbf{W}_1\mathbf{Z}_1' + \mathbf{Z}_{11}(\theta), \mathbf{Z}_{11}\mathbf{Z}_1' \mid W_{\theta 1}, \mathbf{Q}_\theta, \theta) \tag{A.28}$$

$$\leq I(\mathbf{W}_1; \mathbf{W}_1\mathbf{Z}_1' + \mathbf{Z}_{11}(\theta), \mathbf{Z}_{11}\mathbf{Z}_1' \mid W_{\theta 1}, \mathbf{Q}_\theta, \mathbf{Z}_1', \theta) \tag{A.29}$$

$$= 0. \tag{A.30}$$

In (A.28) we used the fact that given $W_{\theta 1}$, the random variable $A'^{[\theta]}_{[1:N]}$ is a function of $\mathbf{W}_1\mathbf{Z}_1' + \mathbf{Z}_{11}(\theta), \mathbf{Z}_{11}\mathbf{Z}_1'$ because of (A.24), and the fact that for any random variables A, B, C, we must have $I(A; f(B) \mid C) \leq I(A; B \mid C)$. In (A.29) we used the fact that conditioning on an independent random variable cannot reduce mutual information, i.e., $I(A; B \mid C) \leq I(A; B \mid C, D)$ if $I(A; D \mid C) = 0$, and the fact that $\mathbf{Z}_1'$ is independent of $\mathbf{W}_1$ after conditioning on $W_{\theta 1}, \mathbf{Q}_\theta, \theta$ by construction of the scheme as described in Section 2.6. The last step is justified as follows. Because of the conditioning on $\mathbf{Z}_1'$, its value is a constant for which there are only three possibilities: $\mathbf{Z}_1'$ is either the zero vector, or it is equal to $\mu\mathbf{Q}_\theta$ for some non-zero $\mu \in \mathbb{F}_q$, or it is neither zero nor equal to $\mu\mathbf{Q}_\theta$. If $\mathbf{Z}_1'$ is the zero vector, then the mutual information is automatically zero because $\mathbf{W}_1$ is eliminated entirely. If $\mathbf{Z}_1' = \mu\mathbf{Q}_\theta$ for some non-zero μ, then $\mathbf{W}_1\mathbf{Z}_1' = \mu W_{\theta 1}$ and the mutual information is again zero because of the conditioning on $W_{\theta 1}$. Finally, if $\mathbf{Z}_1'$ is neither zero nor a scaled version of $\mathbf{Q}_\theta$, then $\mathbf{Z}_{11}\mathbf{Z}_1'$ is a sum of uniformly random noise terms in $\mathbb{F}_q$, at least one of which is independent of $\mathbf{Z}_{11}(\theta)$ and $\mathbf{Z}_1'$. So in this case also the mutual information is zero. This completes the proof of Corollary 2.3. □

A.2.3 Proof of Corollary 2.4

The proof of Corollary 2.4 is presented next. Recall that in the scheme of the proof of Theorem 2.4, the user obtains the following three symbols from the answers,

$$\mathbf{W}\mathbf{Q}_\theta = W_\theta \tag{A.31}$$

$$\mathbf{W}\mathbf{Z}' + \mathbf{Z}\mathbf{Z}' \tag{A.32}$$

$$\mathbf{W}\mathbf{Z}' + \mathbf{Z}\mathbf{B}\mathbf{Z}' + \mathbf{Z}\mathbf{B}\mathbf{Q}_\theta. \tag{A.33}$$

We show that symmetric security holds, i.e., conditioned on $\mathbf{Z}'$, from these three symbols the user learns nothing about the undesired messages $W_1, \cdots, W_{\theta-1}, W_{\theta+1}, \cdots, W_K$. When $\mathbf{Z}'$ is the zero vector, the symbol $\mathbf{W}\mathbf{Z}'$ is zero as well, leaking nothing about the undesired messages. Now consider (A.32). If $\mathbf{Z}'$ is not the zero vector, then the symbol $\mathbf{W}\mathbf{Z}'$ is protected by an independent noise term. Similarly, consider (A.33) and consider three possibilities: $\mathbf{B}(\mathbf{Z}' + \mathbf{Q}_\theta)$ is either zero, or equal to $\mathbf{Z}'$, or not zero and not equal to $\mathbf{Z}'$. If $\mathbf{B}(\mathbf{Z}' + \mathbf{Q}_\theta)$ is the zero vector, then because $\mathbf{B}$ is invertible, we must have $\mathbf{Z}' = \mathbf{Q}_\theta$, so the symbol $\mathbf{W}\mathbf{Z}' = \mathbf{W}\mathbf{Q}_\theta$ is the desired message, again leaking nothing about undesired messages. If $\mathbf{B}(\mathbf{Z}' + \mathbf{Q}_\theta) = \mathbf{Z}'$ then (A.33) is redundant, i.e., same as (A.32), so it leaks no new information. Finally, if

$\mathbf{B}(\mathbf{Z}'+\mathbf{Q}_\theta)$ is not zero and not equal to $\mathbf{Z}'$, then $\mathbf{Z}\mathbf{B}(\mathbf{Z}'+\mathbf{Q}_\theta)$ is independent of $\mathbf{Z}\mathbf{Z}'$, so that (A.33) is protected by an independent noise term. Therefore, in all cases, the user learns nothing about undesired messages, and this completes the proof of symmetric security. $\square$

Appendix B

Appendix of Chapter 3

B.1 Lemmas

Lemma B.1. *The optimal value of total normalized download,* $\min_{\mathcal{D}}(D_1+D_2+\cdots+D_N)$*, in Theorem 3.2 is equal to the fractional matching number of* $\mathcal{G}[\mathcal{V},\mathcal{E}]$*.*

Proof. Let us consider the non-degenerate scenario, $\rho_{\min}>X+T$, because otherwise the asymptotic capacity is zero. According to Theorem 3.2, the optimal value of total normalized download $\min_{\mathcal{D}}(D_1+D_2+\cdots+D_N)$ is expressed as the result of the following linear program.

$$D^* = \min \sum_{n\in[N]} D_n \tag{B.1}$$

$$\text{such that,} \tag{B.2}$$

$$\sum_{n:\ n\in e} D_n \geq 1, \qquad \forall e \in \mathcal{E} \tag{B.3}$$

$$D_n \geq 0, \qquad \forall n \in [N] \tag{B.4}$$

Since the linear program is bounded and feasible, by the strong duality of linear programming, we have as its dual the following linear program.

$$D^* = \max \sum_{e \in \mathcal{E}} x_e \tag{B.5}$$

$$\text{such that,} \tag{B.6}$$

$$\sum_{e:\ e \ni n} x_e \leq 1, \qquad \forall n \in [N] \tag{B.7}$$

$$x_e \geq 0, \qquad \forall e \in \mathcal{E} \tag{B.8}$$

Thus, the optimal converse bound D^* is precisely the maximum weight of a fractional 1-matching in $\mathcal{G}$. Therefore, the converse bound in Theorem 3.2 coincides with the achievability bound in Theorem 3.1 if and only if $D^* = \frac{N}{\rho_{\min} - X - T}$. This completes the proof of Lemma 3.1. □

Lemma B.2. *For distinct non-zero values $\beta_1, \cdots, \beta_n$ and for $v_1, \cdots, v_n$ defined as*

$$v_i \triangleq \left(\prod_{j \in [n] \setminus \{i\}} (\beta_i - \beta_j) \right)^{-1}, \qquad i \in [n] \tag{B.9}$$

the following identity is satisfied,

$$\sum_{i \in [n]} v_i \beta_i^j = 0, \qquad \forall j \in \{0, 1, \cdots, n-2\}. \tag{B.10}$$

Proof. The proof of Lemma B.2 follows directly from the properties of dual GRS codes for which we refer the reader to [33]. For our purpose let us recall that given two n-dimensional vectors

$$\mathbf{u} = [u_1, u_2, \cdots, u_n] \tag{B.11}$$

$$\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_n] \tag{B.12}$$

where $u_1,u_2,\cdots,u_n$ are non-zero, while $\beta_1,\beta_2,\cdots,\beta_n$ are non-zero and distinct, the canonical generator matrix for the Generalized Reed-Solomon code $\text{GRS}_{k,n}(\mathbf{u},\boldsymbol{\beta})$ is given by

$$\begin{bmatrix} u_1 & u_2 & \cdots & u_n \\ u_1\beta_1 & u_2\beta_2 & \cdots & u_n\beta_n \\ \vdots & \vdots & \cdots & \vdots \\ u_1\beta_1^{k-1} & u_2\beta_2^{k-1} & \cdots & u_n\beta_n^{k-1} \end{bmatrix} \tag{B.13}$$

The dual code of a GRS code is also a GRS code. Specifically, the dual for $\text{GRS}_{k,n}(\mathbf{u},\boldsymbol{\beta})$ is $\text{GRS}_{n-k,n}(\mathbf{v},\boldsymbol{\beta})$ where $\mathbf{v}=[v_1,v_2,\cdots,v_n]$ and $v_i=\left(u_i\prod_{j\in[n]\setminus\{i\}}(\beta_i-\beta_j)\right)^{-1}$. For the purpose of Lemma B.2 let us set $u_1=u_2=\cdots=u_n=1$. Since the dual of a code C is a code $C^{\perp}$ that spans the null space of C, we have

$$\begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ v_1\beta_1 & v_2\beta_2 & \cdots & v_n\beta_n \\ \vdots & \vdots & \cdots & \vdots \\ v_1\beta_1^{k-1} & v_2\beta_2^{k-1} & \cdots & v_n\beta_n^{k-1} \end{bmatrix} \begin{bmatrix} 1 & \beta_1 & \cdots & \beta_1^{n-k-1} \\ 1 & \beta_2 & \cdots & \beta_2^{n-k-1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \beta_n & \cdots & \beta_n^{n-k-1} \end{bmatrix} = \mathbf{0} \tag{B.14}$$

which implies that

$$\sum_{i\in[n]} v_i\beta_i^j = 0 \tag{B.15}$$

for $j\in\{0,1,\cdots,n-2\}$. This completes the proof of Lemma B.2. □

Lemma B.3. *For two positive integers n,L such that $n>L$, and for distinct non-zero values $\beta_1,\cdots,\beta_n$ such that $\beta_i+\ell\neq 0,\forall i\in[n],\ell\in[L]$ and for $v_1,\cdots,v_n$ defined as*

$$v_i \triangleq \left(\prod_{j\in[n]\setminus\{i\}}(\beta_i-\beta_j)\right)^{-1}, \qquad i\in[n] \tag{B.16}$$

the following $L\times L$ matrix

$$\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \beta_1 & \cdots & \beta_n \\ \vdots & \vdots & \vdots \\ \beta_1^{L-1} & \cdots & \beta_n^{L-1} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \frac{v_1}{f_1-\beta_1} & \cdots & \frac{v_1}{f_L-\beta_1} \\ \vdots & \vdots & \vdots \\ \frac{v_n}{f_1-\beta_n} & \cdots & \frac{v_n}{f_L-\beta_n} \end{bmatrix}}_{\mathbf{B}} \tag{B.17}$$

is invertible.

Proof. Let us define the matrix $\mathbf{C}$ as follows.

$$\mathbf{C} = \begin{bmatrix} v_1 & v_1\beta_1 & \dots & v_1\beta_1^{n-L-1} \\ \vdots & \vdots & \vdots & \vdots \\ v_n & v_n\beta_n & \dots & v_n\beta_n^{n-L-1} \end{bmatrix} \tag{B.18}$$

Guaranteed by Lemma 2.5, which is also a standard result for Cauchy-Vandermonde matrices [37], the $n\times n$ matrix $[\mathbf{B}|\mathbf{C}]$ is invertible. Besides, guaranteed by Lemma B.2 and the definitions of $\beta_1,\cdots,\beta_n$ and $v_1,\cdots,v_n$, the rows of the $L\times n$ matrix $\mathbf{A}$ generate the null space of the matrix $\mathbf{C}$. Therefore, we have $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}\times[\mathbf{B}|\mathbf{C}]) = \text{rank}([\mathbf{AB}|\mathbf{AC}]) = \text{rank}([\mathbf{AB}|\mathbf{0}]) = \text{rank}(\mathbf{AB})$, where $\mathbf{0}$ is the $L\times(n-L)$ zero matrix. Note that the rank of the transposed Vandermonde matrix $\mathbf{A}$ is L, thus we have $\text{rank}(\mathbf{AB}) = L$, which indicates that the matrix $\mathbf{AB}$ is invertible. This completes the proof of Lemma B.3. □

Lemma B.4. *For all $m\in[M], k\in[K_m]$, $\mathcal{X}\subset\mathcal{R}_m$, $|\mathcal{X}|\leq X$,*

$$I\left(S_{[N]\setminus\mathcal{R}_m}, S_{\mathcal{X}}; W_{m,k}, Q_{[N]}^{[m,k]}\right) = 0. \tag{B.19}$$

Proof.

$$
\begin{aligned}
&I(S_{[N]\setminus\mathcal{R}_m},S_{\mathcal{X}};W_{m,k},Q_{[N]}^{[m,k]}) && \text{(B.20)}\\
&=I(W_{m,k};S_{[N]\setminus\mathcal{R}_m},S_{\mathcal{X}})+I(Q_{[N]}^{[m,k]};S_{[N]\setminus\mathcal{R}_m},S_{\mathcal{X}}\mid W_{m,k}) && \text{(B.21)}\\
&\leq I(W_{m,k};S_{[N]\setminus\mathcal{R}_m},S_{\mathcal{X}})+I(Q_{[N]}^{[m,k]};S_{[N]\setminus\mathcal{R}_m},S_{\mathcal{X}},W_{m,k}) && \text{(B.22)}\\
&=I(W_{m,k};S_{[N]\setminus\mathcal{R}_m},S_{\mathcal{X}}) && \text{(B.23)}\\
&\leq I(W_{m,k};\overline{\mathcal{W}}',\overline{W}_{m,k}^{(\mathcal{X})}) && \text{(B.24)}\\
&=I(W_{m,k};\overline{W}_{m,k}^{(\mathcal{X})})+I(W_{m,k};\overline{\mathcal{W}}'|\overline{W}_{m,k}^{(\mathcal{X})}) && \text{(B.25)}\\
&=I(W_{m,k};\overline{\mathcal{W}}'|\overline{W}_{m,k}^{(\mathcal{X})}) && \text{(B.26)}\\
&\leq I(W_{m,k},\overline{W}_{m,k}^{(\mathcal{X})};\overline{\mathcal{W}}') && \text{(B.27)}\\
&\leq I(\overline{W}_{m,k};\overline{\mathcal{W}}') && \text{(B.28)}\\
&=0. && \text{(B.29)}
\end{aligned}
$$

where $\overline{\mathcal{W}}'=(\overline{W}_{m',k'},\forall m'\in[M],k'\in[K_m],(m',k')\neq(m,k))$, and $\overline{W}_{m,k}^{(\mathcal{X})}=(\overline{W}_{m,k}^{(n)},n\in\mathcal{X})$. Steps of the proof are justified as follows. (B.21) and (B.22) follow from the chain rule and the non-negativity of mutual information. (B.23) follows from (3.18), while (B.24), follows from the definition of replicated storage in (3.11). (B.25) is the chain rule of mutual information, while (B.26) follows from the security constraint in (3.16). (B.27) follows from chain rule and the non-negativity of mutual information. In (B.28) we used the fact that $(W_{m,k},\overline{W}_{m,k}^{(\mathcal{X})})$ is function of $\overline{W}_{m,k}$, and the last step follows from (3.8). This completes the proof of Lemma B.4. □

Lemma B.5. *For all $m\in[M],k\in[K_m]$, $\mathcal{X},\mathcal{T}\subset\mathcal{R}_m$,*

$$I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{[N]}^{[m,k]})\leq I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]}). \tag{B.30}$$

Proof.

$$I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{[N]}^{[m,k]})$$

$$=H(A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{[N]}^{[m,k]})-H(A_{\mathcal{T}}^{[m,k]}\mid W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{[N]}^{[m,k]}) \tag{B.31}$$

$$\leq H(A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})-H(A_{\mathcal{T}}^{[m,k]}\mid W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{[N]}^{[m,k]}) \tag{B.32}$$

$$\begin{aligned}&=H(A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})-H(A_{\mathcal{T}}^{[m,k]}\mid W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})\\&\quad+H(A_{\mathcal{T}}^{[m,k]}\mid W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})-H(A_{\mathcal{T}}^{[m,k]}\mid W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{[N]}^{[m,k]})\end{aligned} \tag{B.33}$$

$$=I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})+I(A_{\mathcal{T}}^{[m,k]};Q_{[N]}^{[m,k]}\mid W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]}) \tag{B.34}$$

$$\leq I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})+I(A_{\mathcal{T}}^{[m,k]},W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m};Q_{[N]}^{[m,k]}\mid Q_{\mathcal{T}}^{[m,k]}) \tag{B.35}$$

$$\leq I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})+I(A_{\mathcal{T}}^{[m,k]},S_{[N]};Q_{[N]}^{[m,k]}\mid Q_{\mathcal{T}}^{[m,k]}) \tag{B.36}$$

$$\begin{aligned}&=I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]})\\&\quad+I(S_{[N]};Q_{[N]}^{[m,k]}\mid Q_{\mathcal{T}}^{[m,k]})+I(A_{\mathcal{T}}^{[m,k]};Q_{[N]}^{[m,k]}\mid S_{[N]},Q_{\mathcal{T}}^{[m,k]})\end{aligned} \tag{B.37}$$

$$=I(W_{m,k};A_{\mathcal{T}}^{[m,k]}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]}) \tag{B.38}$$

(B.31) follows from the definition of mutual information, (B.32) follows because dropping conditioning cannot reduce entropy, (B.33) adds and subtracts the same term so nothing changes, (B.34) uses the definition of mutual information, (B.35) uses the chain rule of mutual information and the fact that mutual information is always non-negative, (B.36) uses the fact that $\left(W_{m,k},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m}\right)$ is a function of $S_{[N]}$ according to (3.9) and (3.11), and (B.37) uses chain rule of mutual information. For (B.38) we use the fact that $S_{[N]}$ is independent of $Q_{[N]}^{[m,k]}$ according to (3.18), and $A_{\mathcal{T}}^{[m,k]}$ is fully determined by $S_{[N]},Q_{\mathcal{T}}^{[m,k]}$ according to (3.20). This completes the proof of Lemma B.5. □

Lemma B.6. *For any $m\in[M]$, $\mathcal{T}\subset\mathcal{R}_m$, $|\mathcal{T}|\leq T$,*

$$I(Q_{\mathcal{T}}^{[m,\kappa]},A_{\mathcal{T}}^{[m,\kappa]},S_{[N]};\kappa)=0 \tag{B.39}$$

Proof.

$$I(Q_{\mathcal{T}}^{[m,\kappa]},A_{\mathcal{T}}^{[m,\kappa]},S_{[N]};\kappa)=I(Q_{\mathcal{T}}^{[m,\kappa]};\kappa)+I(S_{[N]};\kappa\mid Q_{\mathcal{T}}^{[m,\kappa]})+I(A_{\mathcal{T}}^{[m,\kappa]};\kappa\mid S_{[N]},Q_{\mathcal{T}}^{[m,\kappa]}) \tag{B.40}$$

$$=I(Q_{\mathcal{T}}^{[m,\kappa]};\kappa)+I(S_{[N]};\kappa\mid Q_{\mathcal{T}}^{[m,\kappa]}) \tag{B.41}$$

$$\leq I(Q_{\mathcal{T}}^{[m,\kappa]};\kappa)+I(S_{[N]};\kappa,Q_{\mathcal{T}}^{[m,\kappa]}) \tag{B.42}$$

$$=0 \tag{B.43}$$

(B.40) is the chain rule of mutual information, (B.41) follows because $A_{\mathcal{T}}^{[\mu,\kappa]}$ is fully determined by $S_{[N]},Q_{\mathcal{T}}^{[\mu,\kappa]}$ according to (3.20). The next step, (B.42) follows because of the chain rule of mutual information and the non-negativity of mutual information, and (B.43) follows from (3.18),(3.19). This completes the proof of Lemma B.6. □

Lemma B.7. *For any $m\in[M],k\in[K_m]$ and subsets $\mathcal{X},\mathcal{T}\subset\mathcal{R}_m$ such that $|\mathcal{X}|\leq X$,*

$$I\Big(\mathcal{W}_{m,\mathcal{K}}\ ;\ \mathcal{W}_{m,\mathcal{K}'}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k]}\Big)=0 \tag{B.44}$$

where $\mathcal{K}\subset[K_m]$, $\mathcal{K}'=[K_m]\setminus\mathcal{K}$, $\mathcal{W}_{m,\mathcal{K}}=(W_{m,k},k\in\mathcal{K})$ and $\mathcal{W}_{m,\mathcal{K}'}=(W_{m,k},k\in\mathcal{K}')$.

Proof. Let us define $\overline{\mathcal{W}}_{\mathcal{M}'}=(\overline{W}_{m',k},\forall m'\in[M],k\in[K_{m'}],m'\neq m)$. $\overline{\mathcal{W}}_{m,\mathcal{K}}=(\overline{W}_{m,k},k\in\mathcal{K})$. $\overline{\mathcal{W}}_{m,\mathcal{K}'}=(\overline{W}_{m,k},k\in\mathcal{K}')$. $\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})}=(\overline{\mathcal{W}}_{m,k}^{(n)},n\in\mathcal{X},k\in\mathcal{K})$. $\overline{\mathcal{W}}_{m,\mathcal{K}'}^{(\mathcal{X})}=(\overline{\mathcal{W}}_{m,k}^{(n)},n\in\mathcal{X},k\in\mathcal{K}')$.

$$I(\mathcal{W}_{m,\mathcal{K}};\mathcal{W}_{m,\mathcal{K}'}\mid S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k']}) \tag{B.45}$$

$$\leq I(\mathcal{W}_{m,\mathcal{K}};\mathcal{W}_{m,\mathcal{K}'},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m},Q_{\mathcal{T}}^{[m,k']}) \tag{B.46}$$

$$=I(\mathcal{W}_{m,\mathcal{K}};\mathcal{W}_{m,\mathcal{K}'},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m})+I(\mathcal{W}_{m,\mathcal{K}};Q_{\mathcal{T}}^{[m,k']}\mid\mathcal{W}_{m,\mathcal{K}'},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m}) \tag{B.47}$$

$$\leq I(\mathcal{W}_{m,\mathcal{K}};\mathcal{W}_{m,\mathcal{K}'},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m})+I(Q_{\mathcal{T}}^{[m,k']};\mathcal{W}_{m,\mathcal{K}},\mathcal{W}_{m,\mathcal{K}'},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m}) \tag{B.48}$$

$$=I(\mathcal{W}_{m,\mathcal{K}};\mathcal{W}_{m,\mathcal{K}'},S_{\mathcal{X}},S_{[N]\setminus\mathcal{R}_m}) \tag{B.49}$$

$$\leq I(\mathcal{W}_{m,\mathcal{K}};\mathcal{W}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{\mathcal{M}'},\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})},\overline{\mathcal{W}}_{m,\mathcal{K}'}^{(\mathcal{X})}) \tag{B.50}$$

$$\leq I(\mathcal{W}_{m,\mathcal{K}};\overline{\mathcal{W}}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{\mathcal{M}'},\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})}) \tag{B.51}$$

$$=I(\mathcal{W}_{m,\mathcal{K}};\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})})+I(\mathcal{W}_{m,\mathcal{K}};\overline{\mathcal{W}}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{\mathcal{M}'}\mid\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})}) \tag{B.52}$$

$$=I(\mathcal{W}_{m,\mathcal{K}};\overline{\mathcal{W}}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{\mathcal{M}'}\mid\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})}) \tag{B.53}$$

$$\leq I(\mathcal{W}_{m,\mathcal{K}},\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})};\overline{\mathcal{W}}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{\mathcal{M}'}) \tag{B.54}$$

$$\leq I(\overline{\mathcal{W}}_{m,\mathcal{K}};\overline{\mathcal{W}}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{\mathcal{M}'}) \tag{B.55}$$

$$=0. \tag{B.56}$$

(B.46), (B.47), (B.48) follow from the chain rule and the non-negativity of mutual information. (B.49) holds because of (3.18), while in (B.50), we used the definition of the storage as in (3.11). (B.51) follows because $\left(\mathcal{W}_{m,\mathcal{K}'},\overline{\mathcal{W}}_{m,\mathcal{K}'}^{(\mathcal{X})}\right)$ is function of $\overline{\mathcal{W}}_{m,\mathcal{K}'}$. (B.52) is again the chain rule of mutual information, and (B.53) follows from the X-security constraint as in (3.16). (B.54) follows from the chain rule and the non-negativity of mutual information, while in (B.55), we used the fact that $\left(\mathcal{W}_{m,\mathcal{K}},\overline{\mathcal{W}}_{m,\mathcal{K}}^{(\mathcal{X})}\right)$ is function of $\overline{\mathcal{W}}_{m,\mathcal{K}}$. The last step holds because of (3.8). This completes the proof of Lemma B.7. □

Appendix C

Appendix of Chapter 5

C.1 Multi-Message X-Secure T-Private Information Retrieval

C.1.1 Problem Statement

Consider K messages stored at N distributed servers, $W_1,W_2,\ldots,W_K$. Each message is represented by L random symbols from the finite field $\mathbb{F}_q$.

$$H(W_1)=H(W_2)=\cdots=H(W_K)=L, \tag{C.1}$$

$$H(W_{[K]})=KL, \tag{C.2}$$

in q-ary units. The information stored at the n-th server is denoted by S_n, $n\in[N]$. X-secure storage, $0\leq X\leq N$, guarantees that any X (or fewer) colluding servers learns nothing about messages.

$$I(S_{\mathcal{X}};W_{[K]})=0, \quad \forall \mathcal{X}\subset[N],|\mathcal{X}|=X. \tag{C.3}$$

To make information retrieval possible, messages must be function of $S_{[N]}$.

$$H(W_{[K]}|S_{[N]})=0. \tag{C.4}$$

The multi-message T-private information retrieval allows the user to retrieve M messages simultaneously. The user privately and uniformly generate a set of indices of desired messages $\mathcal{K}$, $\mathcal{K}\subset[K],|\mathcal{K}|=M$. To retrieve desired messages privately, the user generates N queries $Q^{\mathcal{K}}_{[N]}$. The n-th query $Q^{\mathcal{K}}_{n}$ is sent to the n-th server. The user has no prior knowledge of information stored at servers, i.e.,

$$I(S_{[N]};\mathcal{K},Q^{\mathcal{K}}_{[N]})=0. \tag{C.5}$$

T-privacy, $0\leq T\leq N$, guarantees that any T (or fewer) colluding servers learns nothing about $\mathcal{K}$.

$$I(Q^{\mathcal{K}}_{\mathcal{T}},S_{\mathcal{T}};\mathcal{K})=0, \quad \forall \mathcal{T}\subset[N],|\mathcal{T}|=T. \tag{C.6}$$

Upon receiving user's query $Q^{\mathcal{K}}_{n}$, the n-th server responds user with an answer $A^{\mathcal{K}}_{n}$, which is function of the query and its storage, i.e.,

$$H(A^{\mathcal{K}}_{n}|Q^{\mathcal{K}}_{n},S_n)=0. \tag{C.7}$$

The user must be able to recover desired messages $W_{\mathcal{K}}$ from all answers $A^{\mathcal{K}}_{[N]}$.

$$H(W_{\mathcal{K}}|A^{\mathcal{K}}_{[N]},Q^{\mathcal{K}}_{[N]},\mathcal{K})=0. \tag{C.8}$$

The rate of a multi-message XSTPIR scheme is defined by the number of q-ary symbols of desired messages that are retrieved per downloaded q-ary symbol,

$$R = \frac{H(W_{\mathcal{K}})}{\sum_{n\in[N]} A_n^{\mathcal{K}}} = \frac{ML}{D}. \tag{C.9}$$

$D = \sum_{n\in[N]} A_n^{\mathcal{K}}$ is expected number of downloaded q-ary symbols from all servers. The capacity of multi-message XSTPIR is the supremum of rate over all feasible schemes, denoted as $C_{\text{MM-XSTPIR}}(N,X,T,K,M)$.

Note that setting $X = 0$ reduces the problem to basic multi-message T-private information retrieval where storage is not secure. The setting $T = 0$ reduces the problem to X-secure storage with no privacy constraint.

C.1.2 Upper Bound of the Capacity of Multi-Message XSTPIR

To prove Theorem 5.1, we need following lemmas.

Lemma C.1. *For all $\mathcal{K},\mathcal{K}' \subset [K]$, $|\mathcal{K}| = |\mathcal{K}'| = M$, $\forall \mathcal{T} \subset [N], |\mathcal{T}| = T$, we have*

$$\left(Q_{\mathcal{T}}^{\mathcal{K}}, A_{\mathcal{T}}^{\mathcal{K}}, S_{[N]}, W_{[K]}\right) \sim \left(Q_{\mathcal{T}}^{\mathcal{K}'}, A_{\mathcal{T}}^{\mathcal{K}'}, S_{[N]}, W_{[K]}\right) \tag{C.10}$$

Proof. It suffices to prove $I\left(Q_{\mathcal{T}}^{\mathcal{K}}, A_{\mathcal{T}}^{\mathcal{K}}, S_{[N]}, W_{[K]}; \mathcal{K}\right) = 0$. The proof is presented as follows.

$$\begin{aligned}
& I\left(Q_{\mathcal{T}}^{\mathcal{K}}, A_{\mathcal{T}}^{\mathcal{K}}, S_{[N]}, W_{[K]}; \mathcal{K}\right) && \text{(C.11)}\\
=& I(Q_{\mathcal{T}}^{\mathcal{K}}; \mathcal{K}) + I(S_{[N]}, W_{[K]}; \mathcal{K} | Q_{\mathcal{T}}^{\mathcal{K}}) + I(A_{\mathcal{T}}^{\mathcal{K}}; \mathcal{K} | Q_{\mathcal{T}}^{\mathcal{K}}, S_{[N]}, W_{[K]}) && \text{(C.12)}\\
=& I(Q_{\mathcal{T}}^{\mathcal{K}}; \mathcal{K}) + I(S_{[N]}, W_{[K]}; \mathcal{K} | Q_{\mathcal{T}}^{\mathcal{K}}) && \text{(C.13)}\\
=& I(Q_{\mathcal{T}}^{\mathcal{K}}; \mathcal{K}) + I(S_{[N]}; \mathcal{K} | Q_{\mathcal{T}}^{\mathcal{K}}) && \text{(C.14)}\\
\leq& I(Q_{\mathcal{T}}^{\mathcal{K}}; \mathcal{K}) + I(S_{[N]}; \mathcal{K}, Q_{\mathcal{T}}^{\mathcal{K}}) && \text{(C.15)}
\end{aligned}$$

$$=0. \tag{C.16}$$

Steps are justified as follows. (C.12) is the chain rule of mutual information. (C.13) holds from the fact that $A^{\mathcal{K}}_{\mathcal{T}}$ is function of $(Q^{\mathcal{K}}_{\mathcal{T}}, S_{[N]})$ according to (C.7). (C.14) follows because $W_{[K]}$ is function of $S_{[N]}$, according to (C.4). (C.15) follows from the chain rule and non-negativity of mutual information. In last step, we simply used (C.5) and (C.6). This completes the proof of Lemma C.1. □

Lemma C.2. *For all $\mathcal{T},\mathcal{X} \subset [N]$, $|\mathcal{T}| = T, |\mathcal{X}| = X$, $\forall \mathcal{K},\mathcal{K}' \subset [K]$, $|\mathcal{K}| = |\mathcal{K}'| = M$, $\forall \kappa \subset [K]$, we have*

$$H(A^{\mathcal{K}}_{\mathcal{T}}|S_{\mathcal{X}}, Q^{\mathcal{K}}_{[N]}, W_\kappa) = H(A^{\mathcal{K}}_{\mathcal{T}}|S_{\mathcal{X}}, Q^{\mathcal{K}}_{\mathcal{T}}, W_\kappa). \tag{C.17}$$

Proof.

$$H(A^{\mathcal{K}}_{\mathcal{T}}|S_{\mathcal{X}}, Q^{\mathcal{K}}_{\mathcal{T}}, W_\kappa) - H(A^{\mathcal{K}}_{\mathcal{T}}|S_{\mathcal{X}}, Q^{\mathcal{K}}_{[N]}, W_\kappa) \tag{C.18}$$

$$= I(A^{\mathcal{K}}_{\mathcal{T}}; Q^{\mathcal{K}}_{[N]}|S_{\mathcal{X}}, Q^{\mathcal{K}}_{\mathcal{T}}, W_\kappa) \tag{C.19}$$

$$\leq I(A^{\mathcal{K}}_{\mathcal{T}}, S_{\mathcal{X}}, W_\kappa; Q^{\mathcal{K}}_{[N]}|Q^{\mathcal{K}}_{\mathcal{T}}) \tag{C.20}$$

$$\leq I(A^{\mathcal{K}}_{\mathcal{T}}, S_{[N]}, W_\kappa; Q^{\mathcal{K}}_{[N]}|Q^{\mathcal{K}}_{\mathcal{T}}) \tag{C.21}$$

$$= I(A^{\mathcal{K}}_{\mathcal{T}}, S_{[N]}; Q^{\mathcal{K}}_{[N]}|Q^{\mathcal{K}}_{\mathcal{T}}) \tag{C.22}$$

$$= I(S_{[N]}; Q^{\mathcal{K}}_{[N]}|Q^{\mathcal{K}}_{\mathcal{T}}) + I(A^{\mathcal{K}}_{\mathcal{T}}; Q^{\mathcal{K}}_{[N]}|Q^{\mathcal{K}}_{\mathcal{T}}, S_{[N]}) \tag{C.23}$$

$$= I(S_{[N]}; Q^{\mathcal{K}}_{[N]}|Q^{\mathcal{K}}_{\mathcal{T}}) \tag{C.24}$$

$$\leq I(S_{[N]}; Q^{\mathcal{K}}_{[N]}) \tag{C.25}$$

$$= 0. \tag{C.26}$$

Steps are justified as follows. (C.19) is the definition of mutual information. (C.20) follows from the chain rule and non-negativity of mutual information. In (C.21), we added terms in mutual information. (C.22) holds from the fact that $W_{[K]}$ is function of $S_{[N]}$, according to

(C.4). (C.23) is the chain rule of mutual information. (C.24) follows from the fact that $A_{\mathcal{T}}^{\mathcal{K}}$ is fully determined by $(Q_{\mathcal{T}}^{\mathcal{K}}, S_{[N]})$ according to (C.7). (C.25) follows from the chain rule and non-negativity of mutual information, while the last step holds from (C.5). This completes the proof of Lemma C.2. □

Lemma C.3. *Denote D_n the expected number of q-ary symbols downloaded from the n-th server. For all $\mathcal{X} \subset [N]$, $|\mathcal{X}| = X$, $\overline{\mathcal{X}} = [N] \setminus \mathcal{X}$, $\forall \mathcal{K}_1 \subset [K]$, $|\mathcal{K}_1| = M$, we have*

$$ML \leq \sum_{n \in \overline{\mathcal{X}}} D_n - H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}, W_{\mathcal{K}_1}). \tag{C.27}$$

Proof.

$$\begin{align}
ML = H(W_{\mathcal{K}_1}) &= I(W_{\mathcal{K}_1}; A_{[N]}^{\mathcal{K}_1} | Q_{[N]}^{\mathcal{K}_1}) \tag{C.28}\\
&\leq I(W_{\mathcal{K}_1}; A_{[N]}^{\mathcal{K}_1}, S_{\mathcal{X}} | Q_{[N]}^{\mathcal{K}_1}) \tag{C.29}\\
&= I(W_{\mathcal{K}_1}; S_{\mathcal{X}} | Q_{[N]}^{\mathcal{K}_1}) + I(W_{\mathcal{K}_1}; A_{[N]}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.30}\\
&\leq I(W_{\mathcal{K}_1}, Q_{[N]}^{\mathcal{K}_1}; S_{\mathcal{X}}) + I(W_{\mathcal{K}_1}; A_{[N]}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.31}\\
&= I(W_{\mathcal{K}_1}; S_{\mathcal{X}}) + I(Q_{[N]}^{\mathcal{K}_1}; S_{\mathcal{X}} \mid W_{\mathcal{K}_1}) + I(W_{\mathcal{K}_1}; A_{[N]}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.32}\\
&\leq I(Q_{[N]}^{\mathcal{K}_1}; W_{\mathcal{K}_1}, S_{\mathcal{X}}) + I(W_{\mathcal{K}_1}; A_{[N]}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.33}\\
&= I(W_{\mathcal{K}_1}; A_{\mathcal{X}}^{\mathcal{K}_1}, A_{\overline{\mathcal{X}}}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.34}\\
&= I(W_{\mathcal{K}_1}; A_{\overline{\mathcal{X}}}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.35}\\
&= H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) - H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_1} | W_{\mathcal{K}_1}, S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}) \tag{C.36}\\
&\leq \sum_{n \in \overline{\mathcal{X}}} D_n - H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_1} | S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}_1}, W_{\mathcal{K}_1}). \tag{C.37}
\end{align}$$

Steps are justified as follows. (C.28) follows from (C.8), while in (C.29), we add terms in mutual information. In (C.30), (C.31), (C.32) and (C.33), we repeatedly used the chain rule and non-negativity of mutual information, while (C.33) and (C.34) holds from the independence of query and storage, according to (C.5). (C.35) holds from the fact that

$A_{\mathcal{X}}^{\mathcal{K}_1}$ is fully determined by $(Q_{[N]}^{\mathcal{K}_1}, S_{\mathcal{X}})$ according to (C.7). (C.36) is the definition of mutual information, while (C.37) follows from the fact that dropping conditions can not reduce entropy. This completes the proof of Lemma C.3. □

Lemma C.4. *For all $\mathcal{X} \subset [N]$, $|\mathcal{X}| = X$, $\forall \mathcal{K} \subset [K]$, $|\mathcal{K}| = M$, $\forall \kappa \subset [K]$, we have*

$$I(W_{\mathcal{K}}; S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}}, W_{\kappa}) = |\mathcal{K} \cap \kappa| L. \tag{C.38}$$

Proof.

$$\begin{aligned}
&I(W_{\mathcal{K}}; S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}}, W_{\kappa}) \\
&= I(W_{\mathcal{K}}; W_{\kappa}) + I(W_{\mathcal{K}}; S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}} | W_{\kappa}) && \text{(C.39)} \\
&= |\mathcal{K} \cap \kappa| L + I(W_{\mathcal{K}}; S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}} | W_{\kappa}). && \text{(C.40)}
\end{aligned}$$

(C.39) is the chain rule of mutual information, and (C.40) follows from (C.1) and (C.2). Let us consider the RHS term, we have

$$\begin{aligned}
&I(W_{\mathcal{K}}; S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}} | W_{\kappa}) \\
&\leq I(S_{\mathcal{X}}, Q_{[N]}^{\mathcal{K}}; W_{\mathcal{K}}, W_{\kappa}) && \text{(C.41)} \\
&= I(S_{\mathcal{X}}; W_{\mathcal{K}}, W_{\kappa}) + I(Q_{[N]}^{\mathcal{K}}; W_{\mathcal{K}}, W_{\kappa} | S_{\mathcal{X}}) && \text{(C.42)} \\
&= I(Q_{[N]}^{\mathcal{K}}; W_{\mathcal{K}}, W_{\kappa} | S_{\mathcal{X}}) && \text{(C.43)} \\
&\leq I(Q_{[N]}^{\mathcal{K}}; W_{\mathcal{K}}, W_{\kappa}, S_{\mathcal{X}}) && \text{(C.44)} \\
&\leq I(Q_{[N]}^{\mathcal{K}}; S_{[N]}) && \text{(C.45)} \\
&= 0. && \text{(C.46)}
\end{aligned}$$

Steps are justified as follows. (C.41) and (C.42) follows from the chain rule and non-negativity of mutual information, while (C.43) follows from the X-secure constraint in (C.3). (C.44) holds from the chain rule and non-negativity of mutual information, and (C.45) follows from the fact that $(W_{\mathcal{K}},W_{\kappa},S_{\mathcal{X}})$ is function of $S_{[N]}$. The last step follows from (C.5). This completes the proof of Lemma C.4. □

Now we are ready to formally present the proof of Theorem 5.1.

Proof. First, let us consider $X<N\leq X+T$. For this setting, let us assume that $\mathcal{K}_i=[i:i+M-1],i\in[K-M+1]$. Note that by the selection of $\mathcal{K}_i$'s, $\forall i\in[K-M]$, we have

$$|\mathcal{K}_{i+1}\cap(\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i)|=(M-1). \tag{C.47}$$

Now let us consider the RHS term in (C.37). For all $i\in[K-M]$, we have

$$\begin{aligned}
&H(A_{\mathcal{X}}^{\mathcal{K}_i}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_i},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i})\\
&=H(A_{\mathcal{X}}^{\mathcal{K}_i}|S_{\mathcal{X}},Q_{\mathcal{X}}^{\mathcal{K}_i},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.48)}\\
&=H(A_{\mathcal{X}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{\mathcal{X}}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.49)}\\
&=H(A_{\mathcal{X}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.50)}\\
&=H(W_{\mathcal{K}_{i+1}},A_{\mathcal{X}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.51)}\\
&=H(W_{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i})+H(A_{\mathcal{X}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_{i+1}}) && \text{(C.52)}\\
&=L+H(A_{\mathcal{X}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_{i+1}}). && \text{(C.53)}
\end{aligned}$$

Steps are justified as follows. (C.48) follows from Lemma C.2, while (C.49) follows from Lemma C.1. (C.50) again follows from Lemma C.2. (C.51) follows from (C.7) and (C.8). (C.52) is the chain rule of entropy, while the last step follows from Lemma C.4 and (C.47). Applying (C.53) repeatedly for $i=1,2,\ldots,K-M$, we have

$$\begin{aligned}
ML&\leq\sum_{n\in\overline{\mathcal{X}}}D_n-H(A_{\mathcal{X}}^{\mathcal{K}_1}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_1},W_{\mathcal{K}_1}) && \text{(C.54)}\\
&=\sum_{n\in\overline{\mathcal{X}}}D_n-L-H(A_{\mathcal{X}}^{\mathcal{K}_2}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_2},W_{\mathcal{K}_1\cup\mathcal{K}_2}) && \text{(C.55)}
\end{aligned}$$

$$
\begin{aligned}
&=\cdots && \text{(C.56)}\\
&=\sum_{n\in\overline{\mathcal{X}}} D_n-(K-M)L. && \text{(C.57)}
\end{aligned}
$$

Averaging over all $\mathcal{X}$, we have

$$
D=\sum_{n\in[N]} D_n \geq \frac{N}{N-X}KL. \tag{C.58}
$$

Therefore we have

$$
R=\frac{ML}{D}\leq\frac{M(N-X)}{KN}. \tag{C.59}
$$

Thus

$$
C_{\text{MM-XSTPIR}}(N,X,T,K,M)\leq\frac{M(N-X)}{KN},\quad X<N\leq X+T. \tag{C.60}
$$

Next, let us consider $N>X+T$. For this setting, let us assume that $\mathcal{K}_i=\{M(i-1)+1,M(i-1)+2,\ldots,Mi\}$, $\forall i\in[\lfloor\frac{K}{M}\rfloor]$. Note that $\mathcal{K}_i$'s are disjoint sets. Similarly, let us consider the RHS term in (C.37). Consider any set $\mathcal{T}\subset\overline{\mathcal{X}}$, $|\mathcal{T}|=T$, For all $i,i+1\in[\lfloor\frac{K}{M}\rfloor]$, we have

$$
\begin{aligned}
&H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_i}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_i},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i})\\
&\geq H(A_{\mathcal{T}}^{\mathcal{K}_i}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_i},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.61)}\\
&=H(A_{\mathcal{T}}^{\mathcal{K}_i}|S_{\mathcal{X}},Q_{\mathcal{T}}^{\mathcal{K}_i},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.62)}\\
&=H(A_{\mathcal{T}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{\mathcal{T}}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.63)}\\
&=H(A_{\mathcal{T}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}). && \text{(C.64)}
\end{aligned}
$$

Steps are justified as follows. (C.61) follows from the fact that dropping terms can not increase entropy. (C.62) follows from Lemma C.2. (C.63) follows from Lemma C.1, while (C.64) again follows from Lemma C.2. Now let us average (C.64) over all $\mathcal{T}$ and apply Han's inequality.

$$
\begin{aligned}
&H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_i}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_i},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i})\\
&\geq \frac{T}{N-X}H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.65)}\\
&= \frac{T}{N-X}H(W_{\mathcal{K}_{i+1}},A_{\overline{\mathcal{X}}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i}) && \text{(C.66)}\\
&= \frac{T}{N-X}\Big(H(W_{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_i})+H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_{i+1}})\Big) && \text{(C.67)}\\
&= \frac{T}{N-X}\Big(ML+H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_{i+1}}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_{i+1}},W_{\mathcal{K}_1\cup\cdots\cup\mathcal{K}_{i+1}})\Big). && \text{(C.68)}
\end{aligned}
$$

(C.65) follows from the Han's inequality, and (C.66) follows from (C.7) and (C.8). (C.66) is the chain rule of entropy, while the last step holds from Lemma C.4 and the fact that $\mathcal{K}_i$'s are disjoint sets. Now let us apply (C.68) repeatedly for $i=1,2,\ldots,\lfloor\frac{K}{M}\rfloor-1$, we have

$$
\begin{aligned}
ML &\leq \sum_{n\in\overline{\mathcal{X}}} D_n - H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_1}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_1},W_{\mathcal{K}_1}) && \text{(C.69)}\\
&\leq \sum_{n\in\overline{\mathcal{X}}} D_n - \frac{T}{N-X}\Big(ML+H(A_{\overline{\mathcal{X}}}^{\mathcal{K}_2}|S_{\mathcal{X}},Q_{[N]}^{\mathcal{K}_2},W_{\mathcal{K}_1\cup\mathcal{K}_2})\Big) && \text{(C.70)}\\
&\leq \ldots && \text{(C.71)}\\
&\leq \sum_{n\in\overline{\mathcal{X}}} D_n - ML\left(\left(\frac{T}{N-X}\right)+\cdots+\left(\frac{T}{N-X}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right). && \text{(C.72)}
\end{aligned}
$$

Thus we have

$$
\sum_{n\in\overline{\mathcal{X}}} D_n \geq ML\left(1+\left(\frac{T}{N-X}\right)+\cdots+\left(\frac{T}{N-X}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right). \tag{C.73}
$$

Averaging over all $\mathcal{X}$, we have

$$
D=\sum_{n\in[N]} D_n \geq ML\frac{N}{N-X}\left(1+\left(\frac{T}{N-X}\right)+\cdots+\left(\frac{T}{N-X}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right). \tag{C.74}
$$

Therefore,

$$R=\frac{ML}{D}\leq\frac{N-X}{N}\left(1+\left(\frac{T}{N-X}\right)+\cdots+\left(\frac{T}{N-X}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right)^{-1}. \tag{C.75}$$

So we have,

$$\begin{aligned}&C_{\text{MM-XSTPIR}}(N,X,T,K,M)\\&\leq\frac{N-X}{N}\left(1+\left(\frac{T}{N-X}\right)+\cdots+\left(\frac{T}{N-X}\right)^{\lfloor\frac{K}{M}\rfloor-1}\right)^{-1},\quad N>X+T.\end{aligned} \tag{C.76}$$

This completes the proof of Theorem 5.1. □

Remark C.1. *Note that when $X=0$, i.e., the basic multi-message T-private information retrieval problem where storage is not secure, the proof of Theorem 5.1 follows directly, and the resulting upper bound is obtained by setting $X=0$.*

Remark C.2. *Note that when $T=0$, i.e., the problem with X-secure storage and no privacy requirement, we have*

$$\begin{aligned}&C_{MM\text{-}XSTPIR}(N,X,T=0,K,M)\\&\leq\begin{cases}0, & N\leq X,\\ \dfrac{N-X}{N}, & N>X.\end{cases}\end{aligned} \tag{C.77}$$

C.2 Proof of Lemma 5.2

To prove Lemma 5.2, we need the following lemmas.

Lemma C.5. *For independent random matrices $\bar{\mathbf{A}}\in\mathbb{F}_q^{l\times k},\bar{\mathbf{B}}\in\mathbb{F}_q^{k\times k}$, if the elements of $\bar{\mathbf{B}}$*

are i.i.d. uniform then

$$\lim_{q\to\infty} H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}}) = H(\bar{\mathbf{A}}) \tag{C.78}$$

in q-ary units.

Proof. Define σ as 0 if $\bar{\mathbf{B}}$ is singular, and 1 otherwise. Then we have

$$\begin{aligned}
H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}}) &= H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma) && \text{(C.79)}\\
&= H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma=1)P(\sigma=1) + H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma=0)P(\sigma=0) && \text{(C.80)}\\
&= H(\bar{\mathbf{A}} \,|\, \bar{\mathbf{B}},\sigma=1)P(\sigma=1) + H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma=0)P(\sigma=0) && \text{(C.81)}\\
&= H(\bar{\mathbf{A}})P(\sigma=1) + H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma=0)P(\sigma=0) && \text{(C.82)}\\
&= H(\bar{\mathbf{A}})\prod_{i=1}^{k}(1-q^{-i}) + H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma=0)\left(1-\prod_{i=1}^{k}(1-q^{-i})\right) && \text{(C.83)}
\end{aligned}$$

In (C.81) we used the fact that given a square non-singular (invertible) matrix $\bar{\mathbf{B}}$, the matrix $\bar{\mathbf{A}}\bar{\mathbf{B}}$ is an invertible function of the matrix $\bar{\mathbf{A}}$. In (C.83) we used the result from [125] that the probability of a matrix $\bar{\mathbf{B}}$ drawn uniformly from $\mathbb{F}_q^{k\times k}$ being singular is exactly $1-\prod_{i=1}^{k}(1-q^{-1})$. Now, since $H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}},\sigma=0)$ is a finite value bounded between 0 and lk, as $q\to\infty$ we have $H(\bar{\mathbf{A}}\bar{\mathbf{B}} \,|\, \bar{\mathbf{B}}) = H(\bar{\mathbf{A}})$. □

The random matrices $\mathbf{A}$,$\mathbf{B}$ in the next two lemmas are as defined in Lemma 5.2. Note that we assume that $q\to\infty$ throughout the remainder of this section.

Lemma C.6. *When $K \geq M$, let us express $\mathbf{A}$ as*

$$\mathbf{A} = [\ \underbrace{(\mathbf{A}_1)_{L\times M}}_{\textit{First } M \textit{ columns}}\ |\ \underbrace{(\mathbf{A}_2)_{L\times(K-M)}}_{\textit{Last } K-M \textit{ columns}}\ ]. \tag{C.84}$$

Similarly, let us express $\mathbf{B}$ *as*

$$\mathbf{B} = \begin{bmatrix} (\mathbf{B}_1)_{M\times M} \\ (\mathbf{B}_2)_{(K-M)\times M} \end{bmatrix} \begin{matrix} \}\textit{First } M \textit{ rows} \\ \}\textit{Last } K-M \textit{ rows.} \end{matrix} \tag{C.85}$$

Then we have

$$H(\mathbf{AB} \,|\, \mathbf{B}_1,\mathbf{B}_2,\mathbf{A}_2) = H(\mathbf{A}_1). \tag{C.86}$$

Proof. As $q \to \infty$, the square matrix $\mathbf{B}_1$ is invertible with probability 1. Therefore, using Lemma C.5, $H(\mathbf{AB} \,|\, \mathbf{B}_1,\mathbf{B}_2,\mathbf{A}_2) = H(\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 \,|\, \mathbf{B}_1,\mathbf{B}_2,\mathbf{A}_2) = H(\mathbf{A}_1\mathbf{B}_1 \,|\, \mathbf{B}_1,\mathbf{B}_2,\mathbf{A}_2) = H(\mathbf{A_1} \,|\, \mathbf{B}_1,\mathbf{B}_2,\mathbf{A}_2) = H(\mathbf{A}_1)$. □

Lemma C.7. *When* $K < M$*, let us express* $\mathbf{B}$ *as*

$$\mathbf{B} = [\; \underbrace{(\mathbf{B}_1)_{K\times K}}_{\textit{First } K \textit{ columns}} \;|\; \underbrace{(\mathbf{B}_2)_{K\times(M-K)}}_{\textit{Last } M-K \textit{ columns}} \;]. \tag{C.87}$$

Then we have

$$H(\mathbf{AB} \,|\, \mathbf{B}_1,\mathbf{B}_2) = H(\mathbf{A}). \tag{C.88}$$

In particular, when $K < L$*, we have*

$$H(\mathbf{AB} \,|\, \mathbf{B}_1) = H(\mathbf{A}) + H(\mathbf{B}_2). \tag{C.89}$$

Proof. As $q \to \infty$, the square matrix $\mathbf{B}_1$ is invertible with probability 1. Therefore, $H(\mathbf{AB} \,|\, \mathbf{B}_1,\mathbf{B}_2) = H(\mathbf{AB}_1,\mathbf{AB}_2 \,|\, \mathbf{B}_1,\mathbf{B}_2) = H(\mathbf{A},\mathbf{AB}_2 \,|\, \mathbf{B}_1,\mathbf{B}_2) = H(\mathbf{A} \,|\, \mathbf{B}_1,\mathbf{B}_2) = H(\mathbf{A})$.

When $K < L$, the matrix $\mathbf{A}$ has full column rank with probability 1, so that given $\mathbf{A}$, the matrix $\mathbf{B}_2$ is an invertible function of $\mathbf{AB}_2$. Therefore, $H(\mathbf{AB}\,|\,\mathbf{B}_1)=H(\mathbf{AB_1},\mathbf{AB_2}\,|\,\mathbf{B}_1)=H(\mathbf{A},\mathbf{AB_2}\,|\,\mathbf{B}_1)=H(\mathbf{A}\,|\,\mathbf{B}_1)+H(\mathbf{AB_2}\,|\,\mathbf{B}_1,\mathbf{A})=$ $H(\mathbf{A}\,|\,\mathbf{B}_1)+H(\mathbf{B}_2\,|\,\mathbf{B}_1,\mathbf{A})=H(\mathbf{A})+H(\mathbf{B}_2)$.

□

Now we are ready to prove the first part of Lemma 5.2.

C.2.1 Proof of Lemma 5.2: (5.20)

Proof. Case 1. $K\leq\min(L,M)$.

First, let us consider the upper bound. Note that we can rewrite matrix $\mathbf{A}$ as

$$\mathbf{A}=\begin{bmatrix}(\mathbf{A}_1)_{K\times K}\\(\mathbf{A}_2)_{(L-K)\times K}\end{bmatrix}\begin{matrix}\}\text{First } K \text{ rows}\\\}\text{Last } L-K \text{ rows.}\end{matrix}\tag{C.90}$$

Similarly, let us rewrite matrix $\mathbf{B}$ as

$$\mathbf{B}=[\ \underbrace{(\mathbf{B}_1)_{K\times K}}_{\text{First } K \text{ columns}}\ |\ \underbrace{(\mathbf{B}_2)_{K\times(M-K)}}_{\text{Last } M-K \text{ columns}}\ ].\tag{C.91}$$

Note that as $q\to\infty$, square matrices $\mathbf{A}_1$ and $\mathbf{B}_1$ are invertible with probability 1. Thus we have

$$\mathbf{A}=\begin{bmatrix}\mathbf{I}_K\\\mathbf{A}_2\mathbf{A}_1^{-1}\end{bmatrix}\mathbf{A}_1,\tag{C.92}$$

$$\mathbf{B}=\mathbf{B}_1\ \left[\mathbf{I}_K\,|\,\mathbf{B}_1^{-1}\mathbf{B}_2\right].\tag{C.93}$$

Therefore,

$$\mathbf{AB} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{A}_2\mathbf{A}_1^{-1} \end{bmatrix} (\mathbf{A}_1\mathbf{B}_1) \left[\mathbf{I}_K | \mathbf{B}_1^{-1}\mathbf{B}_2\right]. \tag{C.94}$$

Then we have

$$\begin{aligned} H(\mathbf{AB}) &\leq H(\mathbf{A}_2\mathbf{A}_1^{-1}, \mathbf{A}_1\mathbf{B}_1, \mathbf{B}_1^{-1}\mathbf{B}_2) && \text{(C.95)} \\ &\leq K(L-K) + K^2 + K(M-K) && \text{(C.96)} \\ &= LK + KM - K^2, && \text{(C.97)} \end{aligned}$$

in q-ary units. On the other hand, from (C.89) of Lemma C.7, we have

$$\begin{aligned} H(\mathbf{AB}) \geq & H(\mathbf{AB} \,|\, \mathbf{B}_1) && \text{(C.98)} \\ = & H(\mathbf{A}) + H(\mathbf{B}_2) && \text{(C.99)} \\ = & LK + KM - K^2, && \text{(C.100)} \end{aligned}$$

in q-ary units. This completes the proof of (5.20) for Case 1.

Case 2. $M \leq \min(L, K)$.

Let us consider the upper bound first. Since $\mathbf{AB}$ has dimension $L \times M$, it is trivial that $H(\mathbf{AB}) \leq LM$ in q-ary units. On the other hand, from Lemma C.6, we have $H(\mathbf{AB}) \geq H(\mathbf{AB} \,|\, \mathbf{B}_1, \mathbf{B}_2, \mathbf{A}_2) = H(\mathbf{A}_1) = LM$ in q-ary units. This completes the proof of (5.20) for Case 2.

Case 3. $L \leq \min(K, M)$. By symmetry this case is identical to Case 2. This completes the proof of (5.20) for Lemma 5.2. □

C.2.2 Proof of Lemma 5.2: (5.21),(5.22)

Proof. First let us prove that $H(\mathbf{AB}\,|\,\mathbf{B}) = \min(LM, LK)$. If $K \leq M$, then from (C.88) of Lemma C.7, we have $H(\mathbf{A}\times\mathbf{B}\,|\,\mathbf{B}) = H(\mathbf{A}) = LK$ in q-ary units. Now consider $K > M$. We have $H(\mathbf{AB}\,|\,\mathbf{B}) \leq H(\mathbf{AB}) = LM$, in q-ary units. On the other hand, from Lemma C.6, we have $H(\mathbf{A}\times\mathbf{B}\,|\,\mathbf{B}) = H(\mathbf{A}\times\mathbf{B}\,|\,\mathbf{B}_1,\mathbf{B}_2) \geq H(\mathbf{A}\times\mathbf{B}\,|\,\mathbf{B}_1,\mathbf{B}_2,\mathbf{A}_2) = H(\mathbf{A}_1) = LM$ in q-ary units. By symmetry, $H(\mathbf{A}\times\mathbf{B}|\mathbf{A}) = \min(LM, KM)$ can be similarly proved. This completes the proof of Lemma 5.2. □

Appendix D

Appendix of Chapter 6

D.1 N-CSA Codes for X-secure B-byzantine N-linear Coded Distributed Batch Computation

Let us consider the problem of X-secure B-byzantine N-linear coded distributed batch computation (XSBNCDBC) over a finite field $\mathbb{F}_q$, where the shares $\widetilde{X^{(n)}}^{[S]}, n \in [N]$ are coded in an X-secure fashion, i.e., any X colluding servers learn nothing about the data, $x_{[L]}^{(n)}$. Formally, we have

$$I\left(\widetilde{X^{(n)}}^{\mathcal{X}}; x_{[L]}^{(n)}\right) = 0, \quad \forall \mathcal{X} \subset [S], |\mathcal{X}| = X, n \in [N]. \tag{D.1}$$

Furthermore, we assume that there exists a set of servers $\mathcal{B}$, $\mathcal{B} \subset [S]$, $|\mathcal{B}| \leq B$, known as Byzantine servers. The user knows the number of Byzantine servers B but the realization of the set $\mathcal{B}$ is not known to the user *apriori*. The Byzantine servers respond to the user arbitrarily, possibly introducing errors. However, the remaining servers, i.e., all servers $s \in [S] \setminus \mathcal{B}$, if they respond at all, respond truthfully with the function h_s. We will follow the problem statement and definitions of N-CDBC in all other aspects. The goal in this section is to present a generalized N-CSA codes construction for XSBNCDBC, which achieves the recovery threshold $R=K_c(N+\ell-1)+N(X-1)+2B+1$. To construct N-CSA codes for XSBNCDBC, let $f_{1,1},f_{1,2},\cdots,f_{\ell,K_c}$ and $\alpha_1,\alpha_2,\cdots,\alpha_S$ be $(S+L)$ distinct elements from $\mathbb{F}_q$, where $q\geq S+L$. For all $n\in[N]$, let $(z_{l,k,x}^{(n)})_{l\in[\ell],k\in[K_c],x\in[X]}$ be independent uniformly random noise vectors from V_n, that are used to guarantee the security. The independence between data and random noise symbols is specified as follows.

$$H(\mathbf{x}_{[L]},(z_{l,k,x}^{(n)})_{n\in[N],l\in[\ell],k\in[K_c],x\in[X]})=H(\mathbf{x}_{[L]})+\sum_{\substack{n\in[N],l\in[\ell],\\ k\in[K_c],x\in[X]}}H(z_{l,k,x}^{(n)}). \tag{D.2}$$

For all $l\in[\ell],s\in[S]$, let us define

$$\Delta_s^{,K_c}=\prod_{k\in[K_c]}(f_{l,k}-\alpha_s). \tag{D.3}$$

For all $n\in[N],l\in[\ell],k\in[K_c]$, we define

$$x_{l,k}^{(n)}=x_{K_c(l-1)+k}^{(n)}. \tag{D.4}$$

For all $s\in[S],n\in[N]$, we construct $\widetilde{X^{(n)}}^s$ as follows.

$$\widetilde{X^{(n)}}^s=(\widetilde{X^{(n)}}_1^s,\widetilde{X^{(n)}}_2^s,\cdots,\widetilde{X^{(n)}}_\ell^s), \tag{D.5}$$

where for $l\in[\ell]$, let us set

$$\widetilde{X^{(n)}}_l^s=\Delta_s^{,K_c}\left(\sum_{k\in[K_c]}\frac{1}{f_{l,k}-\alpha_s}x_{l,k}^{(n)}+\sum_{x\in[X]}\alpha_s^{x-1}z_{l,k,x}^{(n)}\right). \tag{D.6}$$

Now it is readily seen that the X-security of data is guaranteed by the i.i.d. and uniformly distributed noise terms, i.e., $(z_{l,k,x}^{(n)})_{n\in[N],l\in[\ell],k\in[K_c],x\in[X]}$ that are coded according to an MDS(X,S) code (a Reed-Solomon code). The answer returned by the s^{th} server is constructed as follows.

$$Y_s=\sum_{l\in[\ell]}\frac{1}{\Delta_s^{,K_c}}\Omega(\widetilde{X^{(1)}}_{l}^{s},\widetilde{X^{(2)}}_{l}^{s},\cdots,\widetilde{X^{(N)}}_{l}^{s}). \tag{D.7}$$

Now let us see why it is possible to recover the desired evaluations from the answers of any $R=K_c(N+\ell-1)+N(X-1)+1$ servers. Note that Y_s can be rewritten as follows.

$$\begin{aligned}
Y_s&=\sum_{l\in[\ell]}\frac{1}{\Delta_s^{,K_c}}\Omega(\widetilde{X^{(1)}}_{l}^{s},\widetilde{X^{(2)}}_{l}^{s},\cdots,\widetilde{X^{(N)}}_{l}^{s}) &\text{(D.8)}\\
&=\sum_{l\in[\ell]}(\Delta_s^{,K_c})^{N-1}\Omega\left(\sum_{k\in[K_c]}\frac{1}{f_{l,k}-\alpha_s}x_{l,k}^{(1)}+\sum_{x\in[X]}\alpha_s^{x-1}z_{l,k,x}^{(1)},\cdots,\right.\\
&\qquad\left.,\cdots,\sum_{k\in[K_c]}\frac{1}{f_{l,k}-\alpha_s}x_{l,k}^{(N)}+\sum_{x\in[X]}\alpha_s^{x-1}z_{l,k,x}^{(N)}\right) &\text{(D.9)}\\
&=\sum_{l\in[\ell]}\sum_{k\in[K_c]}\frac{\prod_{k'\in[K_c]\backslash\{k\}}(f_{l,k'}-\alpha_s)^{N-1}}{(f_{l,k}-\alpha_s)}\Omega(x_{l,k}^{(1)},\cdots,x_{l,k}^{(N)})+\sum_{i\in[(K_c-1)(N-1)+NX]}\alpha_s^{i-1}I_i. &\text{(D.10)}
\end{aligned}$$

In (D.10), we rewrite (D.9) following the same argument that we used in Section 6.4.3. Note that $I_i,i\in[(K_c-1)(N-1)+NX]$ represent various linear combinations of $\Omega(\cdot)$, which can be found explicitly by expanding (D.9). Their exact forms are irrelevant, hence omitted for ease of exposition. Now we can see that the answers from any $R=K_c(N+\ell-1)+N(X-1)+2B+1$ servers, whose indices are denoted as $s_1,s_2,\cdots,s_R$, are coded according to the following $R\times(R-2B)$ generator matrix of an MDS($R-2B$,R) code.

$$\begin{bmatrix}
\frac{1}{f_{1,1}-\alpha_{s_1}} & \frac{1}{f_{1,2}-\alpha_{s_1}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_1}} & 1 & \alpha_{s_1} & \cdots & \alpha_{s_1}^{R-2B-L-1}\\
\frac{1}{f_{1,1}-\alpha_{s_2}} & \frac{1}{f_{1,2}-\alpha_{s_2}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_2}} & 1 & \alpha_{s_2} & \cdots & \alpha_{s_2}^{R-2B-L-1}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\frac{1}{f_{1,1}-\alpha_{s_R}} & \frac{1}{f_{1,2}-\alpha_{s_R}} & \cdots & \frac{1}{f_{\ell,K_c}-\alpha_{s_R}} & 1 & \alpha_{s_R} & \cdots & \alpha_{s_R}^{R-2B-L-1}
\end{bmatrix}. \tag{D.11}$$

Thus the user (decoder) can correct up to $(R-(R-2B))/2=B$ errors in the answers. Upon error correction, the user is able to recover desired evaluations, which appear along the dimensions spanned by the Cauchy part. This completes the proof of recovery threshold $R=K_c(N+\ell-1)+N(X-1)+2B+1$.

Remark D.1. *Because of the X-secure constraint, the systematic construction presented in Section 6.4.4 cannot be applied to N-CSA codes for XSBNCDBC.*

Remark D.2. *GCSA codes for coded distributed batch matrix multiplication presented in Section 6.5 can similarly be generalized to allow X-secure and B-Byzantine settings. Such a generalization is straightforward, thus omitted here.*