ABSTRACT

Title of dissertation: Private Information Read-Update-Write with Applications to Distributed Learning

Sajani Vithana, Doctor of Philosophy, 2023

Dissertation directed by: Professor Sennur Ulukus
Department of Electrical and Computer Engineering

Data privacy has gained significant interest in information theory with the emergence of a wide range of data-driven technologies in the recent past. Data privacy is compromised primarily when the data origins (private users) download or upload information. This dissertation focuses on how information-theoretic privacy of the users' data can be guaranteed when downloading and uploading information, along the lines of private information retrieval (PIR) and private read-update-write (PRUW), which have applications in privacy-preserved distributed learning.

First, we consider the problem of semantic PIR, in which multiple non-colluding databases store a number of files, out of which a user desires to download one without revealing its index to any of the databases. Semantic PIR deviates from classical PIR by allowing the files to have arbitrary semantics such as different file sizes and arbitrary popularity profiles. As the main result of this work, we characterize the capacity of semantic PIR, with achievable schemes and a converse proof. We provide capacity results for semantic PIR with replicated databases, MDS coded databases and colluded databases.

As PIR deals with private *reading*, we consider private *writing* in the second work, which is an immediate conceptual extension of PIR. In the problem of private read-update-write (PRUW), a user downloads, updates and writes back a specific section of a storage system while guaranteeing information-theoretic privacy of the index of the section updated and the values of the updates. PRUW has applications in distributed learning such as privacy-preserved federated learning (FL) with sparsification and private federated submodel learning (FSL). In FSL, a machine learning model is divided into multiple submodels based on different types of data used for training, and a given user only downloads and updates the submodel relevant to the user's local data. To guarantee the privacy of the users participating in the FSL process, both the updating submodel index and the values of the updates must be kept private from the server that stores the model. This is achieved by PRUW, where the required submodel is downloaded in the *reading phase* without revealing the submodel index to the databases (similar to PIR), and the updates are sent back to the databases in the *writing phase* without revealing the values of the updates or the submodel index. We provide a basic PRUW scheme to perform private FSL that achieves the lowest known total communication cost of private FSL thus far. In this work, we introduce the concept of combining multiple parameter updates into a single bit in terms of a Lagrange polynomial in such a way that it can be privately decomposed into the respective individual updates and added to the relevant positions at the server.

Third, we consider the problem of private FSL with top r sparsification, in which the user-server communications are significantly reduced by only sharing the

most significant r fractions of parameters and updates in the reading and writing phases. However, this introduces additional privacy requirements as the positions of the sparse updates/parameters leak information about the users' private data in addition to their values and the submodel index. To this end, we provide a PRUW scheme that performs top r sparsification in FSL while guaranteeing the information-theoretic privacy of the updating submodel index, values of the sparse updates and the positions of the sparse updates/parameters using a permutation technique.

Fourth, we study random sparsification in FSL, in which the user only downloads and uploads a specific fraction of randomly selected parameters and updates to reduce the communications. The problem is formulated in terms of a rate-distortion characterization, where we derive the minimum achievable communication cost for a given amount of allowed distortion. We show that a linear rate-distortion relation is achievable while guaranteeing the information-theoretic privacy of the updating submodel index, the values of the sparse updates and the positions of the sparse updates/parameters.

Fifth, we extend the ideas of PRUW to FL with top r sparsification. While the same permutation technique introduced in FSL with top r sparsification is applicable to this setting, it incurs a significantly large storage cost for FL. To alleviate this, we modify the permutation technique in such a way that the storage cost is reduced at the expense of a certain amount of information leakage, using a model segmentation mechanism. In general, we provide the trade-off between the communication cost, storage cost and information leakage in private FL with top r sparsification, along

with achievable schemes with different properties.

In all of the above PRUW settings, we require multiple non-colluding databases to store the central model to guarantee information-theoretic privacy of the users' local data. In the sixth work, we consider the practical scenario of private FSL, where the databases storing the central model are allowed to have arbitrary storage constraints. As the main result of this work, we develop a PRUW scheme and a storage mechanism for FSL that efficiently utilize the available space in each database to store the submodel parameters in such a way that the total communication cost is minimized while guaranteeing information-theoretic privacy of the updating submodel index and the values of the updates. The proposed storage mechanism is a hybrid of MDS coded storage and divided storage, which focuses on finding the optimum MDS codes and fractions of submodels stored in each database for any given set of homogeneous or heterogeneous storage constraints.

Seventh, we go beyond privacy and consider deception in information retrieval. We introduce the problem of deceptive information retrieval (DIR) which is a conceptual extension of PIR. In DIR, the user downloads a required file out of multiple files stored in a system of databases and reveals information about a different file as what was required, to the databases. In other words, the user deceives the databases by making their prediction on the user-required file index incorrect with high probability. We propose a scheme to perform DIR that achieves a given required level of deception, with the goal of minimizing the download cost. The proposed scheme incurs higher download costs compared to PIR for positive levels of deception, and achieves the PIR capacity when the level of deception specified is zero.

Private Information Read-Update-Write with Applications to Distributed Learning

by

Sajani Vithana

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
Professor Sennur Ulukus, Chair/Advisor
Professor Behtash Babadi
Professor Alexander Barg
Professor Adrian Papamarcou
Professor Lawrence C. Washington

Dedication

To my family.

Acknowledgments

I am extremely grateful to my advisor, Professor Sennur Ulukus for her continuous guidance, support, and valuable advice in every step of my PhD journey. Her dedication and passion for research have always inspired me. Five years of interactive research discussions and conversations with her have taught me how to develop meaningful and technically sound engineering problems and opened me up to many interesting research directions. I also learned a lot from her profound knowledge and intuitive approach to analyzing problems. Her optimism, encouragement, and kind words have always lifted my spirit and motivated me during hard times. My PhD journey with Professor Ulukus has been an amazing learning experience, and I consider myself very lucky to be one of her students.

I would like to thank my committee members, Professors Behtash Babadi, Alexander Barg, Adrian Papamarcou and Lawrence C. Washington for their time and valuable feedback on my research. I am also thankful to all the professors that have I interacted with over the past five years at UMD. Specifically, I want to thank Professors Prakash Narayan, Richard La, Eric Slud and Dmitry Dolgopyat for teaching interesting courses from which I learned many important concepts.

I am very fortunate to have spent the past five years with an amazing group of colleagues in our research group: Priyanka Kaswan, Matin Mortaheb, Cemil Vahapoglu, Purbesh Mitra, Mustafa Doger, Subhankar Banerjee, Shreya Meel, Sahan Liyanaarachchi, Mohamed Nomeir, Arunabh Srivastava, Alptug Aytekin, Zhusheng Wang, Batuhan Arasli, Brian Kim, Baturalp Buyukates and Melih Bastopcu. I

have always enjoyed the intriguing research discussions with Zhusheng, Mohamed, Alptug and Shreya. Special thanks to Priyanka for the last-minute homework discussions, adventures at conferences and for all the exciting conversations on research and life. My sincere gratitude goes to Karim Banawan for his collaboration in my first project, from which I learned a lot.

I am grateful to all the new and old friends I met at UMD, including Anuththara, Lasitha, Yashish, Dushyanthi, Shane, Umesha, Ruwanthi and Vinoj for making UMD feel like home, and for the great adventures we had, hiking and camping in national parks across the country. I also want to thank Paulo and Lauren for being great friends and for the good times we had, playing intramural sports and countless board games. The list goes on, and I am very thankful for all my friends who always made me laugh and kept my mental and physical health in check.

My deep and sincere gratitude goes to my parents Badra Hewavithana and Ranjith Vithana, for all the sacrifices they have made along the way, for their words of wisdom and for always believing in me. Right from the start, they encouraged and supported me to follow my dreams and passions and were always there to help me bounce back whenever I failed. I also want to thank my brother Sanura for his weird jokes, words of encouragement, and for always being there for me.

Last but not least, I want to thank my loving husband Dilhara, who has been my rock throughout the PhD journey. As a fellow PhD student, he perfectly understood my aspirations, dreams and goals, and always encouraged and supported me to bring out the best in me. His positive, cheerful, and enthusiastic mindset inspires me every day to find joy in everything I do.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Data privacy has gained significant interest among the researchers in computer science and information theory over the years due to the increasing dependency on private and sensitive data in a wide range of emerging technologies. For example, artificial intelligence (AI) and machine learning (ML), which are powered by users' private data, have become the driving force in many of the essential applications in the modern world. As much as the AI-based applications such as navigation and fraud detection systems have become indispensable to the users, it is the responsibility of the technology to ensure the privacy of the data origins using which the learning processes are powered-up, due to ethical and legal reasons, as well as to improve the sustainability and reliability of the services. While data privacy has been an active area of research in computer science for quite a long time with computational privacy guarantees, it has gained significant recent interest in information theory with *perfect* privacy guarantees that ensure zero information leakage. Typically, the information of a private user is leaked when the user downloads (reads), or uploads (writes) information from/to another party. This dissertation discusses

specific cases of information retrieval and transmission, and provides methods to perform them while guaranteeing information-theoretic privacy of the users' private data, along with the corresponding fundamental performance limits.

In Chapter 2, we focus on guaranteeing information-theoretic privacy of a user who only reads, i.e., downloads, information from a data storage system. In particular, we study the problem of private information retrieval (PIR), introduced in the seminal paper [1], in which a user retrieves a message (file), out of several messages stored in multiple replicated and non-colluding databases, without revealing any information about the identity of the desired message. Recently, this problem has attracted significant attention in information theory where the fundamental limits of the problem based on absolute guarantees (in contrast to computational guarantees as in [2]) have been investigated. In [3], the notion of PIR capacity is introduced as the maximum ratio of the desired message size to the total download size. Since then, many variants of the classical PIR problem have been studied. References [4–8] study the setting with colluding databases, [9–11] investigate the problem with coded databases and [12–15] consider the setting with databases that are both coded and colluded. PIR with storage constrained databases and Byzantine databases are studied in [16–24] and [25–27], respectively. Multi-message PIR, where a user requires to download multiple message more efficiently, compared to the naive method of downloading one message at a time is studied in [28, 29]. Another interesting variant is symmetric PIR, where the privacy of databases is also taken into account on top of user's privacy. Here, the users gain no information about other messages, except for the one that is required. This is studied in [30–34].

Methods of reducing the download cost of PIR with the aid of side information, caches, and privacy leakage is studied in [35–42], [43–47] and [48–50], respectively.

In all these works, two assumptions are made: All messages have the same size[1], L, and all messages are requested uniformly by the users. These assumptions are highly idealistic from a practical point of view. For instance, consider a streaming example where the storage database has a catalog of different movies and TV shows. These media files cannot be assumed to have the same level of popularity or the same size. Consequently, each message stored in the databases exhibits different *semantics*, in the sense that each message has a different size and a different prior probability of retrieval. With this backdrop, we introduce the problem of semantic PIR, in which we investigate how a PIR scheme should be implemented over databases holding messages with different semantics. The retrieval rate is defined to be the ratio of the *expected* message size to the *expected* download cost. Due to the privacy constraint, the download cost needs to be the same for all messages; thus, the expected download cost is equal to the download cost for each individual message. Hence, the retrieval rate achieved by a given scheme is equal to the weighted average of all individual message retrieval rates. We investigate the semantic PIR capacity as a function of the system parameters: number of databases N, number of messages K, message priors p_i, and message lengths L_i. We ask how semantic PIR capacity compares to classical PIR capacity, and whether there is a PIR capacity

[1]With the exception of [30], which characterizes the capacity of the symmetric PIR (SPIR) problem for heterogeneous file sizes (without considering prior probabilities of retrieval) to be $R_k = \frac{L_k}{\max_i L_i}\left(1-\frac{1}{N}\right)$, where R_k is the rate of retrieving message k. The achievable scheme follows by dividing the files into partitions of length $N-1$ and repeating the original SPIR scheme in each partition. This scheme zero-pads shorter messages so that their lengths are equal to that of the longest message.

gain from exploiting the message semantics.

As the main result of this work, we characterize the exact semantic PIR capacity for arbitrary parameters. To that end, we present two achievable schemes; the first scheme is deterministic, in the sense that the *query structure* is fixed, and the second scheme is stochastic, in the sense that the user picks a query structure *randomly* from a list of possible structures. For the deterministic scheme, we present a systematic method to determine the subpacketization level for each message. Note that this is crucial in our semantic problem due to the heterogeneous message sizes, unlike the majority of the literature that utilizes uniform subpacketization within their schemes [51]. This scheme uses non-uniform subpacketization where the block size considered in each download differs from one message to another. The query structure of the deterministic scheme resembles the query structure of [3], in that, our scheme uses the same k-sums idea of [3]. The second achievable scheme is composed of several query options that the user may use with equal probability to retrieve any message. In this scheme, the messages are divided into several blocks depending on the number of databases. The message is retrieved using a single set of queries, which is chosen uniformly randomly from the query options to ensure privacy. This is similar to the scheme presented in [49] with an extension to more than two databases (see also [52]). We provide a matching converse that takes into account the heterogeneity of message sizes, resulting in settling the semantic PIR capacity. Additionally, we provide two extensions of the semantic PIR problem, namely, semantic PIR from MDS-coded databases, where each database stores individual bits of a (N, K) MDS code, and semantic PIR from colluding databases,

where T out of the N databases are allowed to collude. For both extensions, we derive the exact PIR capacities by providing corresponding optimal schemes and converse results.

The capacity of semantic PIR given by $C = \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_2}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\mathbb{E}[L]}\right)^{-1}$ is a function of the message sizes, the a priori probability distribution, the number of databases and the number of messages. The expression implies that for certain message sizes and priors, the classical PIR capacity may be exceeded by exploiting the semantics of the messages even if the zero-padding needed in classical PIR to equalize the message sizes is ignored. Concretely, our results imply: 1) When message lengths are the same, semantic PIR capacity is equal to the classical PIR capacity no matter what the message priors are, i.e., priors cannot be exploited to increase the PIR capacity if the message lengths are the same. 2) For certain cases, such as when the prior probability distribution favors longer files (i.e., longer files are more popular), the semantic PIR capacity exceeds the classical PIR capacity which depends only on the number of databases and the number of messages.[2] 3) For all priors and lengths, semantic PIR achieves larger rates compared to what classical PIR would achieve by simply zero-padding the messages to bring them to the same length, as it assumes.

In Chapters 3-7 we present different variants of the basic the problem of private read-update-write (PRUW), in which we investigate how information-theoretic privacy can be guaranteed when a user *reads* (downloads) and *writes* (uploads)

[2]By classical PIR capacity, we mean the classical PIR capacity expression, which may not be attainable for heterogeneous file sizes in practice.

information to a data storage system. Note that *private writing* is an immediate conceptual extension of *private reading* present in PIR, which has a number of applications in privacy-preserving distributed learning. To motivate PRUW we address two issues present in distributed learning, namely, the information leakage of participating users and the significantly large communication cost. In distributed learning techniques such as federated learning (FL) [53–56], millions of users collectively train a ML model stored at a central server by downloading the model, updating it with their local data and sending the updates back to the server. Even though the user's local data is not directly shared with the server, it has been shown that the updates sent to the server leak information about the user's local data in FL [57–63]. Different methods have been developed to minimize this information leakage such as classical cryptographic protocols as in secure aggregation [64], differential privacy (DP) [65] via noise addition, data sampling and data shuffling, e.g., [66–78]. However, these methods do not guarantee *perfect* privacy of each individual user's local data. In Chapters 3-7, we introduce new techniques via PRUW, that can be used to guarantee information-theoretic privacy of the users participating in the FL process.

Apart from the privacy concerns, distributed learning techniques such as FL incur significantly large communication costs since many users communicate parameters and updates with the server in multiple rounds. Moreover, FL requires all users to download and update the entire model even when the users do not have all types of data required to update the entire model. Gradient sparsification [79–86] , gradient quantization [87–90] and federated submodel learning (FSL) [91–95] are among the solutions proposed in the literature to mitigate these inefficiencies. In gradient

sparsification, the users only communicate a selected set of gradients (most significant/randomly chosen) as opposed to sending all gradient updates corresponding to all parameters in the model to the central server. In gradient quantization, the values of the gradients are quantized and represented using a fewer number of bits. In FSL, the machine learning model is divided into multiple submodels based on different types of data used to train it, and each user only downloads and updates the submodel that can be updated by the user's own local data. This saves the communication cost and makes the distributed learning process more efficient.

Although FSL and gradient sparsification are efficient in terms of the communication cost, they leak more information on the user's local data to the servers. In FSL, the index of the updating submodel clearly depends on the types of data that the users have. Therefore, in FSL both the updating submodel index as well as the values of the updates need to be kept private from the server in order to guarantee the privacy of each user's local data. In FL with gradient sparsification, when the sparse gradients are chosen based on their significance, the indices/positions of these selected gradients leak information about the user's private data, in addition to the information leaked by the values of the gradients. Hence, we need to hide both the values and the indices of the sparse gradient updates from the server in FL with sparsification to ensure user-privacy. All of the above privacy requirements can be fulfilled with the concepts introduced in PRUW. Formally, PRUW refers to the general process of privately reading a required segment of a data storage system and writing back to the same/different segment privately, without revealing the segment indices or the values written, to the storage system. Private FSL and private FL

with sparsification are both applications of PRUW.

In Chapter 3, we investigate the problem of private FSL along the lines of PRUW. Private FSL consists of two phases in terms of the communications. 1) Reading phase in which the user downloads the required submodel from the storage system without revealing its index. 2) Writing phase in which the user uploads the updates back to the same submodel without revealing the submodel index or the values of updates. Note that the reading phase is similar to PIR. Existing works on private FSL [91–96] provide schemes with different notions of privacy. References [91, 93] consider locally differential privacy, in which a predetermined amount of information of the user is leaked to the databases. Reference [78] presents a group-wise aggregation scheme (related to the writing phase in FSL) based on local differential privacy. The schemes in [92] and [95] consider information-theoretic privacy of the submodel index and the values of updates. However, they are less efficient in terms of the communication cost, compared to the schemes presented in [94] and Chapters 3-7 that are based on cross subspace alignment (CSA) [97] for different variants of FSL that consider information-theoretic privacy. Reference [94] presents a private FSL scheme for the general case with arbitrary number of colluding databases with the presence of dropouts. An efficient private FSL scheme that is over-designed with extra noise terms to reduce the communication cost is presented in Chapter 3, along with its extensions to storage constrained databases in Chapter 7 and further means of reducing the communication cost with sparsification in Chapters 4 and 5.

The system model we consider for private FSL consists of N non-colluding

databases storing M independent submodels that contain elements from a finite field $\mathbb{F}_q$. We consider information-theoretic privacy of the values of updates and the index of the submodel updated by the user, and define the reading and writing costs to be the normalized total downloads and uploads, respectively. The privacy constraints are guaranteed by adding random noise to quantities that need to be kept private. This is a direct application of Shannon's one time pad theorem which states that when X is a random variable with an arbitrary distribution and Y is a uniformly distributed random variable (random noise), both within a finite field, $X+Y$ is uniformly distributed and is independent of X. In Chapter 3, we improved the scheme in [95] by reducing the writing cost from $\approx \frac{N}{2}$ to $\frac{2}{1-\frac{2}{N}}$ for the case with N non-colluding databases. The improved scheme is able to perform FSL by only downloading and uploading approximately twice as many bits as the size of a submodel when N is large, while guaranteeing information-theoretic privacy of the updating submodel index and the values of the updates. This is achieved by introducing the concept over-designing the storage with extra noise terms to aid the process of combining multiple parameter updates into a single bit (by users in the writing phase) in the form of a modified Lagrange polynomial, such that it can be privately decomposed into the respective individual updates and added to the relevant positions at the databases. The basic PRUW scheme introduced in Chapter 3 is one of the main building blocks of the schemes presented for different variants of PRUW described in Chapters 4-6.

In Chapters 4-5, we investigate further means of reducing the communication cost in private FSL. In basic private FSL described in Chapter 3, it is assumed that

each individual user downloads and updates all parameters of the required submodel. However, the communication cost can be reduced further by only downloading and updating a selected set of parameters within the submodel. This is in fact gradient sparsification in FSL, i.e., the combination of the two main applications of PRUW. In Chapters 4 and 5, we consider two types of sparsification in FSL, namely, top r and random sparsification. In top r sparsification, only a given fraction of the most significant parameters/updates are downloaded and uploaded in the reading and writing phases. In random sparsification, a random set of parameters is read in the reading phase, and the same/different random set of parameters is updated in the writing phase. A given amount of distortion is introduced in both sparsification methods, which in general has little or no impact on the accuracy of the model. In fact, sparsification is a widely used technique in most learning tasks to reduce the communication cost, which even performs better than the non-sparse models in certain cases [80, 86, 98].

In private FSL with top r sparsification, each user only uploads the most significant r fraction of updates of the updating submodel in the writing phase, and downloads only a r' fraction of parameters in the reading phase[3]. This ensures that the most significant gradient variations in the training process are communicated while incurring significantly reduced communication costs compared to non-sparse training. In general, top r sparsification requires users and the server to communicate the updates/parameters as well as their positions in order to update the

[3]The r' fraction of parameters can be the union of the sparse sets of parameters updated by all users in the previous iteration, or they can be chosen in a specific way as in [80].

model correctly. Note that even if the values of the sparse updates/parameters are privacy-protected as in non-sparse private FSL described in Chapter 3, specifying the positions leak information of the user's local data. For example, in the writing phase, the databases learn that the updates corresponding to the non-specified positions are less significant compared to the ones that are specified. Therefore, in private FSL with top r sparsification, three components need to be kept private, 1) updating submodel index, 2) values of sparse updates 3) positions of sparse updates. In Chapter 4, we provide a scheme that performs private FSL with top r sparsification while guaranteeing information-theoretic privacy of the three components stated above. Privacy of components 1 and 2 are satisfied by random noise addition, i.e., one time padding, and for the third component, we introduce an update/parameter shuffling mechanism, in which the true positions are permuted and transmitted in a specific way such that the permutations can be reversed at the receiving end, while guaranteeing information-theoretic privacy of the true positions. The proposed scheme significantly reduces the reading and writing costs of private FSL from $\frac{2}{1-\frac{2}{N}}$ (without sparsification) to $\approx \frac{2r}{1-\frac{4}{N}}$, where r is the sparsification rate, which is in general around $\approx 10^{-2}$ to $\approx 10^{-3}$.

Essentially, private FSL with top r sparsification discussed in Chapter 4 is an extension of basic private FSL presented in Chapter 3, with the additional challenge of guaranteeing the information-theoretic privacy of the positions of the selected sets of updates/parameters resulting from top r sparsification. The main contributions of this work are as follows; 1) introduction of the concept and the system model for top r sparsification in PRUW in relation to private FSL, 2) scheme that performs private

FSL with top r sparsification while guaranteeing information-theoretic privacy of the updating submodel index, values of the sparse updates, and the positions of the sparse updaes, 3) introduction of a permutation/shuffling mechanism to privately convey information about the indices of a set of selected elements of a vector in a distributed setting.

In Chapter 5, we investigate random sparsification in private FSL. Here, the user only downloads and uploads a randomly selected set of parameters and updates of the required submodel, without revealing the submodel index, the values of the sparse updates and the indices of the sparse parameters/updates. The unread (unwritten) parameters (updates) result in a certain amount of distortion in the reading (writing) phase. Therefore, we formulate this problem in terms of a rate-distortion characterization in PRUW, where the goal is to minimize the total communication cost for a given amount of allowed distortion. The distortion is defined as the Hamming distance between the actual and downloaded/uploaded data. We consider N non-colluding databases storing M independent submodels, with given reading and writing distortion budgets denoted by $\tilde{D}_r$ and $\tilde{D}_w$, respectively. In the proposed scheme, based on the allowed distortions in the two phases, we find the optimum subpacket[4] sizes, and download and upload only a single bit per subpacket, per database in the reading and writing phases, respectively. These single bits are combinations of randomly chosen subsets of parameters/updates within the respecitve subpakcet, and are structured in a specific way such that they can be

[4]A subpacket is a block of bits of all submodels stored in databases, on which the proposed scheme is defined. The scheme is repeatedly applied on all such subpackets in the entire storage.

privately decomposed into the sparse individual parameters/updates at the receiving end. The queries and storage in the proposed scheme are designed such that they are compatible with the sparse update decomposition process. Moreover, the storage structure in databases is carefully designed to accommodate the possible differences between the subpacket sizes in the reading and writing phases. We show that a linear rate-distortion trade-off in private FSL is achievable, i.e., a reading (writing) cost of $\approx \frac{2}{1-\frac{2}{N}}(1-\tilde{D})$ is achievable where $\tilde{D}$ is the allowed amount of distortion in the reading (writing) phase.

Next, we extend the ideas of PRUW in FSL with top r sparsification to FL in Chapter 6. As explained before, top r sparsification is a widely used sparsification technique in FL, where only the most significant r fraction of parameters/updates are shared between the users and the central server to reduce the communications in the FL process. However, the values as well as the positions (indices) of the sparse updates leak information about the user's local data. The positions of the sparse updates leak information about the most and least significant sets of parameters for a given user, which can be used to infer information about the user's private data. Thus, in order to guarantee the privacy of users participating in the sparse FL process, two components need to be kept private, namely, 1) values of sparse updates, 2) indices of sparse updates. In this chapter, we develop schemes to perform the user-database communications in FL with top r sparsification while guaranteeing information-theoretic privacy of the values and the indices of the sparse updates.

The system model consists of multiple non-colluding databases storing the FL model, and a single user that communicates with the databases in the training

process. The proposed schemes are based on a permutation technique, in which a coordinator initializes a random permutation of sets of parameters, and sends it to the users. The coordinator then places noise added permutation reversing matrices at each database in such a way that the databases learn nothing about the underlying permutation. All communications between the user and the databases take place in terms of the permuted indices, which guarantees the privacy of the indices of the sparse updates. However, the parameters in each database get updated in the correct order, with the aid of the noise added permutation reversing matrices. The main drawback of this method is the considerably large storage cost incurred by the large permutation reversing matrices. To that end, we propose schemes that reduce the storage cost by decreasing the size of the noise added permutation reversing matrices, at the expense of a given amount of information leakage. This is achieved by dividing the ML model into multiple segments and carrying out permutations within each segment. The number of segments is chosen based on the allowed amount of information leakage and the storage capacity of the databases.

We present four schemes to perform user-database communications in private FL with top r sparsification with different properties such as lower communication costs, lower storage costs or lower amounts of information leakage. The four schemes differ from each other based on the storage structure (MDS coded or uncoded) and the permutation mechanism (only within-segment permutations or within and inter-segment permutations) used. MDS coded storage decreases the storage cost while increasing the communication cost, and the two-stage permutations (within and inter-segment permutations) decrease the information leakage

significantly compared to single-stage permutations (only within-segment permutations), while slightly increasing the communication cost. Based on the specifications and limitations of the given FL task, one can choose the most suitable scheme. In general, we present the trade-off between the communication cost, storage complexity and information leakage in private FL with sparsification.

In Chapeter 7, we consider the problem of PRUW with storage constrained databases, focusing on the application of private FSL. This problem is motivated by the fact that it requires multiple non-colluding databases to store the submodels to guarantee information-theoretic privacy of the user-required submodel index and the values of the updates in FSL. In practice, these non-colluding databases may have arbitrary storage constraints, which requires a flexible PRUW scheme that efficiently utilizes the available storage space in all databases to achieve the minimum possible communication cost in FSL. The main goal of this work is to determine storage mechanisms and compatible PRUW schemes that are applicable to any given set of arbitrary storage constraints.

This work is closely related to the work presented in [9,17–20,24,42] in the PIR literature on storage constrained databases. In this work, we extend these ideas to PRUW with the goal of minimizing the total communication cost while guaranteeing the additional privacy and security requirements present in PRUW. Divided storage and coded storage are the two main approaches to storing data in databases with storage constraints. In divided storage, the data is divided into multiple segments, and each segment is only replicated in a subset of databases. In coded storage, multiple data points are combined into a single symbol using specific encoding structures,

and stored at all databases. The concept of combining coded storage and divided storage to meet *homogeneous* storage constraints in *PIR* was introduced in [20]. It is shown that this combination results in better PIR rates compared to what is achieved by coded storage and divided storage individually. In this work, we explore such ideas for both *homogeneous* and *heterogeneous* storage in *PRUW* in the context of a private FSL problem. We propose a hybrid storage mechanism for private FSL that uses both divided and coded storage to store the submodel parameters, followed by a compatible PRUW scheme that achieves the minimum total communication cost within the algorithm for any given set of storage constraints, while guaranteeing the information-theoretic privacy of the user-required submodel index and the values of the updates.

We consider both heterogeneous and homogeneous storage constraints. The former corresponds to the case where different databases have different storage constraints, while the latter corresponds to the case with the same storage constraint across all databases. We provide two different schemes for the two settings, as they are rooted differently to increase the communication/storage efficiency of each case separately. Both schemes are composed of two main steps, namely, the storage mechanism, which assigns the content stored in each database, and the PRUW scheme, which performs the read-write process on the stored data. The PRUW scheme is based on the scheme presented in Chapter 3, with the parameters optimized to achieve the minimum total communication cost, when the submodel parameters are (K, R) MDS coded. In the heterogeneous case, the basic idea of the storage mechanism is to find the optimum (K, R) MDS codes and the corresponding fractions of

submodels stored using them. The general scheme that we propose in this work is applicable to any given set of storage constraints, and is based on a PRUW scheme that achieves lower and higher communication costs when the data is (K, R) MDS coded with odd and even values of $R - K$, respectively. To this end, we show that the use of (K, R) MDS codes with even $R - K$ can be avoided, and the total communication cost can be reduced if the given storage constraints satisfy a certain set of conditions. The class of homogeneous storage constraints satisfies this set of conditions. Hence, we discuss the case of homogeneous storage constraints separately in detail and propose a different storage mechanism that is more efficient compared to the general scheme designed for heterogeneous storage constraints.

In Chapter 8, we go beyond privacy, and consider *deception* in information retrieval. We introduce the problem of deceptive information retrieval (DIR) in this chapter, which is a conceptual extension of PIR. In PIR, the databases' prediction of the user-required file based on the received queries is uniformly distributed across all files. Hence, the probability of error of the database's predictions in a PIR setting with K files is $1 - \frac{1}{K}$. In weakly private information retrieval [48, 99], a certain amount of information on the user-required file index is revealed to the databases to reduce the download cost. In such cases, as the databases have more information on the file index that the user requests, the error probability of the database's prediction is less than $1 - \frac{1}{K}$. In this work, we study the case where the error probability of databases' prediction is larger than $1 - \frac{1}{K}$.

Note that with no information received by the user at all, the databases can make a random guess on the user-required file index, and reach an error probability

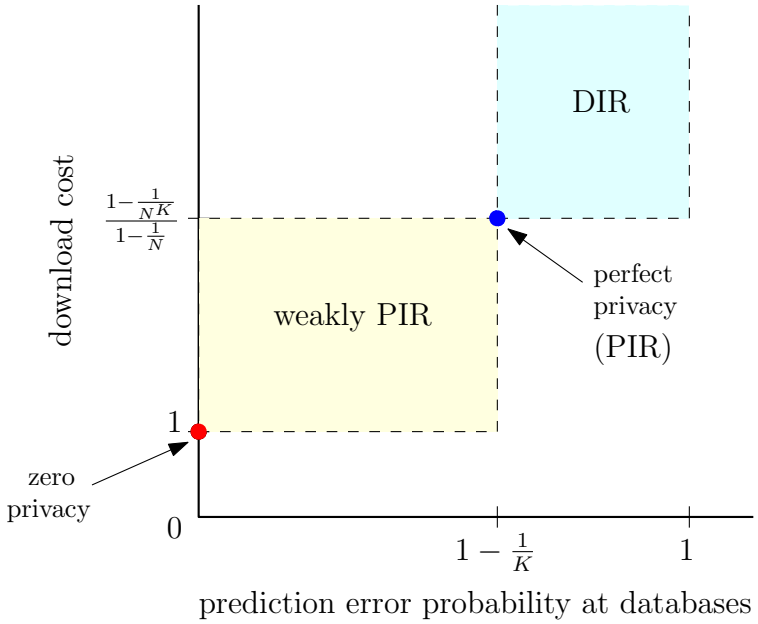


Figure 1.1: Download costs and prediction error probabilities for different types of information retrieval.

of $1-\frac{1}{K}$. Therefore, to result in a prediction error that is larger than $1-\frac{1}{K}$, the user has to *deceive* the databases by sending fake information on the required file index. The goal of this work is to generate a scheme that allows a user to download a required file k, while forcing the databases' prediction on the user-required file index to be ℓ, where $k \neq \ell$, for as many cases as possible. This is coined as deceptive information retrieval (DIR). DIR is achieved by sending *dummy* queries to databases to manipulate the probabilities of sending each query for each file requirement, which results in incorrect predictions at the databases. However, sending dummy queries increases the download cost compared to PIR. Fig. 1.1 shows the behavior of the prediction error probability and the corresponding download costs for different types of information retrieval.[5]

[5]The regions marked as "weakly PIR" and "DIR" in Fig. 1.1 show the points that are conceptually valid for the two cases and does not imply that every point in those regions are achievable. The achievable points corresponding to "weakly PIR" and "DIR" lie within the marked regions.

The concept of deception has been studied as a tool for cyber defense [100–104], where the servers deceive attackers, adversaries and eavesdroppers to eliminate any harmful activities. In all such cases, the deceiver (servers in this case), gains nothing from the deceived, i.e., attackers, adversaries and eavesdroppers. In contrast, the main challenge in DIR is that what needs to be deceived is the same source of information that the user retrieves the required data from. This limits the freedom that a DIR scheme could employ to deceive the databases. To this end, we formulate the problem of DIR based on the key concepts used in PIR, while also incorporating a *time dimension* to aid deception.

The problem of DIR introduced in this chapter considers a system of non-colluding databases storing K independent files that are time-sensitive, i.e., files that keep updating from time to time. We assume that the databases only store the latest version of the files. A given user wants to download arbitrary files at arbitrary time instances. The correctness condition ensures that the user receives the required file, right at the time of the requirement, while the condition for deception requires the databases' prediction on the user-required file to be incorrect with a probability that is greater than $1-\frac{1}{K}$, specified by the predetermined level of deception required in the system.

The scheme that we propose for DIR deceives the databases by sending *dummy* queries to the databases for each file requirement, at distinct time instances. From the user's perspective, each query is designed to play two roles as *real* and *dummy* queries, with two different probability distributions. This allows the user to manipulate the overall probability of sending each query for each message requirement,

which is known by the databases. The databases make predictions based on the received queries and the globally known probability distribution of the queries used for each file requirement. These predictions are incorrect with probability $> 1 - \frac{1}{K}$ as the probability distributions based on which the real queries are sent are different from the globally known overall distribution. This is the basic idea used in the proposed scheme which allows a user to deceive the databases while also downloading the required file. The download cost of the proposed DIR scheme increases with the required level of deception d, and achieves the PIR capacity when $d = 0$.

The conclusions of this dissertation are provided in Chapter 9.

CHAPTER 2

Semantic Private Information Retrieval

2.1 Introduction

In this chapter, we consider the problem of semantic PIR. In semantic PIR, a user retrieves a message out of K independent messages stored in N replicated and non-colluding databases without revealing the identity of the desired message to any individual database. The messages come with *different semantics*, i.e., the messages are allowed to have *non-uniform a priori probabilities* denoted by $p_i > 0$, $i \in \{1, \ldots, K\}$, which are a proxy for their respective popularity of retrieval, and *arbitrary message sizes* L_i, $i \in \{1, \ldots, K\}$. This is a generalization of the classical PIR problem, where messages are assumed to have equal message sizes. We derive the semantic PIR capacity for general K, N, p_i and L_i, along with two achievable schemes and a converse proof. We also derive necessary and sufficient conditions for the semantic PIR capacity to exceed the classical PIR capacity with equal priors and sizes. Our results show that the semantic PIR outperforms classical PIR when longer messages are more popular. However, when messages are equal-length, the non-uniform priors cannot be exploited to improve the retrieval rate

over the classical PIR capacity. We provide two extensions of the semantic PIR problem, namely, semantic PIR from MDS-coded databases and semantic PIR from colluding databases. For both extensions, we derive exact PIR capacities in addition to providing the corresponding optimal schemes.

2.2 Problem Formulation

We consider a setting, where N non-colluding databases store K independent messages (files), $W_1, \ldots, W_K$, in a replicated fashion. The messages exhibit different semantics, i.e., the messages have different sizes and different a priori probabilities of retrieval. The a priori probability of W_i is denoted by[1] p_i, such that $p_i > 0$ for $i = 1, \ldots, K$. The a priori probability distribution is globally known at the databases and the user. We assume that all message symbols are picked from a finite field[2] $\mathbb{F}_s$. The message size of the ith message is denoted by L_i. Without loss of generality, we assume that the messages are ordered with respect to their sizes[3], such that $L_1 \geq L_2 \geq \cdots \geq L_K$. We assume that the messages stored in databases are mutually independent (which in turn implies pairwise independence). Hence, assuming that the message sizes are expressed in s-ary symbols,

$$H(W_i) = L_i, \quad i = 1, \ldots, K \tag{2.1}$$

[1]We assume that $p_i > 0$ for all $i \in [K]$ without loss of generality, as $p_j = 0$ for some j implies that this message, W_j, is either non-existent or never requested by the user. Hence, the setting can be reduced to a semantic PIR problem with $K - 1$ messages, each with $p_i > 0$.

[2]In this work, it suffices to work with the binary field, hence, symbols can be interpreted as bits.

[3]This is for ease of expression of the capacity formula in (2.9). The largest length should have the largest coefficient in the expression in (2.9) in order to have the largest achievable rate and the tightest converse.

$$H(W_1, \ldots, W_K) = \sum_{i=1}^{K} H(W_i) = \sum_{i=1}^{K} L_i \tag{2.2}$$

In semantic PIR, a user needs to retrieve a message W_i without revealing the index i to any individual database. To that end, the user sends a query to each database. The query sent to the nth database to retrieve W_i is denoted by $Q_n^{[i]}$ for $n = 1, \ldots, N$. Prior to retrieval, the user does not have any information about the message contents. Hence, queries sent to the databases to retrieve messages are independent of the messages, i.e., the mutual information between messages and queries is zero,

$$I(W_1, \ldots, W_K; Q_1^{[i]}, \ldots, Q_N^{[i]}) = 0, \quad i = 1, \ldots, K \tag{2.3}$$

Once the databases receive the queries, they generate answer strings to send back to the user. Specifically, the nth database prepares an answer string $A_n^{[i]}$ which is a deterministic function of the stored messages $W_1, \ldots, W_K$ and the received query $Q_n^{[i]}$. Therefore,

$$H(A_n^{[i]} | Q_n^{[i]}, W_1, \ldots, W_K) = 0, \; i = 1, \ldots, K, \; n = 1, \ldots, N \tag{2.4}$$

For a feasible PIR scheme, two conditions need to be satisfied, namely, the correctness and the privacy constraints. These are formally described as follows.

Correctness: The user should be able to perfectly retrieve the desired message as soon as the answer strings to the queries are received from the respective

databases. Therefore,

$$H(W_i|A_1^{[i]},\ldots,A_N^{[i]},Q_1^{[i]},\ldots,Q_N^{[i]})=0, \quad i=1,\ldots,K \tag{2.5}$$

Privacy: To protect the privacy of the desired message index i, the queries should not leak any information about i. Formally, for the nth database, the a posteriori probability of the message index i given a query $Q_n^{[i]}$ should be equal to the a priori probability of the message index i. That is, the random variable representing the desired message index, θ, should be independent of the received set of queries. Therefore,

$$P(\theta=i|Q_n^{[i]})=P(\theta=i),\ i=1,\ldots,K,\ n=1,\ldots,N \tag{2.6}$$

The privacy constraint (2.6) along with the independence of messages and queries (2.3) implies,

$$\begin{aligned}(Q_n^{[i]},A_n^{[i]},W_1,\ldots,W_K)\sim(Q_n^{[j]},A_n^{[j]},W_1,\ldots,W_K),\\ n=1,\ldots,N, \quad i,j=1,\ldots,K, \quad i\neq j\end{aligned} \tag{2.7}$$

An achievable semantic PIR scheme π is a scheme that satisfies the correctness constraint (2.5) and the privacy constraint (2.6) (or equivalently (2.7)). Due to the heterogeneity of message sizes and a priori probabilities, in this work, we define the performance metric, the expected retrieval rate $R(\pi)$ for any scheme $\pi\in\Pi$, where

Π is the set of all PIR schemes satisfying the correctness and privacy constraints given in (2.5) and (2.6), as the ratio of the expected retrieved message size to the expected download size, i.e.,

$$R(\pi) = \frac{\mathbb{E}[L]}{\mathbb{E}[D]}, \quad \pi \in \Pi \tag{2.8}$$

where $\mathbb{E}[L]$ is the expected number of useful bits downloaded and $\mathbb{E}[D]$ is the expected number of total bits downloaded. The expectation $\mathbb{E}[\cdot]$ in $\mathbb{E}[L]$ is with respect to the a priori probability distribution. Note that $\mathbb{E}[L]$ is fixed for any scheme as it is completely determined by the set of message lengths and prior probabilities which are given in the semantic PIR setting. The expectation $\mathbb{E}[\cdot]$ in $\mathbb{E}[D]$ is with respect to the distribution of the queries. Note that $\mathbb{E}[D]$ does not depend on the prior distribution as for any desired message, the download cost must remain the same to preserve privacy. Therefore, $\mathbb{E}[D]$ of a given scheme is completely determined by the structure of the scheme. The semantic PIR capacity is defined as the supremum of the expected retrieval rates over all achievable PIR schemes in Π, i.e., $C = \sup_{\pi \in \Pi} R(\pi)$. Moreover, the optimal semantic PIR scheme $\pi^* \in \Pi$ is an achievable scheme that minimizes the expected download cost, i.e., $\pi^* = \arg\min_{\pi \in \Pi} \mathbb{E}[D]$.

2.3 Main Results and Discussions

In this section, we present the main results of this work. Our first result is a complete characterization of the semantic PIR capacity. The semantic PIR capacity depends on the message sizes and prior probability distribution.

Theorem 2.1 *The semantic PIR capacity with N databases, K messages, message sizes L_i (arranged in decreasing order as $L_1 \geq L_2 \geq \cdots \geq L_K$), and prior probabilities p_i, is*

$$C = \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_2}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\mathbb{E}[L]} \right)^{-1} \tag{2.9}$$

$$= \left(\frac{L_1}{\sum_{i=1}^{K} p_i L_i} + \frac{1}{N}\frac{L_2}{\sum_{i=1}^{K} p_i L_i} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\sum_{i=1}^{K} p_i L_i} \right)^{-1} \tag{2.10}$$

where $\mathbb{E}[L] = \sum_{i=1}^{K} p_i L_i$.

The achievability proof of Theorem 2.1 is presented in Section 2.4 and the converse proof is presented in Section 2.5. Next, we have a few corollaries and remarks.

The following corollary gives a necessary and sufficient condition for the cases at which the semantic capacity exceeds the classical PIR capacity.

Corollary 2.1 (A Necessary and Sufficient Condition for Capacity Gain) *The semantic PIR capacity is strictly larger than the classical PIR capacity (with uniform priors and message sizes) if and only if,*

$$\sum_{i=1}^{K} \frac{1}{N^{i-1}} (L_i - \mathbb{E}[L]) < 0 \tag{2.11}$$

which is further equivalent to,

$$\sum_{i=1}^{K} \sum_{j=1}^{K} \frac{p_j}{N^{i-1}} (L_i - L_j) < 0 \tag{2.12}$$

Proof: The proof follows from comparing the semantic PIR capacity expression in (2.9) and the classical PIR capacity, C_{PIR}, in [3],

$$C_{PIR} = \left(1 + \frac{1}{N} + \cdots + \frac{1}{N^{K-1}}\right)^{-1} \tag{2.13}$$

Hence, $C > C_{PIR}$ implies

$$\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_2}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\mathbb{E}[L]} < 1 + \frac{1}{N} + \cdots + \frac{1}{N^{K-1}} \tag{2.14}$$

Ordering the terms leads to,

$$\sum_{i=1}^{K} \frac{1}{N^{i-1}}(L_i - \mathbb{E}[L]) < 0 \tag{2.15}$$

Noting $L_i = \sum_{j=1}^{K} p_j L_i$, since p_j sum to 1, and $\mathbb{E}[L] = \sum_{j=1}^{K} p_j L_j$ by definition of expectation,

$$\sum_{i=1}^{K}\sum_{j=1}^{K} \frac{p_j}{N^{i-1}}(L_i - L_j) < 0 \tag{2.16}$$

■

Remark 2.1 *The condition in* (2.11) *is a statement about the sum weighted (by* $\frac{1}{N^{i-1}}$*) deviation of message size from its expected value. Note that the expected value of the message size* $\mathbb{E}[L]$ *is a function of the message sizes* L_i *and the prior distribution* p_i *for* $i = 1, \ldots, K$*.*

Remark 2.2 *The intuition behind the condition in Corollary 2.1 is as follows. The set of message lengths and prior probabilities need to result in a large enough expected message length, which further implies that the longer messages need to be more popular, in order for the semantic PIR rate to outperform the classical PIR rate.*

Remark 2.3 *More explicit conditions can be derived for specific cases. For example, consider the case $K = 2$, $N = 2$, and assume that $L_1 > L_2$ (strictly larger). Then,* (2.11) *simplifies to,*

$$(L_1 - (p_1 L_1 + p_2 L_2)) + \frac{1}{2}(L_2 - (p_1 L_1 + p_2 L_2)) < 0 \tag{2.17}$$

$$p_2(L_1 - L_2) + \frac{1}{2} p_1 (L_2 - L_1) < 0 \tag{2.18}$$

$$p_2 - \frac{1}{2} p_1 < 0 \tag{2.19}$$

$$p_1 > \frac{2}{3} \tag{2.20}$$

where (2.19) *follows from $L_1 > L_2$. This means that for $N = 2$ and $K = 2$, the capacity of semantic PIR is greater than the capacity of classical PIR when the a priori probability of the longer message is greater than $\frac{2}{3}$ irrespective of the values of L_1 and L_2.*

As a further explicit example, if the more likely message is 4 times more likely and 4 times longer than the less likely message, i.e., if $p_1 = 4p_2$ and $L_1 = 4L_2$, then the semantic PIR capacity is $C = \frac{34}{45}$ while the classical PIR capacity is $C_{PIR} = \frac{2}{3} = \frac{30}{45}$. That is, for this case, $C_{PIR} = \frac{2}{3} < C = \frac{34}{45}$.

Remark 2.4 *We further expand on Remark 2.3 above by noting the following fact.*

The classical PIR capacity is a formula, as given in (2.13)*, that depends only on the number of databases N and the number of messages K, and is not necessarily achievable by the classical PIR scheme for any given message priors and lengths. To see this, we note that the classical PIR scheme requires equal message sizes. In the example in Remark 2.3 where $p_1 = 4p_2$ and $L_1 = 4L_2$, if we zero-pad the shorter message to make the message lengths the same, we achieve $R_{ach} = p_1\frac{L_1}{D} + p_2\frac{L_2}{D} = \frac{17}{30}$ by noting $D = \frac{3}{2}L_1$ as the length of the longer message is the common message length now, and the classical PIR capacity for this case is $\frac{2}{3}$. Thus, we observe $R_{ach} = \frac{17}{30} < C_{PIR} = \frac{2}{3} < C = \frac{34}{45}$ for this case.*

As a follow up to Remark 2.4, we note that the achievable scheme proposed in this work always outperforms zero-padding shorter messages and applying the classical PIR scheme for so-constructed equal-length messages. This is proved in the following corollary.

Corollary 2.2 *Semantic PIR capacity outperforms classical PIR rate with zero-padding.*

Proof: We first calculate the general achievable rate for the classical PIR scheme with zero-padding, R_{ach}. Noting $L_1 \geq L_2 \geq \cdots \geq L_K$, we zero-pad messages $2, \ldots, K$ until the message sizes are all equal to L_1. Next, we apply the classical PIR scheme with the common message size L_1. Then, the download cost (and the expected download cost) becomes,

$$\mathbb{E}[D] = D = \frac{L_1}{C_{PIR}} \tag{2.21}$$

Now, using C_{PIR} in (2.13) in equation (2.21) above, we obtain,

$$R_{ach} = \frac{\mathbb{E}[L]}{\mathbb{E}[D]} \tag{2.22}$$

$$= \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_1}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_1}{\mathbb{E}[L]} \right)^{-1} \tag{2.23}$$

Note repeated L_1 in the expression in (2.23). Comparing R_{ach} in (2.23) with the semantic PIR capacity in (2.9), we deduce that $R_{ach} \leq C$ as $L_1 \geq L_2 \geq \cdots \geq L_K$.

■

Remark 2.5 *If all messages have equal lengths, irrespective of the prior probabilities, the capacity of semantic PIR becomes equal to that of classical PIR. Note, in this case, $L_i = \mathbb{E}[L]$ and the capacity expression in* (2.9) *reduces to the classical PIR capacity expression in* (2.13)*. Thus, in order to exploit variability in priors to achieve a PIR capacity higher than the classical PIR capacity, we need variability in message lengths.*[4]

Remark 2.6 *Similar to classical PIR, the semantic PIR capacity increases with the number of databases, N. As the number of databases approaches infinity, the capacity approaches $\frac{\mathbb{E}[L]}{L_1}$. The reason why this asymptotic capacity is less than 1 is that the download cost must remain constant at L_1 (as the longest message achieves a rate of 1) irrespective of the desired message. The semantic PIR capacity decreases as the number of messages, K, increases. As K approaches infinity, the semantic*

[4]It is worth noting that classical PIR schemes need to be designed to satisfy the privacy constraint irrespective of the prior distribution. Nevertheless, the performance of the classical PIR schemes does not depend on the prior distribution as they consider uniform message sizes. This is in contrast to the semantic PIR problem, where the heterogeneity of the message sizes can be exploited to enhance the retrieval rate based on the properties of the prior distribution.

PIR capacity is lower bounded by

$$C > \frac{\mathbb{E}[L]}{L_1}\left(1 - \frac{1}{N}\right) \tag{2.24}$$

2.4 Achievability Proof

In this section, we present two PIR schemes that achieve the semantic PIR capacity given in Theorem 2.1. For each scheme, we first formally present the scheme, then we verify its correctness and privacy, calculate its achievable rate, and give explicit examples for illustration.

2.4.1 Achievable Semantic PIR Scheme 1

The scheme is based on the iterative structure of the achievable scheme in [3]. In this scheme, the user downloads k-sums from the messages for $k = 1, \ldots, K$. The novel component in our scheme is the calculation of the number of stages needed to be downloaded from each message based on the message sizes.

This achievable scheme is parameterized by $(K, N, \{L_i\}_{i=1}^{K})$. Based on these parameters, the user prepares queries to retrieve the desired message privately. The basic structure of our achievable scheme is as follows.

1. **Message indexing**: Order the messages in the descending order of message sizes. That is, index 1 is assigned to the longest message and index K is assigned to the shortest message ($L_1 \geq L_2 \geq \cdots \geq L_K$). Calculate

retrieval parameters[5] $\upsilon_1, \upsilon_2, \ldots, \upsilon_K$ corresponding to each message such that $\upsilon_1 \geq \upsilon_2 \geq \cdots \geq \upsilon_K$. The retrieval parameters denote the number of stages that needs to be downloaded from each message. The explicit expressions for these parameters are as follows:

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \\ \vdots \\ \upsilon_K \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \frac{1}{N} & -\frac{N-1}{N^2} & -\frac{N-1}{N^3} & \ldots & -\frac{N-1}{N^K} \\ 0 & \frac{1}{N^2} & -\frac{N-1}{N^3} & \ldots & -\frac{N-1}{N^K} \\ 0 & 0 & \frac{1}{N^3} & \ldots & -\frac{N-1}{N^K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{1}{N^K} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ \vdots \\ L_K \end{bmatrix} \tag{2.25}$$

where α can be chosen as the gcd of the vector elements resulting from the matrix multiplication in the right side of (2.25). This choice will become clear in Section 2.4.1.1.

For the rest of this section, assume that the user wishes to download W_j.

2. **Index preparation:** The user permutes the indices of all messages independently, uniformly, and privately from the databases. I.e., if the number of elements in a subpacket of W_i is ℓ_i, let W_i be denoted by, $W_i = (W_i(1), \ldots, W_i(\ell_i))$ for $i \in \{1, \ldots, K\}$. For each message W_i, the user uniformly and randomly chooses a permutation of the ℓ_i indices out of the $\ell_i!$ options, indicated by $(\gamma_i(1), \gamma_i(2), \ldots, \gamma_i(\ell_i))$, which is independent of all other

[5]This set of parameters determines the nonuniform subpacketization of a given semantic PIR setting with arbitrary message lengths. It also controls the numbers of stages in the next steps of the scheme (numbers of ℓ-sums, $\ell \in \{1, \ldots, K\}$) such that the scheme is private and capacity achieving.

message permutations. Then, the permutation of the elements of W_i is given by, $\Gamma(W_i(1), \ldots, W_i(\ell_i)) = (W_i(\gamma_i(1)), \ldots, W_i(\gamma_i(\ell_i)))$. This process simply shuffles the elements of message vectors uniformly and randomly irrespective of the message requirement. All queries generated by the user in the scheme are based on these permuted indices.

3. **Singletons:** Download υ_k different bits from message W_k from the nth database, where $n = 1, \ldots, N$ and $k = 1, \ldots, K$. Table 2.1 shows the singletons downloaded from the required message W_j and any other message W_i, $i \neq j$. Note that the permuted elements of W_j and W_i are denoted by a's and b's respectively.

Message	Database 1	Database 2	...	Database N
W_j	$a_1, \ldots, a_{\upsilon_j}$	$a_{\upsilon_j+1}, \ldots, a_{2\upsilon_j}$	...	$a_{(N-1)\upsilon_j+1}, \ldots, a_{N\upsilon_j}$
$W_i, i \neq j$	$b_1, \ldots, b_{\upsilon_i}$	$b_{\upsilon_i+1}, \ldots, b_{2\upsilon_i}$	...	$b_{(N-1)\upsilon_i+1}, \ldots, b_{N\upsilon_i}$

Table 2.1: Singleton queries.

4. **Sums of two elements (2-sums):** There are two types of blocks in this step. The first block is the sums involving bits of the desired message, W_j, and the other block is the sums that do not have any bits from W_j. In the first block, download $(N-1)\min\{\upsilon_i, \upsilon_j\}$ bit-wise sums of W_i and W_j each from the N databases for all $i \neq j$. Each sum comprises an already downloaded W_i bit from another database and a new bit of W_j. I.e., if $\upsilon_j > \upsilon_i$ user sends queries of the form $(a_{N\upsilon_j+1}+b_{\upsilon_i+1}), \ldots, (a_{N\upsilon_j+\upsilon_i}+b_{2\upsilon_i}), \ldots, (a_{N\upsilon_j+(N-2)\upsilon_i+1}+b_{(N-1)\upsilon_i+1}), \ldots, (a_{N\upsilon_j+(N-1)\upsilon_i} + b_{N\upsilon_i})$ to database 1. Note that each $\min\{\upsilon_i, \upsilon_j\} = \upsilon_i$ side information bit downloaded from each of the databases 2 to N in the previous

step have been utilized exactly once in the 2-sums of database 1. Queries of the same form are sent to all databases, which contain new bits of W_j and all the already downloaded bits of W_i, $i \neq j$ from the rest of the databases. Each side information bit from the previous step is utilized only once in a given database.

If $\min\{\upsilon_i, \upsilon_j\} = \upsilon_j$ user can randomly pick any υ_j side information bits out of the υ_i bits from each database and follow the same steps as above, ensuring that any given side information bit from a different database in the previous step is utilized only once in a given database.

For the second block, for all possible message pairs (W_{i_1}, W_{i_2}) for $i_1 \neq i_2 \neq j$, download $(N-1)\min\{\upsilon_{i_1}, \upsilon_{i_2}\}$ number of bit-wise sums of W_{i_1} and W_{i_2} each from the N databases. Each sum comprises of fresh bits from W_{i_1} and W_{i_2}.

5. **Repeat step 4** for all k-sums where $k = 3, 4, \ldots, K$. For each k-sum, download k bit-wise sum from k messages. If one of these messages is the desired message, the remaining $(k-1)$-sum is derived from the previous $(k-1)$th round from a different database. Otherwise, download $(N-1)^{k-1}\min\{\upsilon_{i_1}, \ldots, \upsilon_{i_k}\}$ sums from new bits of the undesired messages.

2.4.1.1 Rate of Semantic PIR Scheme 1

In this PIR scheme, the total number of downloaded bits remains constant for all message requirements of the user in order to guarantee privacy. Therefore, $\mathbb{E}[D]$ in (2.8) can be calculated by counting the total number of bits in the set of queries sent

to the databases by the user to download any message. Within the set of queries, there are $\sum_{i=1}^{K} N\upsilon_i$ number of singletons and $\sum_{i=t}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-1}{t-1}$ number of sums of t elements. Therefore,

$$\begin{aligned}
\mathbb{E}[D] &= \sum_{i=1}^{K} N\upsilon_i + \sum_{t=2}^{K}\sum_{i=t}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-1}{t-1} && (2.26)\\
&= N\left[\sum_{i=1}^{K}\upsilon_i + \sum_{i=2}^{K}\sum_{t=2}^{i}(N-1)^{t-1}\upsilon_i\binom{i-1}{t-1}\right] && (2.27)\\
&= N\left[\sum_{i=1}^{K}\upsilon_i + \sum_{i=2}^{K}\upsilon_i\left(\sum_{t=0}^{i-1}(N-1)^{t}\binom{i-1}{t} - 1\right)\right] && (2.28)\\
&= N\left[\sum_{i=1}^{K}\upsilon_i + \sum_{i=2}^{K}\upsilon_i\left(N^{i-1}-1\right)\right] && (2.29)\\
&= \sum_{i=1}^{K}\upsilon_i N^i && (2.30)
\end{aligned}$$

In order to calculate $\mathbb{E}[L]$, assume that the desired message is W_j. There are $N\upsilon_j$ number of singletons of W_j in the set of queries sent to the databases to retrieve W_j. The scheme can recover $N(N-1)^{t-1}\upsilon_j\binom{j-1}{t-1} + N(N-1)^{t-1}\sum_{i=j+1}^{K}\upsilon_i\binom{i-2}{t-2}$ number of W_j bits using the tth block of the scheme (sum of t elements) when $t \leq j$, where the first term in the sum corresponds to t-sums with the shortest message being W_j and the second term corresponds to t-sums with the shortest message being some other message ($\neq W_j$). When $t > j$ this scheme is able to retrieve $\sum_{i=t}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-2}{t-2}$ number of W_j bits as there should be at least $t-j$ number of messages in the sum that are shorter than L_j. Therefore, the total number of

useful bits of W_j retrieved, U_j, is given by,

$$
\begin{aligned}
U_j =& N\upsilon_j + \sum_{t=2}^{j}\left(N(N-1)^{t-1}\upsilon_j\binom{j-1}{t-1} + \sum_{i=j+1}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-2}{t-2}\right) \\
&+ \sum_{t=j+1}^{K}\sum_{i=t}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-2}{t-2} && (2.31)\\
=& N\upsilon_j\left(1+\sum_{t=2}^{j}(N-1)^{t-1}\binom{j-1}{t-1}\right) + \sum_{t=2}^{j}\sum_{i=j+1}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-2}{t-2} \\
&+ \sum_{t=j+1}^{K}\sum_{i=t}^{K} N(N-1)^{t-1}\upsilon_i\binom{i-2}{t-2} && (2.32)\\
=& N\upsilon_j\left(1+(N-1)\binom{j-1}{1}+(N-1)^2\binom{j-1}{2}+\cdots+(N-1)^{j-1}\binom{j-1}{j-1}\right) \\
&+ N\upsilon_{j+1}\left(\sum_{t=2}^{j}(N-1)^{t-1}\binom{j-1}{t-2}\right) + N\upsilon_{j+2}\left(\sum_{t=2}^{j}(N-1)^{t-1}\binom{j}{t-2}\right)+\dots \\
&+ N\upsilon_K\left(\sum_{t=2}^{j}(N-1)^{t-1}\binom{K-2}{t-2}\right) + N\upsilon_{j+1}(N-1)^j\binom{j-1}{j-1} \\
&+ N\upsilon_{j+2}\left((N-1)^j\binom{j}{j-1}+(N-1)^{j+1}\binom{j}{j}\right)+\dots+N\upsilon_K\left((N-1)^j\binom{K-2}{j-1}\right. \\
&\left.+(N-1)^{j+1}\binom{K-2}{j}+\cdots+(N-1)^{K-1}\binom{K-2}{K-2}\right) && (2.33)\\
=& N\upsilon_j(N-1+1)^{j-1} + N\upsilon_{j+1}\left((N-1)\binom{j-1}{0}+(N-1)^2\binom{j-1}{1}\right. \\
&\left.+\cdots+(N-1)^j\binom{j-1}{j-1}\right) + N\upsilon_{j+2}\left((N-1)\binom{j}{0}+(N-1)^2\binom{j}{1}+\dots\right. \\
&\left.+(N-1)^{j+1}\binom{j}{j}\right)+\cdots+N\upsilon_k\left((N-1)\binom{K-2}{0}+(N-1)^2\binom{K-2}{1}\right. \\
&\left.+\cdots+(N-1)^{K-1}\binom{K-2}{K-2}\right) && (2.34)\\
=& N^j\upsilon_j + N(N-1)(N-1+1)^{j-1}\upsilon_{j+1} + N(N-1)(N-1+1)^j\upsilon_{j+2}+\dots \\
&+ N(N-1)(N-1+1)^{K-2}\upsilon_K && (2.35)\\
=& N^j\upsilon_j + (N-1)\sum_{i=j+1}^{K} N^{i-1}\upsilon_i && (2.36)
\end{aligned}
$$

Thus, the scheme retrieves $N^j \upsilon_j + (N-1)\sum_{i=j+1}^{K} N^{i-1}\upsilon_i$ number of useful bits of the required message at a time. Hence, we define *subpacketization* for message W_j as U_j, where

$$U_j = N^j \upsilon_j + (N-1)\sum_{i=j+1}^{K} N^{i-1}\upsilon_i, \quad j = 1, \ldots, K \tag{2.37}$$

We then need the message sizes to be a common multiple of their own subpacketizations,

$$L_j = \alpha U_j, \quad j = 1, \ldots, K \tag{2.38}$$

We note that α should be the same for all j in (2.38) to guarantee privacy.

The requirements in (2.37) and (2.38) can be written succinctly as a matrix equation,

$$\begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_K \end{bmatrix} = \alpha \begin{bmatrix} N & N(N-1) & \ldots & N^{K-1}(N-1) \\ 0 & N^2 & \ldots & N^{K-1}(N-1) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & N^K \end{bmatrix} \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \vdots \\ \upsilon_K \end{bmatrix} \tag{2.39}$$

Since $L_1, \ldots, L_K$ are parameters (inputs) to the scheme, the internal parameters

$\upsilon_1, \ldots, \upsilon_K$ can be calculated by inverting the matrix as,

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \\ \vdots \\ \upsilon_K \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \frac{1}{N} & -\frac{N-1}{N^2} & -\frac{N-1}{N^3} & \ldots & -\frac{N-1}{N^K} \\ 0 & \frac{1}{N^2} & -\frac{N-1}{N^3} & \ldots & -\frac{N-1}{N^K} \\ 0 & 0 & \frac{1}{N^3} & \ldots & -\frac{N-1}{N^K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{1}{N^K} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ \vdots \\ L_K \end{bmatrix} \tag{2.40}$$

Here, α should be chosen to be the greatest common divisor (gcd) of the elements of the vector resulting from multiplying the matrix and the vector on the right side of (2.40). This allows the shortest subpacketization levels for all messages for increased flexibility.

The total number of bits downloaded calculated in (2.30) and the number of useful bits downloaded calculated in (2.36) are both within one subpacketization level. This subpacketization level downloads are repeated α times to download the entire file; see also (2.38). Thus, we calculate the achievable rate of this scheme as,

$$R = \frac{\mathbb{E}[L]}{\mathbb{E}[D]} \tag{2.41}$$

$$= \frac{\sum_{i=1}^{K} p_i U_i}{\sum_{i=1}^{K} N^i \upsilon_i} \tag{2.42}$$

$$= \frac{\frac{1}{\alpha} \sum_{i=1}^{K} p_i L_i}{\sum_{i=1}^{K} \frac{1}{\alpha} N^i (N^{-i} L_i - \sum_{j=i+1}^{K} (N-1) N^{-j} L_j)} \tag{2.43}$$

$$= \frac{\mathbb{E}[L]}{\sum_{i=1}^{K} L_i - (N-1) \sum_{i=1}^{K} \sum_{j=i+1}^{K} N^{-j} L_j N^i} \tag{2.44}$$

$$= \frac{\mathbb{E}[L]}{\sum_{i=1}^{K} L_i - (N-1) \left(\sum_{j=2}^{K} N^{-j+1} L_j + \sum_{j=3}^{K} N^{-j+2} L_j + \cdots + N^{-1} L_K \right)} \tag{2.45}$$

$$=\frac{\mathbb{E}[L]}{L_1 + L_2\left(1-(N-1)N^{-1}\right) + \cdots + L_K\left(1-(N-1)(N^{-(K-1)}+\cdots+N^{-1})\right)} \tag{2.46}$$

$$=\frac{\mathbb{E}[L]}{L_1 + \frac{L_2}{N} + \frac{L_3}{N^2} + \cdots + \frac{L_K}{N^{K-1}}} \tag{2.47}$$

$$=\left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_2}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\mathbb{E}[L]}\right)^{-1} \tag{2.48}$$

where (2.43) follows by applying (2.38) in the numerator and writing υ_i in terms of L_j using (2.40) in the denominator. This concludes the derivation of the achievable rate.

Remark 2.7 *We assume that message L_i has a length which is a multiple of N^i to aid smooth computation of $\upsilon_1, \ldots, \upsilon_K$. This is automatically satisfied by the assumption of all message lengths being multiples of N^K in [3].*

2.4.1.2 Proof of Privacy

Since $L_1 \geq L_2 \geq \cdots \geq L_K$ we have $\upsilon_1 \geq \upsilon_2 \geq \cdots \geq \upsilon_K$. A given database receives a set of queries for $\upsilon_1, \upsilon_2, \ldots, \upsilon_K$ numbers of bits of $W_1, W_2, \ldots, W_K$, respectively, as singletons and $(N-1)^{t-1}\min\{\upsilon_{i_1}, \ldots, \upsilon_{i_t}\}$ bit-wise t-sums of $W_{i_1}, \ldots, W_{i_t}$, for $t = 2, \ldots, K$. According to the query generation procedure, no bit of any message is requested from a given database more than once as a singleton or as an element of a sum. Any given database receives the exact same set of queries in type, irrespective of the desired message of the user. Therefore, two sets of queries corresponding to two different message requirements received by a given database

can only differ from the permutations used in each message in the index preparation step. Let $(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K)$ be a sample realization of permutations used in the query generation process when the message requirement θ is W_k, where $\mathbf{w}_i = \Gamma_i(W_i(1), W_i(2), \ldots, W_i(\ell_i))$ for $i \in \{1, \ldots, K\}$ with permutation functions Γ_i that independently and randomly permute the ℓ_i elements of W_i, where ℓ_i is the subpacketization of W_i. Therefore, the probability of sending the set of queries q for a given message requirement $\theta = k$ is equal to the probability of choosing the corresponding sample realization of permutations of the message bits when downloading W_k. This probability is calculated by,

$$P(Q_n = q|\theta = k) = P(\text{permutation} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K)|\theta = k) \tag{2.49}$$

$$= \prod_{i=1}^{K} P(\text{permutation of } W_i = \mathbf{w_i}|\theta = k) \tag{2.50}$$

$$= \prod_{i=1}^{K} \left(\frac{1}{\ell_i}\right)\left(\frac{1}{\ell_i - 1}\right)\cdots\left(\frac{1}{\ell_i - \ell_i + 1}\right) \tag{2.51}$$

for $n \in \{1, \ldots, N\}$, where Q_n is the random variable representing the set of queries sent to database n. This yields,

$$P(Q_n = q|\theta = i) = P(Q_n = q|\theta = j), \quad i, j \in \{1, \ldots, K\}, \quad n \in \{1, \ldots, N\} \tag{2.52}$$

as $P(Q_n = q|\theta = k)$ is independent of k by the above calculation. The a posteriori probability of the user needing W_i given a realization of the set of queries received

by any given database is given by,

$$P(\theta = i|Q = q) = \frac{P(Q = q|\theta = i)P(\theta = i)}{\sum_{j=1}^{K} P(Q = q|\theta = j)P(\theta = j)} \tag{2.53}$$

Using (2.52),

$$P(\theta = i|Q = q) = \frac{P(Q = q|\theta = i)P(\theta = i)}{\sum_{j=1}^{K} P(Q = q|\theta = i)P(\theta = j)} \tag{2.54}$$

$$= P(\theta = i) \tag{2.55}$$

which ensures that this scheme is private, since it implies that θ and Q are independent.

2.4.2 Examples of Semantic PIR Scheme 1

2.4.2.1 Example 1: $N = 2, K = 2$, $L_1 = 1024$ bits, $L_2 = 256$ bits

First, the message indices are independently and uniformly permuted. The first and the second messages after permutations are denoted by bits a_i and b_i, respectively.

- Message indexing and calculation of υ_i: Messages are indexed such that the first message is the longer one, and the second message is the shorter one. Below, we will give query tables for downloading W_1 and W_2. We calculate υ_1

and υ_2 as,

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} \\ 0 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} \tag{2.56}$$

where α =gcd$\{\frac{L_1}{2} - \frac{L_2}{4}, \frac{L_2}{4}\}$. By direct substitution, we get,

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} 448 \\ 64 \end{bmatrix} \tag{2.57}$$

Hence, α =gcd$\{448, 64\} = 64$. Therefore, $\upsilon_1 = 7$ and $\upsilon_2 = 1$. The sub-packetization levels of W_1 and W_2 are $U_1 = \frac{1024}{64} = 16$ and $U_2 = \frac{256}{64} = 4$, respectively.

- Singletons: Download $\upsilon_1 = 7$ bits of W_1 and $\upsilon_2 = 1$ bit of W_2 each from the two databases.

- Sums of twos: Download $(N-1)\upsilon_2 = 1$ sum of W_1 and W_2 bits each from the two databases. Note that if W_1 is the desired message, the singletons of W_2 are used as a side information with new W_1 bits in the sum and vice versa.

Tables 2.2 and 2.3 show the queries sent to the databases to retrieve W_1 and W_2, respectively.

Database 1	Database 2
$a_1, \ldots, a_7$ b_1	$a_8, \ldots, a_{14}$ b_2
$a_{15} + b_2$	$a_{16} + b_1$

Table 2.2: The query table for the retrieval of W_1.

Database 1	Database 2
$a_1, \ldots, a_7$ b_1	$a_8, \ldots, a_{14}$ b_2
$a_8 + b_3$	$a_1 + b_4$

Table 2.3: The query table for the retrieval of W_2.

The rate achieved by this scheme when downloading W_1 is $R_1 = \frac{16}{18} = \frac{8}{9}$, and the rate achieved by this scheme when downloading W_2 is $R_2 = \frac{4}{18} = \frac{2}{9}$. Therefore, the average rate R achieved by the scheme is,

$$R = \frac{\mathbb{E}[L]}{\mathbb{E}[D]} = \frac{p_1 L_1 + p_2 L_2}{p_1 D + p_2 D} = p_1 \frac{L_1}{D} + p_2 \frac{L_2}{D} = p_1 R_1 + p_2 R_2 = \frac{8}{9} p_1 + \frac{2}{9} p_2 \tag{2.58}$$

This matches the capacity expression in Theorem 2.1 as,

$$C = \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N} \frac{L_2}{\mathbb{E}[L]} \right)^{-1} \tag{2.59}$$

$$= (1024 p_1 + 256 p_2) \left(1024 + \frac{256}{2} \right)^{-1} \tag{2.60}$$

$$= \frac{8}{9} p_1 + \frac{2}{9} p_2 \tag{2.61}$$

The classic PIR capacity for this case with equal priors is,

$$C = \left(1 + \frac{1}{N} \right)^{-1} = \left(1 + \frac{1}{2} \right)^{-1} = \frac{2}{3} \tag{2.62}$$

The semantic PIR capacity in (2.61) exceeds the classical PIR capacity in (2.62) when

$$\frac{8}{9} p_1 + \frac{2}{9} p_2 > \frac{2}{3} \tag{2.63}$$

which is when $p_1 > \frac{2}{3}$. Consequently, when $p_1 > \frac{2}{3}$, there is a strict gain from exploiting message semantics for PIR, in this case.

Remark 2.8 *Although it is apparent in this example that the rate of semantic PIR is lower than the capacity of classical PIR for $p_1 < \frac{2}{3}$, as discussed in Remark 2.3 and Remark 2.4, there is a subtle aspect that should be addressed for a fair comparison. To see this, let us take the case of uniform a priori distribution, i.e., $p_1 = p_2 = \frac{1}{2}$, i.e., a case where $p_1 < \frac{2}{3}$. In this case, the semantic PIR capacity using* (2.61) *is $\frac{5}{9}$. In order to properly use the classical PIR scheme in [3], messages need to be of equal size. One way to do this is to zero-pad the shorter message to be of length 1024 bits as well. In this case, the actual retrieval rate is not $\frac{2}{3}$ as the actual message size of W_2 is much less. Specifically, the total download for this scheme is $D = \frac{L}{R} = \frac{1024}{2/3} = 1536$. The actual retrieval rate for the classical PIR problem is,*

$$R_{ach} = \frac{1/2 \times 1024 + 1/2 \times 256}{1536} = \frac{5}{12} < \frac{5}{9} < \frac{6}{9} \tag{2.64}$$

Thus, the actual achievable rate R_{ach} is $\frac{5}{12}$, which is less than the semantic PIR capacity $\frac{5}{9}$, which is less than the classical PIR capacity $\frac{6}{9}$. Thus, even though the semantic PIR capacity is less than the classical PIR capacity, the semantic PIR capacity (which is achievable) is larger than the classical PIR rate with zero-padding as proved in Corollary 2.2.

2.4.2.2 Example 2: $N = 4, K = 3$, $L_1 = 8192$ bits, $L_2 = 2048$ bits, $L_3 = 512$ bits

First, the message indices are independently and uniformly permuted. The first, second, and third messages after permutations are denoted by bits a_i, b_i and c_i, respectively.

- Message indexing and calculation of υ_i: Messages are indexed such that the first message is the longest one, and the third message is the shortest one. Below, we will give the query table for downloading W_2, i.e., the medium-length message. The bits of W_2 are represented by b_i. We calculate υ_1, υ_2 and υ_3 as,

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \frac{1}{4} & -\frac{3}{16} & -\frac{3}{64} \\ 0 & \frac{1}{16} & -\frac{3}{64} \\ 0 & 0 & \frac{1}{64} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{2.65}$$

where $\alpha = \text{gcd}\{\frac{L_1}{4} - \frac{3L_2}{16} - \frac{3L_3}{64}, \frac{L_2}{16} - \frac{3L_3}{64}, \frac{L_3}{64}\}$. By direct substitution, we get,

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} 1640 \\ 104 \\ 8 \end{bmatrix} \tag{2.66}$$

Hence, $\alpha = \text{gcd}\{1640, 104, 8\} = 8$. Therefore, $\upsilon_1 = 205$, $\upsilon_2 = 13$ and $\upsilon_3 = 1$. The subpacketization levels of W_1, W_2 and W_3 are $U_1 = \frac{8192}{8} = 1024$, $U_2 =$

$\frac{2048}{8} = 256$ and $U_3 = \frac{512}{8} = 64$, respectively.

- Singletons: Download $\upsilon_1 = 205$ bits of W_1, $\upsilon_2 = 13$ bits of W_2 and $\upsilon_3 = 1$ bits of W_3 each from the four databases.

- Sums of twos: Download $(N-1)\upsilon_2 = 39$ sums of W_1 and W_2 and $(N-1)\upsilon_3 = 3$ sums of W_2 and W_3 bits each from the four databases. Use the downloaded singletons from W_1, W_3 as side information with new W_2 bits. Download $(N-1)\upsilon_3 = 3$ bit-wise sums of W_1 and W_3 each from the four databases using fresh bits of both messages.

- Sums of threes: Download $(N-1)^2\upsilon_3 = 9$ bit-wise sums involving all three messages from each database utilizing the downloaded sums of W_1 and W_3 from the other databases in the previous step as side information.

Table 2.4 shows the queries sent to the databases to retrieve W_2.

The rate achieved by this scheme when downloading W_2 is $R_2 = \frac{256}{1092} = \frac{64}{273}$, and the rates achieved when downloading W_1 and W_3 are $R_1 = \frac{1024}{1092} = \frac{256}{273}$ and $R_3 = \frac{64}{1092} = \frac{16}{273}$, respectively. Therefore, the average rate R achieved by this scheme is,

$$R = \frac{\mathbb{E}[L]}{\mathbb{E}[D]} = \frac{p_1 L_1 + p_2 L_2 + p_3 L_3}{p_1 D + p_2 D + p_3 D} \tag{2.67}$$

$$= p_1 \frac{L_1}{D} + p_2 \frac{L_2}{D} + p_3 \frac{L_3}{D} = p_1 R_1 + p_2 R_2 + p_3 R_3 \tag{2.68}$$

$$= \frac{256}{273} p_1 + \frac{64}{273} p_2 + \frac{16}{273} p_3 \tag{2.69}$$

Database 1	Database 2	Database 3	Database 4
$a_1, \ldots, a_{205}$ $b_1, \ldots, b_{13}$ c_1	$a_{206}, \ldots, a_{410}$ $b_{14}, \ldots, b_{26}$ c_2	$a_{411}, \ldots, a_{615}$ $b_{27}, \ldots, b_{39}$ c_3	$a_{616}, \ldots, a_{820}$ $b_{40}, \ldots, b_{52}$ c_4
$a_{206} + b_{53}$ $\vdots$ $a_{218} + b_{65}$ $a_{411} + b_{66}$ $\vdots$ $a_{423} + b_{78}$ $a_{616} + b_{79}$ $\vdots$ $a_{628} + b_{91}$	$a_{411} + b_{92}$ $\vdots$ $a_{423} + b_{104}$ $a_{616} + b_{105}$ $\vdots$ $a_{628} + b_{117}$ $a_1 + b_{118}$ $\vdots$ $a_{13} + b_{130}$	$a_{616} + b_{131}$ $\vdots$ $a_{628} + b_{143}$ $a_1 + b_{144}$ $\vdots$ $a_{13} + b_{156}$ $a_{206} + b_{157}$ $\vdots$ $a_{218} + b_{169}$	$a_1 + b_{170}$ $\vdots$ $a_{13} + b_{182}$ $a_{206} + b_{183}$ $\vdots$ $a_{218} + b_{195}$ $a_{411} + b_{196}$ $\vdots$ $a_{423} + b_{208}$
$b_{209} + c_2$ $b_{210} + c_3$ $b_{211} + c_4$	$b_{212} + c_3$ $b_{213} + c_4$ $b_{214} + c_1$	$b_{215} + c_4$ $b_{216} + c_1$ $b_{217} + c_2$	$b_{218} + c_1$ $b_{219} + c_2$ $b_{220} + c_3$
$a_{821} + c_5$ $a_{822} + c_6$ $a_{823} + c_7$	$a_{824} + c_8$ $a_{825} + c_9$ $a_{826} + c_{10}$	$a_{827} + c_{11}$ $a_{828} + c_{12}$ $a_{829} + c_{13}$	$a_{830} + c_{14}$ $a_{831} + c_{15}$ $a_{832} + c_{16}$
$a_{824} + b_{221} + c_8$ $a_{825} + b_{222} + c_9$ $a_{826} + b_{223} + c_{10}$ $a_{827} + b_{224} + c_{11}$ $a_{828} + b_{225} + c_{12}$ $a_{829} + b_{226} + c_{13}$ $a_{830} + b_{227} + c_{14}$ $a_{831} + b_{228} + c_{15}$ $a_{832} + b_{229} + c_{16}$	$a_{827} + b_{230} + c_{11}$ $a_{828} + b_{231} + c_{12}$ $a_{829} + b_{232} + c_{13}$ $a_{830} + b_{233} + c_{14}$ $a_{831} + b_{234} + c_{15}$ $a_{832} + b_{235} + c_{16}$ $a_{821} + b_{236} + c_5$ $a_{822} + b_{237} + c_6$ $a_{823} + b_{238} + c_7$	$a_{830} + b_{239} + c_{14}$ $a_{831} + b_{240} + c_{15}$ $a_{832} + b_{241} + c_{16}$ $a_{821} + b_{242} + c_5$ $a_{822} + b_{243} + c_6$ $a_{823} + b_{244} + c_7$ $a_{824} + b_{245} + c_8$ $a_{825} + b_{246} + c_9$ $a_{826} + b_{247} + c_{10}$	$a_{821} + b_{248} + c_5$ $a_{822} + b_{249} + c_6$ $a_{823} + b_{250} + c_7$ $a_{824} + b_{251} + c_8$ $a_{825} + b_{252} + c_9$ $a_{826} + b_{253} + c_{10}$ $a_{827} + b_{254} + c_{11}$ $a_{828} + b_{255} + c_{12}$ $a_{829} + b_{256} + c_{13}$

Table 2.4: The query table for the retrieval of W_2.

This matches the capacity expression in Theorem 2.1 as,

$$C = \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N} \frac{L_2}{\mathbb{E}[L]} + \frac{1}{N^2} \frac{L_3}{\mathbb{E}[L]} \right)^{-1} \tag{2.70}$$

$$= (8192p_1 + 2048p_2 + 512p_3) \left(8192 + \frac{2048}{4} + \frac{512}{4^2} \right)^{-1} \tag{2.71}$$

$$= \frac{256}{273} p_1 + \frac{64}{273} p_2 + \frac{16}{273} p_3 \tag{2.72}$$

The classical PIR capacity for this case with equal priors is,

$$C = \left(1 + \frac{1}{N} + \frac{1}{N^2}\right)^{-1} = \left(1 + \frac{1}{4} + \frac{1}{4^2}\right)^{-1} = \frac{16}{21} \tag{2.73}$$

The semantic PIR capacity in (2.72) exceeds the classical PIR capacity in (2.73) when

$$\frac{256}{273}p_1 + \frac{64}{273}p_2 + \frac{16}{273}p_3 > \frac{16}{21} \tag{2.74}$$

which is equivalent to

$$p_1 + \frac{1}{5}p_2 > \frac{4}{5} \tag{2.75}$$

2.4.3 Alternative Description of Semantic PIR Scheme 1

In this section, we present an alternative description to the semantic PIR scheme presented in Section 2.4.1. The two descriptions are identical in terms of the queries generated considering the retrieval of the entire required message (all subpackets). However, the two descriptions differ in subpacketization and the scheme used within a subpacket.

Consider the general semantic PIR setting with K messages with arbitrary message lengths $L_1 \geq L_2 \geq \ldots \geq L_K$ and arbitrary probabilities of retrieval p_i, $i \in \{1, \ldots, K\}$. The alternative description requires the messages to be partitioned in to K segments, such that the first segment contains the first L_K bits of all messages,

the second segment contains the next $L_{K-1}-L_K$ bits of messages $W_1,\ldots,W_{K-1}$ and the ℓth segment for $\ell \in \{3,\ldots,K\}$ contains $L_{K-\ell+1}-L_{K-\ell+2}$ bits of $W_1,\ldots,W_{K-\ell+1}$ that follow the bits in the $(\ell-1)$st segment.

Apply the classical PIR scheme in [3] to the 1) first segment with K messages with a subpacketization of N^K, 2) second segment with $K-1$ messages with a subpacketization of N^{K-1} and 3) ℓth segment with $K-\ell+1$ messages with a subpacketization of $N^{K-\ell+1}$, for $\ell \in \{3,\ldots,K\}$. The above three steps need to be followed irrespective of the message requirement for privacy. Note that the schemes used in each segment are private [3], and the fact that the K schemes corresponding to the K segments are always used, even though the required message may not be within a given segment, guarantees privacy. The achievable rate of this scheme is calculated as follows. The fixed download cost is given by,

$$
\begin{aligned}
D =& \frac{L}{R} && (2.76)\\
=& \frac{L_K}{\left(1+\frac{1}{N}+\cdots+\frac{1}{N^{K-1}}\right)^{-1}} + \frac{L_{K-1}-L_K}{\left(1+\frac{1}{N}+\cdots+\frac{1}{N^{K-2}}\right)^{-1}} + \ldots + \frac{L_2-L_3}{\left(1+\frac{1}{N}\right)^{-1}}\\
&+ \frac{L_1-L_2}{1} && (2.77)\\
=& L_K\frac{1}{N^{K-1}} + L_{K-1}\frac{1}{N^{K-2}} + \ldots + L_2\frac{1}{N} + L_1 && (2.78)
\end{aligned}
$$

Therefore, the achievable rate is,

$$
\begin{aligned}
R =& \frac{\mathbb{E}[L]}{D} && (2.79)\\
=& \frac{\mathbb{E}[L]}{L_K\frac{1}{N^{K-1}} + L_{K-1}\frac{1}{N^{K-2}} + \ldots + L_2\frac{1}{N} + L_1} && (2.80)
\end{aligned}
$$

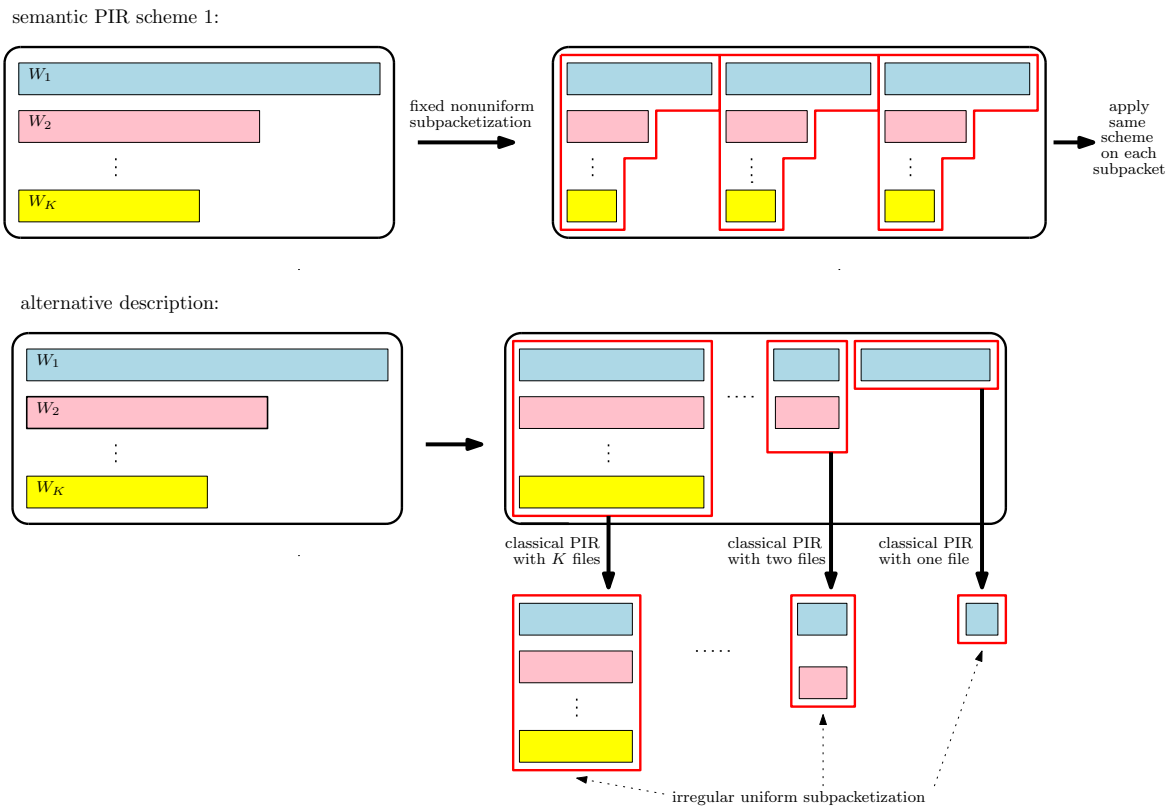


Figure 2.1: Comparison of the two descriptions of semantic PIR scheme 1.

which is the capacity of semantic PIR in (2.9). Note that the description in Section 2.4.1 provides a *systematic* way of calculating the nonuniform subpacketization based on the given set of message lengths. The scheme is then described on a single subpacket, which is repeatedly applied throughout the retrieval process in the same way. On the other hand, the alternative description has different uniform subpacketizations for different segments. Therefore, the scheme needs to be specified for each segement separately. This is illustrated in Fig. 2.1.

2.4.4 Achievable PIR Scheme 2

The scheme is stochastic in the sense that the user has a list of different possible query structures and the user picks one of these structures randomly. This is unlike

the previous scheme where the structure is deterministic and the randomness comes from the random permutations of indices.

This scheme is developed for arbitrary number of databases and arbitrary message lengths that are multiples of $N-1$; the deterministic scheme in Sections 2.4.1 and 2.4.2 assumed message lengths that are multiples of N^K. The scheme can be viewed as an extension of the achievable scheme in [49] to work with arbitrary number of databases and heterogeneous message sizes. Our scheme shares similarities with [52]. However, our scheme differs in that it introduces database symmetry to the scheme. The basic structure of the achievable scheme is as follows.

1. **Message indexing:** Index all messages such that $L_1 \geq L_2 \geq \cdots \geq L_K$. Divide all messages into $N-1$ blocks. Let W_i^m be the mth block of W_i.

   For the rest of this section, assume that the user requires to download W_j.

2. **Single blocks:** Use $N-1$ out of the N databases to download each block of W_j and download nothing from the remaining database. Consider all N cyclic shifts of the blocks around the databases to obtain N options for different queries that can be used to download W_j. These N queries require the user to download L_j bits in total, resulting in no side information.

3. **Sums of two blocks/single blocks:** Choose one database to download W_i^1 where $i \neq j$ and download $W_j^m + W_i^1$ for $m = 1, \ldots, N-1$ from the remaining $N-1$ databases. Create N query options in total by considering all N cyclic shifts of the blocks, around the databases. Repeat the procedure for W_i^ℓ where

$\ell = 2, \ldots, N-1$. There are a total of $N(N-1)\binom{K-1}{1}$ query options of this type.

4. **Sums of three blocks/sums of two blocks:** Choose one database to download $W_{i_1}^1 + W_{i_2}^1$ where $i_1, i_2 \neq j$ and download $W_j^m + W_{i_1}^1 + W_{i_2}^1$ for $m = 1, \ldots, N-1$ from the remaining $N-1$ databases. Create N query options in total by considering all N cyclic shifts of the blocks around the databases. Repeat the procedure for $W_{i_1}^{\ell_1} + W_{i_2}^{\ell_2}$ where $\ell_1, \ell_2 \in \{2, \ldots, N-1\}$. There are $N(N-1)^2\binom{K-1}{2}$ query options of this type.

5. **Repeat step 4** up to sums of K blocks/sums of $K-1$ blocks.

The above steps describe all the N^K query options, out of which the user selects one with equal probability to retrieve the required message. Note that due to the cyclic shifts of all queries, this scheme has database symmetry, and the exact same set of queries constitutes the possible set of queries received by any given database, irrespective of the desired message of the user.

Once the user chooses a query to be sent to the N databases, out of the N^K options, each database might have to compute sums of messages with different lengths. All messages except the longest in the sum are zero-padded to the left to have equal-length blocks. Then, bit-wise sums are calculated.

Once the answers are received from the databases, the user might need to subtract messages of different lengths to recover the required message. In this case, according to the design of the scheme, the subtrahend will always be shorter than or equal to the length of the minuend. Hence, the subtraction operation in this

context will not be any different than the usual operation.

Remark 2.9 *Each query is chosen with probability* $\frac{1}{N^K}$ *as there are* $\sum_{t=0}^{K}(N-1)^t\binom{K}{t} = N^K$ *number of query options in total. Each element of the sum corresponds to the number of t-sums within the set of all possible queries that can be sent to a given database.*

2.4.4.1 Rate of Semantic PIR Scheme 2

In this PIR scheme, each query option is utilized by the user with a probability of $\frac{1}{N^K}$ to download any desired message. When analyzing all possible queries that can be sent to all databases, we note that they have the same entries (in a shuffled way) irrespective of the desired message. Since all query entries are equally probable to be sent to the databases, we calculate $\mathbb{E}[D]$ by,

$$\mathbb{E}[D] = \sum_{i=1}^{K} p_i \frac{1}{N^K}\left(\sum_{t=1}^{K}\sum_{j=1}^{K-t+1} L_j (N-1)^{t-1}\binom{K-j}{t-1}\right)N \tag{2.81}$$

$$= \frac{1}{N^{K-1}}\sum_{j=1}^{K}\sum_{t=1}^{K-j+1} L_j (N-1)^{t-1}\binom{K-j}{t-1} \tag{2.82}$$

$$= \frac{1}{N^{K-1}}\sum_{j=1}^{K} L_j \sum_{t=0}^{K-j}(N-1)^{t}\binom{K-j}{t} \tag{2.83}$$

$$= \frac{1}{N^{K-1}}\sum_{j=1}^{K} L_j N^{K-j} \tag{2.84}$$

$$= L_1 + \frac{L_2}{N} + \frac{L_3}{N^2} + \cdots + \frac{L_K}{N^{K-1}} \tag{2.85}$$

where the second and third sums in (2.81) correspond to different t-sums and all possible longest messages within the t-sum, respectively. The p_i terms are ignored

in (2.82) as the expected number of downloads per query set does not depend on the desired message.

For a given desired message, the number of downloaded useful bits is the length of the desired message (ignoring zero-padding, as it is ignored by the user upon receiving the answer strings). This remains constant regardless of the query set utilized by the user. Hence,

$$\mathbb{E}[L] = \sum_{i=1}^{K} p_i L_i \tag{2.86}$$

Thus, combining (2.85) and (2.86), the achievable rate of this scheme becomes,

$$R = \frac{\mathbb{E}[L]}{\mathbb{E}[D]} \tag{2.87}$$

$$= \frac{\mathbb{E}[L]}{L_1 + \frac{L_2}{N} + \frac{L_3}{N^2} \cdots + \frac{L_K}{N^{K-1}}} \tag{2.88}$$

$$= \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_2}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\mathbb{E}[L]} \right)^{-1} \tag{2.89}$$

This concludes the derivation of the achievable rate.

2.4.4.2 Proof of Privacy

The scheme is constructed in such a way that any given database always receives a query out of the set of queries given by, $\{\phi, \{W_i^\ell, i \in \{1, \ldots, K\}, \ell \in \{1, \ldots, N-1\}\}, \{W_{i_1}^{\ell_1} + \ldots + W_{i_t}^{\ell_t},$ for $i_1, \ldots, i_t \in \{1, \ldots, K\}, \ell_1, \ldots, \ell_t \in \{1, \ldots, N-1\},\ t \in \{2, \ldots, K\}\}\}$ with equal probability $\frac{1}{N^K}$ irrespective of the message requirement. Therefore, from a given database's perspective, the a posteriori probability of the

user needing message j, upon receiving a query q from a user can be calculated by,

$$P(\theta = i|Q = q) = \frac{P(Q = q|\theta = i)P(\theta = i)}{\sum_{j=1}^{K} P(Q = q|\theta = j)P(\theta = j)} \tag{2.90}$$

$$= \frac{\frac{1}{N^K}P(\theta = i)}{\sum_{j=1}^{K} \frac{1}{N^K}P(\theta = j)} \tag{2.91}$$

$$= P(\theta = i) \tag{2.92}$$

which ensures that this scheme is private, since it implies that θ and Q are independent.

2.4.5 Example of Semantic PIR Scheme 2

2.4.5.1 Example 3: $N = 3, K = 3$, $L_1 = 400$ bits, $L_2 = 300$ bits and $L_3 = 100$ bits

Table 2.5 shows the query options that the user may use with probability $\frac{1}{27}$, to download W_1. Whenever a set of queries for the three databases is chosen with probability $\frac{1}{27}$, the required message is retrieved by subtracting the smaller sum from the larger sums, guaranteeing correctness.

The queries in the first block have zero side information, and retrieve the $N - 1 = 2$ parts of W_1 using $N - 1$ different databases. The second block uses W_2^1 as side information, and retrieve the two parts of W_1 (in terms of a sum of itself and side information) using the other two databases. The same procedure is carried out in blocks 3, 4 and 5, with W_2^1 replaced by W_2^2, W_3^1 and W_3^2. Last four blocks

of Table 2.5 use $W_2^i + W_3^j$ for $j \in 1, 2$ as side information and use sums of three elements ($W_1^k + W_2^i + W_3^j$ for $k = 1, 2$) to retrieve the two parts of W_1.

Probability	Database 1	Database 2	Database 3
$\frac{1}{27}$	W_1^1	W_1^2	ϕ
$\frac{1}{27}$	W_1^2	ϕ	W_1^1
$\frac{1}{27}$	ϕ	W_1^1	W_1^2
$\frac{1}{27}$	$W_1^1 + W_2^1$	$W_1^2 + W_2^1$	W_2^1
$\frac{1}{27}$	$W_1^2 + W_2^1$	W_2^1	$W_1^1 + W_2^1$
$\frac{1}{27}$	W_2^1	$W_1^1 + W_2^1$	$W_1^2 + W_2^1$
$\frac{1}{27}$	$W_1^1 + W_2^2$	$W_1^2 + W_2^2$	W_2^2
$\frac{1}{27}$	$W_1^2 + W_2^2$	W_2^2	$W_1^1 + W_2^2$
$\frac{1}{27}$	W_2^2	$W_1^1 + W_2^2$	$W_1^2 + W_2^2$
$\frac{1}{27}$	$W_1^1 + W_3^1$	$W_1^2 + W_3^1$	W_3^1
$\frac{1}{27}$	$W_1^2 + W_3^1$	W_3^1	$W_1^1 + W_3^1$
$\frac{1}{27}$	W_3^1	$W_1^1 + W_3^1$	$W_1^2 + W_3^1$
$\frac{1}{27}$	$W_1^1 + W_3^2$	$W_1^2 + W_3^2$	W_3^2
$\frac{1}{27}$	$W_1^2 + W_3^2$	W_3^2	$W_1^1 + W_3^2$
$\frac{1}{27}$	W_3^2	$W_1^1 + W_3^2$	$W_1^2 + W_3^2$
$\frac{1}{27}$	$W_1^1 + W_2^1 + W_3^1$	$W_1^2 + W_2^1 + W_3^1$	$W_2^1 + W_3^1$
$\frac{1}{27}$	$W_1^2 + W_2^1 + W_3^1$	$W_2^1 + W_3^1$	$W_1^1 + W_2^1 + W_3^1$
$\frac{1}{27}$	$W_2^1 + W_3^1$	$W_1^1 + W_2^1 + W_3^1$	$W_1^2 + W_2^1 + W_3^1$
$\frac{1}{27}$	$W_1^1 + W_2^2 + W_3^1$	$W_1^2 + W_2^2 + W_3^1$	$W_2^2 + W_3^1$
$\frac{1}{27}$	$W_1^2 + W_2^2 + W_3^1$	$W_2^2 + W_3^1$	$W_1^1 + W_2^2 + W_3^1$
$\frac{1}{27}$	$W_2^2 + W_3^1$	$W_1^1 + W_2^2 + W_3^1$	$W_1^2 + W_2^2 + W_3^1$
$\frac{1}{27}$	$W_1^1 + W_2^1 + W_3^2$	$W_1^2 + W_2^1 + W_3^2$	$W_2^1 + W_3^2$
$\frac{1}{27}$	$W_1^2 + W_2^1 + W_3^2$	$W_2^1 + W_3^2$	$W_1^1 + W_2^1 + W_3^2$
$\frac{1}{27}$	$W_2^1 + W_3^2$	$W_1^1 + W_2^1 + W_3^2$	$W_1^2 + W_2^1 + W_3^2$
$\frac{1}{27}$	$W_1^1 + W_2^2 + W_3^2$	$W_1^2 + W_2^2 + W_3^2$	$W_2^2 + W_3^2$
$\frac{1}{27}$	$W_1^2 + W_2^2 + W_3^2$	$W_2^2 + W_3^2$	$W_1^1 + W_2^2 + W_3^2$
$\frac{1}{27}$	$W_2^2 + W_3^2$	$W_1^1 + W_2^2 + W_3^2$	$W_1^2 + W_2^2 + W_3^2$

Table 2.5: The query table for the retrieval of W_1.

The rate achieved by this scheme when retrieving W_1 is,

$$R_1 = \frac{L_1}{\frac{1}{27}\left(3L_1 + 18(\frac{L_1}{2} \times 2 + \frac{L_2}{2}) + 6(\frac{L_1}{2} \times 2 + \frac{L_3}{2})\right)} \tag{2.93}$$

$$= \frac{L_1}{\frac{1}{27}(27L_1 + 9L_2 + 3L_3)} \tag{2.94}$$

$$=\frac{400}{\frac{1}{27}(27\times 400+9\times 300+3\times 100)}=\frac{36}{46} \tag{2.95}$$

The rate achieved by this scheme when retrieving W_2 is,

$$R_2=\frac{L_2}{\frac{1}{27}\left(3L_2+18\times 3\times \frac{L_1}{2}+6\times (L_2+\frac{L_3}{2})\right)} \tag{2.96}$$

$$=\frac{L_2}{\frac{1}{27}(27L_1+9L_2+3L_3)} \tag{2.97}$$

$$=\frac{300}{\frac{1}{27}(27\times 400+9\times 300+3\times 100)}=\frac{27}{46} \tag{2.98}$$

The rate achieved by this scheme when retrieving W_3 is,

$$R_3=\frac{L_3}{\frac{1}{27}\left(3L_3+18\times 3\times \frac{L_1}{2}+6\times 3\times \frac{L_2}{2}\right)} \tag{2.99}$$

$$=\frac{L_3}{\frac{1}{27}(27L_1+9L_2+3L_3)} \tag{2.100}$$

$$=\frac{100}{\frac{1}{27}(27\times 400+9\times 300+3\times 100)}=\frac{9}{46} \tag{2.101}$$

The overall message retrieval rate for this example is,

$$R=\frac{\mathbb{E}[L]}{\mathbb{E}[D]}=p_1\frac{L_1}{D}+p_2\frac{L_2}{D}+p_3\frac{L_3}{D} \tag{2.102}$$

$$=p_1R_1+p_2R_2+p_3R_3=\frac{36}{46}p_1+\frac{27}{46}p_2+\frac{9}{46}p_3 \tag{2.103}$$

This matches the semantic PIR capacity expression in Theorem 2.1,

$$C=\left(\frac{L_1}{\mathbb{E}[L]}+\frac{1}{N}\frac{L_2}{\mathbb{E}[L]}+\frac{1}{N^2}\frac{L_3}{\mathbb{E}[L]}\right)^{-1} \tag{2.104}$$

$$=(400p_1 + 300p_2 + 100p_3)\left(400 + \frac{300}{3} + \frac{100}{9}\right)^{-1} \tag{2.105}$$

$$=\frac{36}{46}p_1 + \frac{27}{46}p_2 + \frac{9}{46}p_3 \tag{2.106}$$

The classical PIR capacity for this case with equal priors is,

$$C = \left(1 + \frac{1}{N} + \frac{1}{N^2}\right)^{-1} = \left(1 + \frac{1}{3} + \frac{1}{9}\right)^{-1} = \frac{9}{13} \tag{2.107}$$

The semantic PIR capacity in (2.106) exceeds the classical PIR capacity in (2.107) when

$$\frac{36}{46}p_1 + \frac{27}{46}p_2 + \frac{9}{46}p_3 > \frac{9}{13} \tag{2.108}$$

which is equivalent to

$$p_1 + \frac{2}{3}p_2 > \frac{11}{13} \tag{2.109}$$

Remark 2.10 *We note again that the rate calculation presented here for the semantic PIR capacity takes into consideration the zero-padding needed to be added to the shorter message block in order to perform bit-wise message addition for any query realization. The classical PIR capacity expression in* (2.107) *assumes that all messages are of equal size and hence the extra zero-padding is not reflected in that expression. Hence, the actual rate of classical PIR scheme is indeed less than the reported PIR capacity if the messages are of unequal size.*

Remark 2.11 *The second scheme presented above is an extension to more than two databases of the path-based scheme presented in [49]. It is also similar to the scheme provided in [52], except for the fact that the above scheme has database symmetry as opposed to the scheme presented in [52].*

2.5 Converse Proof

In this section, we present the converse proof for Theorem 2.1. This proof is a slight modification of the converse proof presented in [3]. The central intuition of our proof is the fact that the expected length of the answer string generated by a given database should remain the same, irrespective of the identity of the desired message as a consequence of the privacy constraint. The major difference of our proof compared to [3] is the handling of the non-equal message sizes.

We begin the proof of Theorem 2.1 by the definition of message retrieval rate,

$$R = \frac{\mathbb{E}[L]}{\mathbb{E}[D]} \tag{2.110}$$

We choose some permutation $\{i_1, \ldots, i_K\}$ as an arbitrary order of the messages. The denominator of (2.110) can be expanded as follows,

$$\mathbb{E}[D] = \sum_{i=1}^{K} p_i (H(A_1^{[i]}) + \cdots + H(A_N^{[i]})) \tag{2.111}$$

$$= H(A_1^{[i_1]}) + \cdots + H(A_N^{[i_1]}) \tag{2.112}$$

Following the same steps in the converse proof of [3] $\mathbb{E}[D]$ can be lower bounded as,

$$\mathbb{E}[D] \geq L_{i_1} + H(A_n^{[i_2]}|Q_n^{[i_2]}, W_{i_1}) \quad n = 1, \ldots, N \tag{2.113}$$

By summing all N inequalities corresponding to (2.113) and repeating the previous steps for W_{i_2} (with conditioning on W_{i_1}) leads to,

$$N\mathbb{E}[D] \geq NL_{i_1} + L_{i_2} + H(A_n^{[i_3]}|Q_n^{[i_3]}, W_{i_1}, W_{i_2}) \tag{2.114}$$

for $n = 1, \ldots, N$. By summing the corresponding inequalities and continuing with the same procedure for $W_{i_3}, \ldots, W_{i_K}$ yields,

$$\begin{aligned} N^{K-1}\mathbb{E}[D] \geq\ & N^{K-1}L_{i_1} + N^{K-2}L_{i_2} + \cdots + NL_{i_{K-1}} \\ & + I(W_{i_K}; A_1^{[i_K]}, \ldots, A_N^{[i_K]}|Q_1^{[i_K]}, \ldots, Q_N^{[i_K]}, W_{i_1}, \ldots, W_{i_{K-1}}) \end{aligned} \tag{2.115}$$

and therefore, we have,

$$\mathbb{E}[D] \geq L_{i_1} + \frac{1}{N}L_{i_2} + \cdots + \frac{1}{N^{K-2}}L_{i_{K-1}} + \frac{1}{N^{K-1}}L_{i_K} \tag{2.116}$$

which further gives,

$$R \leq \left(\frac{L_{i_1}}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_{i_2}}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_{i_K}}{\mathbb{E}[L]}\right)^{-1} \tag{2.117}$$

The upper bound in (2.117) holds for any permutation $\{i_1, \ldots, i_K\}$, hence, the

tightest upper bound can be obtained by minimizing over all permutations[6]. Consequently,

$$R \leq \min_{\{i_1,\ldots,i_K\}} \left(\frac{L_{i_1}}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_{i_2}}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_{i_K}}{\mathbb{E}[L]} \right)^{-1} \tag{2.118}$$

Since the messages are ordered such that $L_1 \geq L_2 \geq \cdots \geq L_K$, the minimum upper bound is attained at $\{i_1, \ldots, i_K\} = \{1, \ldots, K\}$ as it gives the largest number to the largest coefficient in the lower bound on the download cost. Thus,

$$R \leq \left(\frac{L_1}{\mathbb{E}[L]} + \frac{1}{N}\frac{L_2}{\mathbb{E}[L]} + \cdots + \frac{1}{N^{K-1}}\frac{L_K}{\mathbb{E}[L]} \right)^{-1} \tag{2.119}$$

completing the converse proof.

2.6 Extensions of Semantic PIR

2.6.1 Semantic PIR from MDS-coded Databases

In this section, we present a complete characterization of the capacity of semantic PIR from MDS-coded databases, along with an optimal scheme. The optimal scheme is an extension of the scheme presented in [9]. We consider an (N, M) MDS coded distributed storage system containing K independent messages. The messages are allowed to have different semantics (lengths and prior probabilities). Each message W_i is represented as a matrix in $\mathbb{F}_q^{L_i \times M}$, where the elements of the matrix are uniformly and randomly chosen from $\mathbb{F}_q$. The generator matrix of the (N, M)

[6]Note that the order does not matter in the case of equal message lengths in [3].

code is $H = [h_1, \ldots, h_N]$, where $h_i \in \mathbb{F}_q^M$, $i \in [N]$. The MDS property implies that any combination of up to M columns of H is linearly independent.

Let the jth row of W_i be denoted by $W_j^{[i]}$. Each database n, $n \in [N]$ stores $W_j^{[i]} h_n$ for $j \in [L_i]$, $i \in [K]$. The objective is to download a required message without revealing its index to any of the databases. In order to retrieve W_i, user sends query $Q_n^{[i]}$ to database n, $n \in [N]$ and receives the answer $A_n^{[i]}$ which is a deterministic function of the contents of the database and $Q_n^{[i]}$. The correctness and privacy conditions are the same as (2.5) and (2.6) respectively, and the rate is calculated by,

$$R = \frac{M\mathbb{E}[L]}{\mathbb{E}[D]} \tag{2.120}$$

Theorem 2.2 gives the exact PIR capacity for the semantic PIR problem.

Theorem 2.2 *The capacity of semantic PIR with (N, M) MDS-coded databases with N databases, K messages, message sizes ML_i (arranged as $L_1 \geq L_2 \geq \ldots \geq L_K$) and prior probabilities p_i is given by,*

$$C = \left(\frac{L_1}{\mathbb{E}[L]} + \left(\frac{M}{N}\right) \frac{L_2}{\mathbb{E}[L]} + \cdots + \left(\frac{M}{N}\right)^{K-1} \frac{L_K}{\mathbb{E}[L]} \right)^{-1} \tag{2.121}$$

where $\mathbb{E}[L] = \sum_{i=1}^{K} p_i L_i$.

The achievable scheme is an extension to the first scheme presented in Section 2.4.1. The steps of the achievable scheme are as follows. Assume that the user requires to download W_j.

1. **Message indexing:** Assign indices to messages in the descending order message sizes, i.e., $L_1 \geq L_2 \geq \ldots \geq L_K$. Permute the rows of all messages randomly and independently, privately from the databases.

2. **Single blocks:** Using (2.129), download υ_j different coded bits of W_j from each database. Download υ_i coded bits of W_i, $i \neq j$ from each database such that the coded bits of M different databases correspond to the same row of W_i. This is required to decode the rows of W_i that are used as side information. Therefore, $N\upsilon_i$ coded bits of W_i, $i \neq j$ are downloaded in this step, that belong to $\frac{N\upsilon_i}{M}$ different rows of W_i.

3. **Sums of two elements:** There are two types of blocks in this step. The first block is the sums involving bits of the desired message, W_j , and the other block is the sums that do not have any bits from W_j. In the first block, make use of the side information (singles corresponding to W_i, $i \neq j$) downloaded in the previous step. Consider a 2-sum corresponding to coded bits of W_j, W_i, $i \neq j$. Download $\left(\frac{N}{M} - 1\right) \min\{\upsilon_i, \upsilon_j\}$ 2-sums of the form $(W_{r_n}^{[j]} + W_{s_n}^{[i]})h_n$ from database n, $n \in [N]$ where $W_{r_n}^{[j]}$ are new rows of W_j and $W_{s_n}^{[i]}$ are already decoded rows of W_i in the previous step. Note that the set of M databases that were used to decode $W_{s_n}^{[i]}$ in the previous step does not include database n. The second block of 2-sums contains coded bits corresponding to W_{i_1} and W_{i_2}, where $i_1 \neq i_2 \neq j$. Download $\left(\frac{N}{M} - 1\right) \min\{\upsilon_{i_1}, \upsilon_{i_2}\}$ 2-sums of the form $(W_{t_n}^{[i_1]} + W_{v_n}^{[i_2]})h_n$ from database n, $n \in [N]$ where $W_{t_n}^{[i_1]}$ and $W_{v_n}^{[i_2]}$ are new rows of W_{i_1} and W_{i_2}. Note that coded bits corresponding to the same pair of

rows (t_n, v_n) needs to be downloaded from M different databases in order to correctly decode the side information $W_{t_n}^{[i_1]} + W_{v_n}^{[i_2]}$. Thus, the second block contains $N\left(\frac{N}{M}-1\right)\min\{v_{i_1}, v_{i_2}\}$ coded 2-sums corresponding to W_{i_1} and W_{i_2} belonging to $\frac{N\left(\frac{N}{M}-1\right)\min\{v_{i_1},v_{i_2}\}}{M}$ different rows.

4. **Sums of ℓ elements:** There are two types of blocks similar to sums of two. The first block contains queries of the form $(W_r^{[j]} + W_{r_1}^{[i_1]} + \ldots + W_{r_{\ell-1}}^{[i_{\ell-1}]})h_n$, $i_1 \neq, \ldots, \neq i_{\ell-1} \neq j$, where $W_r^{[j]}$ is a new row of W_j and $W_{r_1}^{[i_1]} + \ldots + W_{r_{\ell-1}}^{[i_{\ell-1}]}$ is an already decoded $(\ell-1)$-sum from the previous step. For a given $(\ell-1)$-tuple $(i_1, \ldots, i_{\ell-1})$, download $\left(\frac{N}{M}-1\right)^{\ell-1} v_{\min\{j,i_1,\ldots,i_{\ell-1}\}}$ such ℓ-sums from each database. The second block contains queries of the form $(W_{t_1}^{[i_1]} + \ldots + W_{t_\ell}^{[i_\ell]})h_n$, $i_1 \neq \ldots \neq i_\ell \neq j$, where $W_{t_1}^{[i_1]}, \ldots, W_{t_\ell}^{[i_\ell]}$ are new rows of $W_{i_1}, \ldots, W_{i_\ell}$. Download $\left(\frac{N}{M}-1\right)^{\ell-1} v_{\min\{i_1,\ldots,i_\ell\}}$ such ℓ-sums from each database such that the coded bits corresponding to a given ℓ-tuple of rows $(r_1, \ldots, r_\ell)$ is downloaded from M different databases. A total of $N\left(\frac{N}{M}-1\right)^{\ell-1} v_{\min\{i_1,\ldots,i_\ell\}}$ coded bits of this form will be downloaded corresponding to $\frac{\left(\frac{N}{M}-1\right)^{\ell-1} v_{\min\{i_1,\ldots,i_\ell\}}}{M}$ different ℓ-tuples of rows of $W_{i_1}, \ldots, W_{i_\ell}$.

5. **Repeat the process** up to sums of K elements.

6. **Query repetition:** To decode each row of W_j, repeat the above process M times, while shifting the queries that contain rows of W_j to its neighboring database and by choosing new sets of rows of W_i, $i \in \{1, \ldots, K\}$, $i \neq j$ in each repetition. The M different linear combinations of each row of W_j allow us to correctly decode W_j.

The achievable rate of the above scheme is calculated as follows. First, note that the download cost remains the same irrespective of the message requirement in order to guarantee privacy. Therefore, the $\mathbb{E}[D]$ term in (2.120) is calculated by summing the number of downloads in each step of the scheme. Within one round of queries, there are $\sum_{i=1}^{K} N\upsilon_i$ singletons and $N\left(\frac{N}{M}-1\right)^{\ell-1}\sum_{i=\ell}^{K}\binom{i-1}{\ell-1}\upsilon_i$ sums of ℓ-elements. Therefore,

$$\frac{\mathbb{E}[D]}{M} = \sum_{i=1}^{K} N\upsilon_i + \sum_{\ell=2}^{K}\sum_{i=\ell}^{K} N\left(\frac{N}{M}-1\right)^{\ell-1}\upsilon_i\binom{i-1}{\ell-1} \tag{2.122}$$

$$= N\left[\sum_{i=1}^{K}\upsilon_i + \sum_{\ell=2}^{K}\upsilon_\ell\sum_{i=2}^{\ell}\left(\frac{N}{M}-1\right)^{i-1}\binom{\ell-1}{i-1}\right] \tag{2.123}$$

$$= M\left[\frac{N}{M}\upsilon_1 + \sum_{\ell=2}^{K}\upsilon_\ell\left(\frac{N}{M}\right)^{\ell}\right] = M\sum_{\ell=1}^{K}\left(\frac{N}{M}\right)^{\ell}\upsilon_\ell \tag{2.124}$$

For the $\mathbb{E}[L]$ term in (2.120), we sum the number of useful bits downloaded in each step of the scheme. Based on the scheme described above, $N\upsilon_j$ rows of W_j are retrieved as singletons, $N\left(\frac{N}{M}-1\right)^{\ell-1}\binom{j-1}{\ell-1}\upsilon_j$ rows of W_j are retrieved as ℓ-sums with W_j being the shortest message and $N\left(\frac{N}{M}-1\right)^{\ell-1}\binom{i-2}{\ell-2}\upsilon_i$ rows of W_j are retrieved as ℓ-sums with W_i, $i \neq j$ being the shortest message in the sum. Denoting U_j as the total number of useful bits downloaded, the number of rows of W_j retrieved is calculated by,

$$\begin{aligned}\frac{U_j}{M} =& N\upsilon_j + \sum_{\ell=2}^{j} N\left(\frac{N}{M}-1\right)^{\ell-1}\binom{j-1}{\ell-1}\upsilon_j \\ &+ \sum_{\ell=2}^{j}\sum_{i=j+1}^{K} N\left(\frac{N}{M}-1\right)^{\ell-1}\binom{i-2}{\ell-2}\upsilon_i\end{aligned}$$

$$+\sum_{\ell=j+1}^{K}\sum_{i=\ell}^{K} N\left(\frac{N}{M}-1\right)^{\ell-1}\binom{i-2}{\ell-2}\upsilon_i \tag{2.125}$$

$$=N\upsilon_j\sum_{\ell=0}^{j-1}\gamma^{\ell}\binom{j-1}{\ell}+N\upsilon_{j+1}\gamma\sum_{\ell=0}^{j-1}\gamma^{\ell}\binom{j-1}{\ell}$$

$$+N\upsilon_{j+2}\gamma\sum_{\ell=0}^{j}\gamma^{\ell}\binom{j}{\ell}+\cdots+N\upsilon_K\gamma\sum_{\ell=0}^{K-2}\gamma^{\ell}\binom{K-2}{\ell} \tag{2.126}$$

$$=M\left[\left(\frac{N^j}{M^j}\right)\upsilon_j+\gamma\sum_{i=j+1}^{K}\left(\frac{N}{M}\right)^{i-1}\upsilon_i\right] \tag{2.127}$$

where $\gamma=\frac{N}{M}-1$. Thus, the subpacketization of W_j is defined as $\frac{U_j}{M}$, which represents the number of rows of W_j, that can be retrieved by a single use of the scheme. Since the total number of rows of W_j, $j\in\{1,\ldots,K\}$ have to be a common multiple of their own subpacketizations,

$$L_j=\alpha\frac{U_j}{M},\quad j\in\{1,\ldots,K\} \tag{2.128}$$

for some $\alpha\in\mathbb{N}$. Solving (2.127) and (2.128) for $\upsilon_1,\ldots,\upsilon_K$ gives,

$$\begin{bmatrix}\upsilon_1\\ \upsilon_2\\ \vdots\\ \upsilon_K\end{bmatrix}=\frac{1}{M\alpha}\begin{bmatrix}\frac{M}{N} & -\left(\frac{M}{N}\right)^2\gamma & \ldots & -\left(\frac{M}{N}\right)^K\gamma\\ 0 & \left(\frac{M}{N}\right)^2 & \ldots & -\left(\frac{M}{N}\right)^K\gamma\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \ldots & \left(\frac{M}{N}\right)^K\end{bmatrix}\begin{bmatrix}L_1\\ L_2\\ \vdots\\ L_K\end{bmatrix} \tag{2.129}$$

In order for the values of $\upsilon_i, i\in\{1,\ldots,K\}$ to be integers, this scheme requires each L_i to be a multiple of N^i. Here, α is the greatest common divisor (gcd) of the elements of the vector resulting from multiplying the matrix and the vector on

the right side of (2.129). This allows the shortest subpacketization levels for all messages.

The total and useful numbers of bits downloaded (in (2.124) and (2.127), respectively) are both within one subpacketization level. These downloads are repeated α times to download the entire message. Thus, the achievable rate is given by,

$$R = \frac{M\mathbb{E}[L]}{\mathbb{E}[D]} = \frac{M\sum_{i=1}^{K} p_i L_i}{\alpha M^2 \sum_{i=1}^{K} \frac{N^i}{M^i} \upsilon_i} \tag{2.130}$$

$$= \frac{\mathbb{E}[L]}{\alpha M \frac{1}{M\alpha} \sum_{i=1}^{K} \frac{N^i}{M^i} \left[\left(\frac{M^i}{N^i}\right) L_i - \left(\frac{N}{M} - 1\right) \sum_{t=i+1}^{K} \left(\frac{M^t}{N^t}\right) L_t \right]} \tag{2.131}$$

$$= \frac{\mathbb{E}[L]}{\sum_{i=1}^{K} \left[L_i - \left(\frac{N}{M} - 1\right) \sum_{t=i+1}^{K} \left(\frac{M^{t-i}}{N^{t-i}}\right) L_t \right]} \tag{2.132}$$

$$= \frac{\mathbb{E}[L]}{L_1 + L_2 \left(\frac{M}{N}\right) + \sum_{i=3}^{K} L_i \left[1 - \left(1 - \frac{M^{i-1}}{N^{i-1}}\right)\right]} \tag{2.133}$$

$$= \left(\frac{L_1}{\mathbb{E}[L]} + \left(\frac{M}{N}\right) \frac{L_2}{\mathbb{E}[L]} + \cdots + \left(\frac{M}{N}\right)^{K-1} \frac{L_K}{\mathbb{E}[L]} \right)^{-1} \tag{2.134}$$

A given database always receives queries of the same type (i.e., $\left(\frac{N}{M} - 1\right)^{\ell-1} \upsilon_{\min\{i_1,\ldots,i_\ell\}}$, $\forall \{i_1, \ldots, i_\ell\} \subset [K]$, ℓ-sums for $\ell \in \{1, \ldots, K\}$) irrespective of the message requirement. According to the query generation procedure, no bit of any message is requested from a given database more than once as a singleton or as an element of a sum. Therefore, a proof similar to what is presented in Section 2.4.1.2 is used to show that this scheme is private.

The above scheme can be alternatively described using the same ideas presented in Section 2.4.3. The alternative description is as follows. Database n,

$n \in \{1, \ldots, N\}$ contains coded bits corresponding to each row of W_i, $i \in \{1, \ldots, K\}$ given by $W_r^{[i]} h_n$, $r \in \{1, \ldots, L_i\}$. Therefore, each database stores L_i coded bits of W_i, where $L_1 \geq L_2 \geq \ldots \geq L_K$. Considering the first L_K coded bits of all messages, the classical MDS-coded PIR scheme in [9] is applied as the first step of the scheme. Then, apply the classical coded PIR scheme using the next $L_{K-1} - L_K$ coded bits of messages W_1 to W_{K-1}. In general, in the ℓth step, the classical coded PIR scheme needs to be applied on the $L_{K-\ell+1} - L_{K-\ell+2}$ coded bits of W_1 to $W_{K-\ell+1}$. The complete scheme should be used irrespective of the message requirement.

The alternative description differs from the main description in subpacketization, and in the scheme used within a subpacket as explained in Section 2.4.3. However, the two descriptions are equivalent when considering the entire retrieval process (all subpackets). The rate achieved by the alternative scheme is given by,

$$\begin{aligned} R =& \mathbb{E}[L] / \Bigg(L_K \left(1 + \frac{M}{N} + \ldots + \frac{M^{K-1}}{N^{K-1}}\right) \\ & + (L_{K-1} - L_K) \left(1 + \frac{M}{N} + \ldots + \frac{M^{K-2}}{N^{K-2}}\right) + \ldots + L_1 - L_2 \Bigg) \end{aligned} \tag{2.135}$$

which is the same as (2.134). A converse proof similar to what is presented in Section 2.5 with the ideas of [9] is used to prove an upper bound on the retrieval rate of semantic PIR from MDS-coded databases, which is the same as (2.134). This proves the capacity expression in (2.121).

2.6.2 Semantic PIR from Colluding Databases

In this section, we present a complete characterization of the capacity of semantic PIR from colluding databases, along with an optimal scheme. This is an extension of the results presented in [4]. We consider K independent messages (W_i, $i \in \{1, \ldots, K\}$) with arbitrary lengths L_i and prior probabilities p_i, stored in N replicated databases. Out of the N databases, any subset up to T databases are allowed to collude. The objective here is to download a user-required message without revealing its index to any T-colluding databases.

Theorem 2.3 *The capacity of semantic PIR from colluding databases, with K messages, message lengths L_i (arranged as $L_1 \geq L_2 \geq \ldots \geq L_K$), prior probabilities p_i and N databases out of which any T are colluding, is given by,*

$$C = \left(\frac{L_1}{\mathbb{E}[L]} + \frac{L_2}{\mathbb{E}[L]} \left(\frac{T}{N}\right) + \ldots + \frac{L_K}{\mathbb{E}[L]} \left(\frac{T}{N}\right)^{K-1} \right)^{-1} \tag{2.136}$$

where $\mathbb{E}[L] = \sum_{i=1}^{K} p_i L_i$.

The optimal scheme is an extension of the scheme presented in Section 2.4.1. The scheme is as follows. Assume that the required message is W_j. Once the messages are indexed based on the decreasing order of lengths, the user needs to generate a set of linear combinations of the message indices given by,

$$x_j = S_j W_j \tag{2.137}$$

where S_j is a random full rank matrix drawn uniformly and independently from all such matrices in $\mathbb{F}_q^{\ell_j \times \ell_j}$ where ℓ_i is the subpacketization of W_j. For each W_m, $m \neq j$, let m_t denote the number of t-sums in the scheme involving W_m but not W_j. Let $m_{t,j}$ be the number of t-sums in the scheme involving both W_m and W_j.[7] Then, the linear combinations of W_i, $i \in [K]$, $i \neq j$ are generated by,

$$\text{first } (m_1 + m_{2,j}) \text{ bits of } x_i$$

$$= \text{MDS}_{(m_1+m_{2,j})\times m_1} S_i[(1 : m_1), :]W_i \tag{2.138}$$

$$\text{next } (m_2 + m_{3,j}) \text{ bits of } x_i$$

$$= \text{MDS}_{(m_2+m_{3,j})\times m_2} S_i[(m_1 + 1 : m_1 + m_2), :]W_i \tag{2.139}$$

$$\vdots$$

$$\text{last } (m_{K-1} + m_{K,j}) \text{ bits of } x_i$$

$$= \text{MDS}_{(m_{K-1}+m_{K,j})\times m_{K-1}} S_i[(\ell_i - m_{K-1} + 1 : \ell_i), :]W_i \tag{2.140}$$

where S_i, $i \in \{1, \ldots, K\}$ are random full rank matrices of $\mathbb{F}_q^{\ell_i \times \ell_i}$ and $\text{MDS}_{a\times b}$ are globally known generator matrices of (a, b) MDS-codes. The first step of the scheme is to calculate υ_i, $i \in \{1, \ldots, K\}$ using (2.129) with M replaced by T. Then, download υ_i, $i \in \{1, \ldots, K\}$ bits of each x_i from each database. Next, from each database, download $\left(\frac{N}{T} - 1\right) \upsilon_{\min\{i_1,\ldots,i_t\}}$ t-sums, $t \in \{2, \ldots, K\}$ involving new bits of $x_{i_1}, \ldots, x_{i_t}$, $\forall \{i_1, \ldots, i_t\} \subset \{1, \ldots, K\}$. This completes the scheme.

For a given t-sum of the form $x_{i_1}(\cdot) + \ldots + x_{i_t}(\cdot)$ with $i_1 \geq i_2 \geq \ldots \geq i_t$, which

[7] The values of m_t and $m_{t,j}$ for $t \in \{1, \ldots, K\}$ are immediate from the steps of scheme which are described later. These values do not depend on the linear combinations.

does include any bit of W_j, let the generator matrix corresponding to each element x_{i_k} in the sum be denoted by G_{i_k}. Then, each G_{i_k} must satisfy, $G_{i_k} = \begin{bmatrix} G_{i_{k+1}} \\ \dots \\ X \end{bmatrix}$, $k \in \{1, \dots, t\}$ where X denotes the set of extra rows in the larger generator matrix. This is required for interference alignment. The proof of privacy in [4] applies to this scheme as well. The fact that the required message is coded differently, in a non redundant manner, ensures the correctness of the scheme as explained in [4].

The optimal scheme above can be alternatively described as follows. In each database, segment the set of messages into K partitions, such that the first segment contains the first L_K bits of all K messages, the second segment contains the next set of $L_{K_1} - L_K$ bits of messages W_1 to W_{K-1} and so on. Then, apply the classical colluded PIR scheme in [4] to the 1) the first segment with K messages, 2) the second segment with $K-1$ messages, 3) the third segment with $K-2$ messages, and so on. Make sure that the complete scheme is used irrespective of the desired message for privacy. The achievable rate of the scheme is equal to the capacity in (2.136). The converse is proved using similar ideas provided in the converse proofs of Section 2.5 and [4].

2.7 Conclusions

In this chapter, we introduced the problem of semantic PIR. In this problem, the stored messages are allowed to have non-uniform popularities, which is captured via an a priori probability distribution p_i, $i \in [K]$, and heterogeneous sizes L_i, $i \in$

$[K]$. We derived the exact semantic PIR capacity as a function of $\{L_i\}_{i=1}^K$ and the expected message size $\mathbb{E}[L]$. The result implies that the semantic PIR capacity is equal to the classical PIR capacity if all messages have equal sizes $L_i = L$ for all $i \in [K]$. We derived a necessary and sufficient condition for the semantic PIR capacity to exceed the classical PIR capacity. In particular, we showed that if the longer messages are retrieved more often, there is a strict retrieval rate gain from exploiting the message semantics.[8] Furthermore, we proved that for all message sizes and priors, the semantic PIR capacity exceeds the achievable rate of classical PIR with zero-padding, which zero-pads all messages to equalize their sizes.

To that end, we proposed two achievable schemes for achieving the semantic PIR capacity. The first one has a deterministic query structure. We have proposed a systematic way of calculating the needed subpacketization levels for the messages. We also provided an alternative description to this scheme which implements the classical PIR scheme in a segmented manner. The similarities and differences between the two descriptions were also discussed. The second scheme has a stochastic query structure, where the user picks one query structure at random from an ensemble of structures. The first scheme has the advantage of having a fixed download cost for all messages for all query structures unlike the stochastic scheme, which has the same expected download cost. Nevertheless, the first scheme suffers from exponential subpacketization levels in contrast to the linear counterpart in the stochastic scheme. We derived a matching converse that extends the converse scheme of [3] to

[8] This does not necessarily mean that $p_1 \geq p_2 \geq \ldots \geq p_K$. It essentially means that the $\mathbb{E}[L]$ should be large enough such that (2.11) is satisfied.

take into account the heterogeneous message sizes and prior probabilities. Finally, the extensions of semantic PIR to coded databases and colluding databases were analyzed separately, where the complete characterizations of the capacities of the two cases were presented along with the corresponding optimal schemes.

CHAPTER 3

Private Read-Update-Write (PRUW)

3.1 Introduction

In this chapter, we investigate the problem of PRUW. In PRUW, a user reads a specific section of a data storage system, updates it, and writes back the updates to the same/different section in the storage system without revealing the section indices or the values of the updates. PRUW has two main applications, namely private FSL and private FL with sparsification. In this chapter, we study PRUW in relation to private FSL. In FSL, a machine learning model is divided into M submodels and stored in N non-colluding databases, from which a given user privately reads, updates and writes back an arbitrary submodel. We consider information-theoretic privacy of the updated submodel index as well as the values of the updates. We provide an efficient PRUW scheme which achieves a lower total communication cost compared to the state-of-the-art. The formulation of basic PRUW presented in this chapter is the building block of the other variants considered in subsequent chapters.

3.2 Problem Formulation

We consider N non-colluding databases storing M independent submodels. Initially, each submodel consists of random symbols picked from a finite field $\mathbb{F}_q$, such that,

$$H(W_k^{[0]}) = L, \quad k \in \{1, \ldots, M\}, \tag{3.1}$$

$$H(W_1^{[0]}, \ldots, W_M^{[0]}) = \sum_{k=1}^{M} H(W_k^{[0]}) = LM, \tag{3.2}$$

where $W_k^{[0]}$ is the initial version of the kth submodel and L is the length of a submodel. At any given time instance, a single user reads, updates and writes a single submodel of interest, while keeping the submodel index and the value of the update private. The submodels are generated in such a way that any given user is equally probable to update any given submodel at a given time instance. The process of updating consists of two phases, namely, the reading phase where the user downloads the required submodel and the writing phase where the user uploads the incremental update back to the databases.

In the reading phase, the user sends queries to the databases to download the required submodel. These queries are deterministic functions of the user-required submodel index and random noise generated by the user, i.e.,

$$H(Q_1^{[t]}, \ldots, Q_N^{[t]} | \theta^{[t]}, Z) = 0, \tag{3.3}$$

where $Q_n^{[t]}$, $n \in \{1, \ldots, N\}$ are the queries sent by the user to the databases at time

t, $\theta^{[t]}$ is the user-required submodel index at time t, and Z represents the random noise used to determine the queries.

The user (at time t) has no prior information on the submodels contained in the databases. Therefore, the queries sent by the user at time t to the databases in the reading phase are independent of the existing submodels,

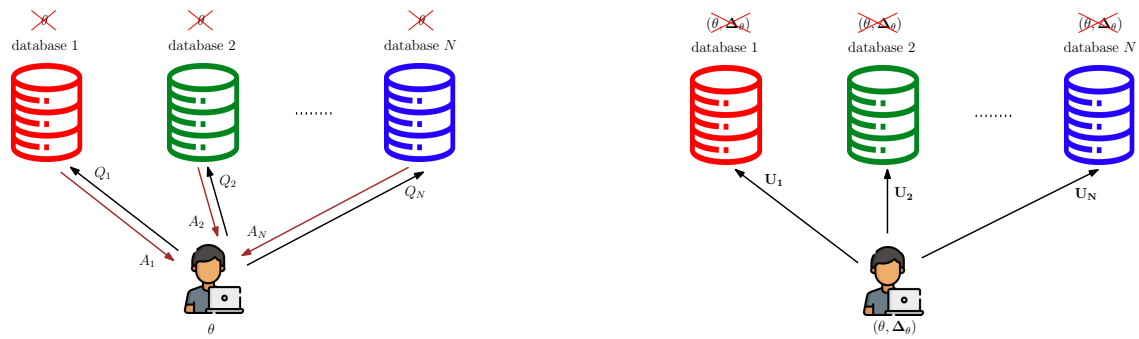
$$I(Q_1^{[t]}, \ldots, Q_N^{[t]}; W_1^{[t-1]}, \ldots, W_M^{[t-1]}) = 0, \quad t \in \mathbb{Z}^+, \tag{3.4}$$

where $W_k^{[t-1]}$, $k \in \{1, \ldots, M\}$ are the existing versions of the submodels (before updating) at time t. After receiving the queries, each database generates an answer and sends it back to the user. This answer is a function of its existing storage and the query received,

$$H(A_n^{[t]} | Q_n^{[t]}, S_n^{[t-1]}) = 0, \quad n \in \{1, \ldots, N\}, \tag{3.5}$$

where $A_n^{[t]}$ is the answer sent by database n at time t and $S_n^{[t-1]}$ is the existing storage (before updating) of database n at time t.

In the writing phase, the user sends information on the updates of the submodel to each database. Any PRUW scheme contains a specific mechanism that privately places these updates at correct positions in each database, since the submodel index and the value of the update are kept private from the databases. The information sent by the user to the databases in the writing phase at time t is a function of the generated updates, updating submodel index, and random noise



(a) Reading phase. (b) Writing phase.

Figure 3.1: A user reads a submodel, updates it, and writes it back to the databases.

generated by the user at time t, i.e.,

$$H(U_1^{[t]}, \ldots, U_N^{[t]} | \Delta_\theta^{[t]}, \theta^{[t]}, \bar{Z}) = 0, \tag{3.6}$$

where $U_n^{[t]}$ is the information on the updates sent by the user to database n at time t, $\Delta_\theta^{[t]}$ is the update generated by the user for submodel $\theta^{[t]}$ and $\bar{Z}$ is random noise generated by the user. Each database calculates an incremental update based on all information received by the user at time t, and adds it to the existing storage to obtain the updated storage.

Any information that is communicated in both phases takes place only between a single user and the system of databases. Users that update the model at different time instances do not communicate with each other. The problem is designed to study the PRUW procedure involving a single user at a given time instance. The same process is independently carried out at each time instance with different users. The system model is illustrated in Figure 3.1.

Next, we formally define the privacy, security and correctness conditions under

which a PRUW setting operates.

Privacy of the submodel index: No information on the indices of the submodels updated by any given user up to time t is allowed to leak to any of the databases with the availability of all storages, queries and updates up to time t. That is, for each database n, $n \in \{1, \ldots, N\}$,

$$I(\theta^{[1:t]}; Q_n^{[1:t]}, U_n^{[1;t]}, S_n^{[0:t]}) = 0, \quad t \in \mathbb{N}, \tag{3.7}$$

where $\theta^{[1:t]} = (\theta^{[1]}, \ldots, \theta^{[t]})$ are the indices of the submodels updated by the user at time instances 1 to t. Similarly, $Q_n^{[1:t]}$, $S_n^{[0:t]}$ and $U_n^{[1:t]}$ represent the queries, storages and information on updates communicated between the user and database n at corresponding time instances indicated in square brackets.[1] [2]

Privacy of the values of updates: No information on the values of the updates up to time t, i.e., $\Delta_\theta^{[1:t]}$ is allowed to leak to any of the databases with all data up to time t. That is, for each database n, $n \in \{1, \ldots, N\}$,

$$I(\Delta_\theta^{[1:t]}; Q_n^{[1:t]}, U_n^{[1:t]}, S_n^{[0:t]}) = 0, \quad t \in \mathbb{N}. \tag{3.8}$$

Security of the stored data: No information on the parameters of submodels up to time t is allowed to leak to any of the databases with all data up to time t.

[1] The notation $[1:t]$ represents all integers from 1 to t.

[2] Note that users start downloading/uploading information starting from time $t = 1$, while the storage is defined starting from time $t = 0$, from which the user downloads at time $t = 1$.

That is, for each database n, $n \in \{1, \ldots, N\}$,

$$I(W_{1:M}^{[0:t]}; S_n^{[0:t]}, Q_n^{[1:t]}, U_n^{[1:t]}) = 0, \quad t \in \mathbb{Z}_0^+. \tag{3.9}$$

Correctness in the reading phase: In the reading phase, the user must be able to correctly decode the required submodel using the queries sent and the answers received from all databases. That is,

$$H(W_\theta^{[t-1]} | Q_{1:N}^{[t]}, A_{1:N}^{[t]}, \theta^{[t]}) = 0, \quad t \in \mathbb{N}, \tag{3.10}$$

where $W_\theta^{[t-1]}$ is the submodel (before updating) required by the user at time t.

Correctness in the writing phase: At time t, all submodels stored in each database must be correctly updated as,

$$W_m^{[t]} = \begin{cases} W_m^{[t-1]} + \Delta_m^{[t]}, & \text{if } m = \theta^{[t]}, \\ W_m^{[t-1]}, & \text{if } m \neq \theta^{[t]}. \end{cases} \tag{3.11}$$

A PRUW scheme for FSL is a scheme that satisfies the above privacy, security and correctness requirements. The reading and writing costs are defined as $C_R = \frac{\mathcal{D}}{L}$ and $C_W = \frac{\mathcal{U}}{L}$, respectively, where $\mathcal{D}$ is the total number of bits downloaded from all databases when retrieving the required submodel, $\mathcal{U}$ is the total number of bits sent to all databases in the writing phase and L is the size of each submodel. The total cost is defined as $C_T = C_R + C_W$.

3.3 Main Result

Theorem 3.1 *Following reading and writing costs are achievable in a PRUW system in FSL with $N \geq 4$ non-colluding databases containing M submodels.*

$$C_R = \begin{cases} \frac{2}{1-\frac{2}{N}}, & \text{if } N \text{ is even} \\ \frac{2}{1-\frac{3}{N}}, & \text{if } N \text{ is odd} \end{cases} \tag{3.12}$$

$$C_W = \begin{cases} \frac{2}{1-\frac{2}{N}}, & \text{if } N \text{ is even} \\ \frac{2-\frac{2}{N}}{1-\frac{3}{N}}, & \text{if } N \text{ is odd} \end{cases}. \tag{3.13}$$

Remark 3.1 *The reading and writing costs decrease with increasing number of databases. When N is large, PRUW in FSL can be carried out by downloading/uploading approximately twice as many bits as the size of a submodel.*

Remark 3.2 *The reading and writing costs are independent of the number of submodels M. However, the cost of uploading the queries in the reading phase to download the required submodel is $M\left(\lfloor\frac{N}{2}\rfloor - 1\right)N$, which depends on the number of submodels. This is ignored in this work as it is negligible compared to the reading and writing costs when normalized by the size of a submodel since $N, M << L$ in general.*

Remark 3.3 *The proposed scheme that achieves the reading and writing costs in* (3.12)-(3.13) *is based on allocating $\lfloor\frac{N}{2}\rfloor - 1$ dimensions of the N dimensional space for data and the rest for noise, to guarantee privacy in a communication efficient*

manner. Therefore, when N is odd, one dimension is wasted, resulting in slightly increased reading and writing costs.

Remark 3.4 *When N is odd, reading and writing costs of $C_R = \frac{2-\frac{2}{N}}{1-\frac{3}{N}}$ and $C_W = \frac{2}{1-\frac{3}{N}}$ (i.e., (3.12) and (3.13) switched) can also be achieved by considering less number of noise terms in storage and downloading from only $N-1$ databases in the reading phase.*

3.4 Basic PRUW Scheme

This scheme can be applied to any PRUW system with $N \geq 4$ non-colluding databases. In this scheme, the privacy-security requirement is satisfied by adding random noise terms within the field $\mathbb{F}_q$ to the queries, updates and storage. This is because the noise added queries, updates and storage are uniformly distributed and independent of their original versions. This is known as Shannon's one time pad and also as crypto lemma [105–107]. Furthermore, if $k \in \mathbb{F}_q$ is a constant and $Z \in \mathbb{F}_q$ is random noise, kZ is also random noise (uniformly distributed in $\mathbb{F}_q$) if k and q are coprime.

Based on the crypto lemma, any given random variable A that takes values in $\mathbb{F}_q$ with an arbitrary distribution is independent of the uniformly distributed random variable $A + Z_1$, where Z_1 is random noise. Applying the crypto lemma again on $A+Z_1$ with another random noise symbol Z_2 results in $(A+Z_1)+Z_2$ being uniformly distributed. Moreover, since $(A + Z_1) + Z_2 = A + (Z_1 + Z_2)$ and $Z_1 + Z_2$ is uniformly distributed (again from crypto lemma), $A + Z_1 + Z_2$ is independent of

A. Therefore, by induction, for any $r \in \mathbb{N}$, $A + \sum_{i=1}^{r} Z_i$ is uniformly distributed and independent of A, where Z_is are random noise symbols.

With the above argument, the privacy and security requirements are satisfied by adding $T_1 \geq 1$, $T_2 \geq 1$ and $T_3 \geq 1$ random noise terms to the submodel parameters in storage, queries and updates, respectively. The scheme provides the optimum values of T_1, T_2 and T_3 that minimize the total cost. In other words, this scheme is over-designed with extra noise terms to make the PRUW process more cost efficient.

We now present the basic scheme with arbitrary values of T_1, T_2, T_3 satisfying all $T_i \geq 1$. The optimum values of T_1, T_2, T_3 that minimize the total cost, i.e., T_1^*, T_2^*, T_3^*, are derived later in this section. Let ℓ be the subpacketization of the scheme, i.e., the scheme is defined on a set of ℓ bits of each submodel, which is called a *subpacket*, and is applied repeatedly in the same way on all subpackets in the model. We choose $\ell = N - T_1 - T_2$. An additional constraint given by $\frac{N+T_3-1}{2} \leq T_1 \leq N - T_2 - 1$ must be satisfied by T_1, T_2, T_3 for a given N.[3]

3.4.1 General Scheme

In this section, we present the scheme for the user at time t to privately read from, and write back to the required submodel. For simplicity of notation, we ignore the superscript t in all submodel parameters, user-required submodel index, and updates.

Storage and initialization: The storage of a single subpacket of all sub-

[3]These conditions will be clarified later in this section.

models in database n is given by,

$$S_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i Z_{1,i}^{[1]} \\ W_{2,1} + (f_1 - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i Z_{2,i}^{[1]} \\ \vdots \\ W_{M,1} + (f_1 - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i Z_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} W_{1,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i Z_{1,i}^{[\ell]} \\ W_{2,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i Z_{2,i}^{[\ell]} \\ \vdots \\ W_{M,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i Z_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix}, \tag{3.14}$$

for each $n \in \{1, \ldots, N\}$, where $W_{i,j}$ is the jth bit of submodel i, $Z_{i,j}^{[k]}$ is the $(j+1)$st noise term for the kth bit of W_i, and $\{f_i\}_{i=1}^{\ell}$, $\{\alpha_n\}_{n=1}^{N}$ are globally known distinct constants chosen from $\mathbb{F}_q$, such that each α_n and $f_i - \alpha_n$ for all $i \in \{1, \ldots, \ell\}$ and $n \in \{1, \ldots, N\}$ are coprime with q. Reading and writing to ℓ bits of the required submodel is explained in the rest of this section. The same procedure is followed $\frac{L}{\ell}$ times for the entire PRUW process, where L is the total length of each submodel.

Reading phase: Assume that the user requires to update W_θ. Then, the user sends the following query to database n, $n \in \{1, \ldots, N\}$ in order to read the

existing version of W_θ,

$$Q_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) + \sum_{i=0}^{T_2-1} \alpha_n^i \tilde{Z}_{1,i} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) + \sum_{i=0}^{T_2-1} \alpha_n^i \tilde{Z}_{\ell,i} \end{bmatrix}, \tag{3.15}$$

where $e_M(\theta)$ is the all zeros vector of size $M \times 1$ with a 1 at the θth position and $\tilde{Z}_{i,j}$s are random noise vectors of size $M \times 1$. Database n, $n \in \{1, \ldots, N\}$ then generates the answer given by,

$$A_n = S_n^T Q_n \tag{3.16}$$

$$= \frac{1}{f_1 - \alpha_n} W_{\theta,1} + \frac{1}{f_2 - \alpha_n} W_{\theta,2} + \ldots + \frac{1}{f_\ell - \alpha_n} W_{\theta,\ell}$$

$$+ \phi_0 + \alpha_n \phi_1 + \ldots + \alpha_n^{T_1+T_2-1} \phi_{T_1+T_2-1} \tag{3.17}$$

where ϕ_is are combinations of noise terms that do not depend on n. The answers received from the N databases in matrix form is as follows.

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \ldots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^\eta \\ \frac{1}{f_1-\alpha_2} & \ldots & \frac{1}{f_\ell-\alpha_2} & 1 & \alpha_2 & \ldots & \alpha_2^\eta \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \ldots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \ldots & \alpha_N^\eta \end{bmatrix} \begin{bmatrix} W_{\theta,1} \\ \vdots \\ W_{\theta,\ell} \\ \phi_0 \\ \phi_1 \\ \vdots \\ \phi_\eta \end{bmatrix} \tag{3.18}$$

where $\eta = T_1 + T_2 - 1$. Since the matrix is invertible, the ℓ bits of W_θ can be retrieved using (3.18). The reading cost is given by,

$$C_R = \frac{N}{\ell} = \frac{N}{N - T_1 - T_2}. \tag{3.19}$$

Writing phase: In the writing phase, the user sends a single bit to each database (per subpacket), which is a combination of the updates of the ℓ bits of W_θ and T_3 random noise bits. The combined update bit is a polynomial of α_n, which allows the databases to privately decompose it into the ℓ individual update bits, with the help of the queries received in the reading phase. Finally, these incremental updates are added to the existing storage to obtain the updated storage. As explained later in this section, the above stated decomposition performed at the databases introduces a few extra terms, which are added to the T_1 random noise terms in storage. From the crypto lemma, the updated T_1 noise terms are also independent and uniformly distributed (i.e., random noise). The reason behind over-designing the system to have extra noise terms in storage is to have a number of noise terms that matches the number of extra terms introduced by the *decomposition* performed at the databases in the writing phase. The combined single update bit that the user sends to database n is given by,

$$U_n = \sum_{i=1}^{\ell} \tilde{\Delta}_{\theta,i} \prod_{j=1, j\neq i}^{\ell} (f_j - \alpha_n) + \prod_{j=1}^{\ell} (f_j - \alpha_n) \sum_{m=0}^{T_3-1} \alpha_n^m Z_m, \tag{3.20}$$

for each $n \in \{1, \ldots, N\}$, where Z_ms are random noise bits, $\tilde{\Delta}_{\theta,i} = \frac{\Delta_{\theta,i}}{\prod_{j=1,j\neq i}^{\ell}(f_j - f_i)}$

with $\Delta_{\theta,i}$ being the update for the ith bit of W_θ. Once database n receives U_n, it calculates the incremental update that needs to be added to the existing storage in order to obtain the new and updated storage. This calculation requires the following two definitions and two lemmas.

Definition 3.1 *(Scaling matrix)*

$$D_n = \begin{bmatrix} (f_1 - \alpha_n)I_M & 0 & \dots & 0 \\ 0 & (f_2 - \alpha_n)I_M & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & (f_\ell - \alpha_n)I_M \end{bmatrix}, \tag{3.21}$$

for each $n \in \{1, \dots, N\}$, where I_M is the identity matrix of size $M \times M$.

Definition 3.2 *(Null shaper)*

$$\Omega_n = \begin{bmatrix} \left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_1)}\right) I_M & & \\ & \ddots & \\ & & \left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_\ell)}\right) I_M \end{bmatrix}, \tag{3.22}$$

for each $n \in \{1, \dots, N\}$, where $\mathcal{F}$ is any subset of databases satisfying $|\mathcal{F}| = 2T_1 - N - T_3 + 1$.

Lemma 3.1 *The combined update U_n in* (3.20) *can be decomposed to distinguish*

the kth update from the rest, for each $k \in \{1, \ldots, \ell\}$ as follows,

$$\frac{U_n}{f_k - \alpha_n} = \frac{1}{f_k - \alpha_n} \Delta_{\theta,k} + P_{\alpha_n}(\ell + T_3 - 2), \tag{3.23}$$

where $P_{\alpha_n}(\ell + T_3 - 2)$ is a plynomial in α_n of degree $\ell + T_3 - 2$. The coefficients of $\alpha_n^i s$ in $P_{\alpha_n}(\ell + T_3 - 2)$ are fixed for all n.

Lemma 3.2 *The term $\frac{1}{f_k - \alpha_n}$ for $k \in \{1, \ldots, \ell\}$ remains distinguishable after multiplying by the corresponding term in the null shaper for each $n \notin \mathcal{F}$, i.e.,*

$$\left(\frac{\prod_{r \in \mathcal{F}} (\alpha_r - \alpha_n)}{\prod_{r \in \mathcal{F}} (\alpha_r - f_k)} \right) \frac{1}{f_k - \alpha_n} = \frac{1}{f_k - \alpha_n} + P_{\alpha_n}(|\mathcal{F}| - 1), \tag{3.24}$$

where $P_{\alpha_n}(|\mathcal{F}| - 1)$ is a polynomial in α_n of degree $|\mathcal{F}| - 1$.

The proofs of Lemma 3.1 and Lemma 3.2 are given in Appendix 3.6.[4] With these definitions and lemmas, the incremental update is calculated by,[5]

$$\bar{U}_n = D_n \times \Omega_n \times U_n \times Q_n \tag{3.25}$$

[4]The intuition behind Lemmas 3.1 and 3.2 is as follows: In Lemma 3.1, the term U_n is a single bit that contains information about ℓ parameter updates. At the databases, this combined single bit update must be decomposed into the ℓ separate updates, and placed at relevant positions. To achieve this without leaking any information to the databases, the updates are combined in a specific way, i.e., the first part of U_n is a Lagrange polynomial. Lemma 3.1 presents a result on Lagrange polynomial division, which shows how each update can be separated from the rest by dividing the combined update by a specific factor. Lemma 3.2 is useful in placing the zeros of the incremental update polynomial at certain α_n's (α_ns such that $n \in \mathcal{F}$) so that the writing cost can be saved by not writing to the databases that correspond to those α_n's.

[5]The set $\mathcal{F}$ must satisfy $|\mathcal{F}| \geq 0$, and in cases where $|\mathcal{F}| = 0$, (3.25) is modified as $\bar{U}_n = D_n \times U_n \times Q_n$.

$$= D_n \times \Omega_n \times \begin{bmatrix} \frac{U_n}{f_1-\alpha_n} e_M(\theta) + U_n \sum_{i=0}^{T_2-1} \alpha_n^i \tilde{Z}_{1,i} \\ \frac{U_n}{f_2-\alpha_n} e_M(\theta) + U_n \sum_{i=0}^{T_2-1} \alpha_n^i \tilde{Z}_{2,i} \\ \vdots \\ \frac{U_n}{f_\ell-\alpha_n} e_M(\theta) + U_n \sum_{i=0}^{T_2-1} \alpha_n^i \tilde{Z}_{\ell,i} \end{bmatrix}. \tag{3.26}$$

Using Lemma 3.1,

$$\begin{aligned} \bar{U}_n =& D_n \times \Omega_n \times \begin{bmatrix} \frac{1}{f_1-\alpha_n} \Delta_{\theta,1} e_M(\theta) + e_M(\theta) \sum_{i=0}^{\ell+T_3-2} \alpha_n^i \xi_i^{[1]} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \Delta_{\theta,\ell} e_M(\theta) + e_M(\theta) \sum_{i=0}^{\ell+T_3-2} \alpha_n^i \xi_i^{[\ell]} \end{bmatrix} \\ &+ D_n \times \Omega_n \times \begin{bmatrix} \begin{bmatrix} \sum_{i=0}^{\ell+T_2+T_3-2} \alpha_n^i \tilde{\xi}_{1,i}^{[1]} \\ \vdots \\ \sum_{i=0}^{\ell+T_2+T_3-2} \alpha_n^i \tilde{\xi}_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \sum_{i=0}^{\ell+T_2+T_3-2} \alpha_n^i \tilde{\xi}_{1,i}^{[\ell]} \\ \vdots \\ \sum_{i=0}^{\ell+T_2+T_3-2} \alpha_n^i \tilde{\xi}_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix} \\ =& D_n \times \begin{bmatrix} \left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r-\alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r-f_1)} \right) \frac{1}{f_1-\alpha_n} \Delta_{\theta,1} e_M(\theta) \\ \left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r-\alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r-f_2)} \right) \frac{1}{f_2-\alpha_n} \Delta_{\theta,2} e_M(\theta) \\ \vdots \\ \left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r-\alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r-f_\ell)} \right) \frac{1}{f_\ell-\alpha_n} \Delta_{\theta,\ell} e_M(\theta) \end{bmatrix} \end{aligned} \tag{3.27}$$

$$+ D_n \times \begin{bmatrix} \begin{bmatrix} \sum_{i=0}^{\ell+T_2+T_3-2+|\mathcal{F}|} \alpha_n^i \tilde{\eta}_{1,i}^{[1]} \\ \vdots \\ \sum_{i=0}^{\ell+T_2+T_3-2+|\mathcal{F}|} \alpha_n^i \tilde{\eta}_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \sum_{i=0}^{\ell+T_2+T_3-2+|\mathcal{F}|} \alpha_n^i \tilde{\eta}_{1,i}^{[\ell]} \\ \vdots \\ \sum_{i=0}^{\ell+T_2+T_3-2+|\mathcal{F}|} \alpha_n^i \tilde{\eta}_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix}. \tag{3.28}$$

From Lemma 3.2,

$$\bar{U}_n = \begin{bmatrix} \Delta_{\theta,1} e_M(\theta) \\ \Delta_{\theta,2} e_M(\theta) \\ \vdots \\ \Delta_{\theta,\ell} e_M(\theta) \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1 - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i \hat{\eta}_{1,i}^{[1]} \\ \vdots \\ (f_1 - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i \hat{\eta}_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_\ell - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i \hat{\eta}_{1,i}^{[\ell]} \\ \vdots \\ (f_\ell - \alpha_n) \sum_{i=0}^{T_1-1} \alpha_n^i \hat{\eta}_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix}, \tag{3.29}$$

where (3.27) and (3.28) are due to the fact that U_n and the diagonal elements of Ω_n are polynomials in α_n of degrees $\ell + T_3 - 1$ and $|\mathcal{F}|$, respectively. The polynomial coefficients $\xi_i^{[j]}$, $\tilde{\xi}_i^{[j]}$, $\tilde{\eta}_i^{[j]}$ and $\hat{\eta}_i^{[j]}$ are combined noise terms that do not depend on n. (3.29) is immediate from $|\mathcal{F}| = 2T_1 - N - T_3 + 1$ and $\ell = N - T_1 - T_2$. Note that for databases $n \in \mathcal{F}$, $\Omega_n = 0$, which makes the incremental update of those databases equal to zero. This means that the user could save the writing cost by not sending

the update bit U_n in the writing phase to those databases in $\mathcal{F}$. For each database $n \in \{1, \ldots, N\} \backslash \mathcal{F}$, the incremental update in (3.29) is in the same format as the storage in (3.14). Therefore, the updated storage is given by,

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n, \quad n \in \{1, \ldots, N\} \backslash \mathcal{F}, \tag{3.30}$$

while $S_n^{[t]} = S_n^{[t-1]}$ for $n \in \mathcal{F}$, where $S_n^{[t-1]}$ and $S_n^{[t]}$ are the storages of database n before and after the update, respectively.[6] The writing cost of this scheme is given by,

$$C_W = \frac{N - |\mathcal{F}|}{\ell} = \frac{2N - 2T_1 + T_3 - 1}{N - T_1 - T_2}. \tag{3.31}$$

3.4.2 Total Communication Cost and Optimal Values of T_1, T_2, T_3

From (3.19) and (3.31), the total communication cost is,

$$C_T = C_R + C_W = \frac{3N - 2T_1 + T_3 - 1}{N - T_1 - T_2}. \tag{3.32}$$

The general scheme described in Section 3.4.1 and the total cost in (3.32) are presented for arbitrary T_1, T_2, T_3 satisfying $T_i \geq 1$ for $i = 1, 2, 3$, and $\frac{N+T_3-1}{2} \leq T_1 \leq N - T_2 - 1$, where the last condition is derived from $|\mathcal{F}| \geq 0$ and $\ell \geq 1$. In this subsection, we present the optimum values of T_1, T_2, T_3 that minimize the total cost

[6] Note that W_θ is still updated in databases $n \in \mathcal{F}$ even though the noise added storage has not changed. This is because the zeros of the incremental update polynomials occur at those α_ns that correspond to $n \in \mathcal{F}$.

for a given number of databases N. It is clear that the total cost in (3.32) increases with T_2 and T_3. Therefore, the optimum values of T_2 and T_3 such that the privacy constraints are satisfied are $T_2^* = T_3^* = 1$. Then, the resulting total cost is,

$$C_T = \frac{3N - 2T_1}{N - T_1 - 1}, \tag{3.33}$$

which is increasing in T_1, since $\frac{dC_T}{dT_1} = \frac{N+2}{(N-T_1-1)^2} > 0$. Thus, the optimum value of T_1 satisfying the constraint of $\frac{N+T_3-1}{2} \leq T_1 \leq N - T_2 - 1$ with $T_2^* = T_3^* = 1$ is $T_1^* = \lceil \frac{N}{2} \rceil$. The corresponding optimum subpacketization is $\ell^* = \lfloor \frac{N}{2} \rfloor - 1$ and the optimum reading and writing costs are given in (3.12) and (3.13), respectively.[7]

3.4.3 Example

Consider an example setting where $N = 6$ non-colluding databases store $M = 3$ submodels, and a user who wants to download and update submodel 2 at time t, i.e., $\theta^{[t]} = 2$. The subpacketization (number of parameters considered in a single subpacket) for this example is $\ell = \lfloor \frac{N}{2} \rfloor - 1 = 2$, and the numbers of noise terms added to the storage, queries and updates are given by $T_1 = \lceil \frac{N}{2} \rceil = 3$, $T_2 = 1$ and $T_3 = 1$. The storage of a single subpacket of all submodels in database n,

[7] In this work, we consider the minimization of the total cost (reading+writing cost). The region of achievable reading and writing costs, i.e., the trade-off between reading and writing costs can also be studied by assigning different subpacketizations to the reading and writing phases. It can be shown that the total cost is minimized when the reading and writing subpacketizations are the same, which is the case presented in this work.

$n \in \{1, \ldots, N\}$ is given by (see (3.14)),

$$S_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n)(Z_{1,0}^{[1]} + \alpha_n Z_{1,1}^{[1]} + \alpha_n^2 Z_{1,2}^{[1]}) \\ W_{2,1} + (f_1 - \alpha_n)(Z_{2,0}^{[1]} + \alpha_n Z_{2,1}^{[1]} + \alpha_n^2 Z_{2,2}^{[1]}) \\ W_{3,1} + (f_1 - \alpha_n)(Z_{3,0}^{[1]} + \alpha_n Z_{3,1}^{[1]} + \alpha_n^2 Z_{3,2}^{[1]}) \end{bmatrix} \\ \\ \begin{bmatrix} W_{1,2} + (f_2 - \alpha_n)(Z_{1,0}^{[2]} + \alpha_n Z_{1,1}^{[2]} + \alpha_n^2 Z_{1,2}^{[2]}) \\ W_{2,2} + (f_2 - \alpha_n)(Z_{2,0}^{[2]} + \alpha_n Z_{2,1}^{[2]} + \alpha_n^2 Z_{2,2}^{[2]}) \\ W_{3,2} + (f_2 - \alpha_n)(Z_{3,0}^{[2]} + \alpha_n Z_{3,1}^{[2]} + \alpha_n^2 Z_{3,2}^{[2]}) \end{bmatrix} \end{bmatrix} \tag{3.34}$$

Reading Phase: The user sends the following query to database n, $n \in \{1, \ldots, N\}$, to download submodel 2 (see (3.15))

$$Q_n = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} \tilde{Z}_{1,1} \\ \tilde{Z}_{1,2} \\ \tilde{Z}_{1,3} \end{bmatrix} \\ \\ \frac{1}{f_2 - \alpha_n} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} \tilde{Z}_{2,1} \\ \tilde{Z}_{2,2} \\ \tilde{Z}_{2,3} \end{bmatrix} \end{bmatrix}, \tag{3.35}$$

Database n, $n \in \{1, \ldots, N\}$ sends the answer corresponding to the received query as,

$$A_n = S_n^T Q_n = \frac{W_{2,1}}{f_1 - \alpha_n} + \frac{W_{2,2}}{f_2 - \alpha_n} + \phi_0 + \alpha_n \phi_1 + \alpha_n^2 \phi_2 + \alpha_n^3 \phi_3, \tag{3.36}$$

where ϕ_i are the terms that result from combining all coefficients of α_n^i in the dot product. Note that ϕ_i are the same across all answers from all databases. The user obtains the required two bits $W_{2,1}$ and $W_{2,2}$, using the answers from the N databases as,

$$\begin{bmatrix} A_1 \\ \vdots \\ A_6 \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \frac{1}{f_2-\alpha_1} & 1 & \alpha_1 & \alpha_1^2 & \alpha_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_6} & \frac{1}{f_2-\alpha_6} & 1 & \alpha_6 & \alpha_6^2 & \alpha_6^3 \end{bmatrix} \begin{bmatrix} W_{2,1} \\ W_{2,2} \\ \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}, \tag{3.37}$$

since the matrix is invertible. The resulting reading cost is $C_R = \frac{6}{2} = 3$.

Writing Phase: In the writing phase, the user combines the two updates of the two parameters in the subpacket into a single symbol (see (3.20)) as follows and sends is to database n, $n \in \{1, \ldots, N\}$,

$$U_n = \Delta_{2,1} \frac{f_2 - \alpha_n}{f_2 - f_1} + \Delta_{2,2} \frac{f_1 - \alpha_n}{f_1 - f_2} + (f_1 - \alpha_n)(f_2 - \alpha_n) Z, \tag{3.38}$$

where Z is a random noise symbol. Database n, $n \in \{1, \ldots, N\}$ then calculates the incremental update as (see (3.25)),[8]

$$\bar{U}_n = D_n \times U_n \times Q_n \tag{3.39}$$

[8]Note that the null shaper Ω_n is not used in this example as $|\mathcal{F}| = 2T_1 - N - T_3 + 1 = 0$.

$$= D_n \times \begin{bmatrix} \frac{U_n}{f_1-\alpha_n} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + U_n \begin{bmatrix} \tilde{Z}_{1,1} \\ \tilde{Z}_{1,2} \\ \tilde{Z}_{1,3} \end{bmatrix} \\ \frac{U_n}{f_2-\alpha_n} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + U_n \begin{bmatrix} \tilde{Z}_{2,1} \\ \tilde{Z}_{2,2} \\ \tilde{Z}_{2,3} \end{bmatrix} \end{bmatrix} \tag{3.40}$$

$$= D_n \times \begin{bmatrix} \begin{bmatrix} 0 \\ \frac{\Delta_{2,1}}{f_1-\alpha_n} \\ 0 \end{bmatrix} + \begin{bmatrix} \sum_{i=0}^{2} \alpha_n^i \eta_{1,i}^{[1]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{2,i}^{[1]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{3,i}^{[1]} \end{bmatrix} \\ \begin{bmatrix} 0 \\ \frac{\Delta_{2,2}}{f_2-\alpha_n} \\ 0 \end{bmatrix} + \begin{bmatrix} \sum_{i=0}^{2} \alpha_n^i \eta_{1,i}^{[2]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{2,i}^{[2]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{3,i}^{[2]} \end{bmatrix} \end{bmatrix} \tag{3.41}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 \\ \Delta_{2,1} \\ 0 \end{bmatrix} + (f_1 - \alpha_n) \begin{bmatrix} \sum_{i=0}^{2} \alpha_n^i \eta_{1,i}^{[1]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{2,i}^{[1]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{3,i}^{[1]} \end{bmatrix} \\ \begin{bmatrix} 0 \\ \Delta_{2,2} \\ 0 \end{bmatrix} + (f_2 - \alpha_n) \begin{bmatrix} \sum_{i=0}^{2} \alpha_n^i \eta_{1,i}^{[2]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{2,i}^{[2]} \\ \sum_{i=0}^{2} \alpha_n^i \eta_{3,i}^{[2]} \end{bmatrix} \end{bmatrix} \tag{3.42}$$

where (3.41) is obtained by applying Lemma 3.1. Since the incremental update in (3.42) is in the same form as the storage in (3.34), the storage is updated as $S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n$, and the resulting writing cost is $C_W = \frac{6}{2} = 3$.

3.4.4 Proof of Privacy and Security

The following facts are required for the proofs of privacy and security. In the proposed scheme, the submodel index θ is indicated by $e_M(\theta)$. However, the queries sent to each of the databases are independent from $e_M(\theta)$ due to the random noise terms added to it, from Shannon's one time pad theorem. Similarly, the submodel values $W_{i,j}$ are independent from the storage S_n of each database and the values of updates $\Delta_{i,j}$ are independent from the uploads in the writing phase U_n, due to the random noise terms added.

Privacy of the submodel index: For any $\bar{m} \in \{1, \ldots, M\}^t$, $\bar{u}_n \in \mathbb{F}_q^t$, $\bar{r}_n \in \mathbb{F}_q^{M\ell t}$ and $\bar{s}_n \in \mathbb{F}_q^{M\ell(t+1)}$,

$$\begin{aligned} &P(\theta^{[1:t]} = \bar{m} | Q_n^{[1:t]} = \bar{r}_n, U_n^{[1:t]} = \bar{u}_n, S_n^{[0:t]} = \bar{s}_n) \\ &\qquad = \frac{P(Q_n^{[1:t]} = \bar{r}_n, U_n^{[1:t]} = \bar{u}_n, S_n^{[0:t]} = \bar{s}_n | \theta^{[1:t]} = \bar{m}) P(\theta^{[1:t]} = \bar{m})}{P(Q_n^{[1:t]} = \bar{r}_n, U_n^{[1:t]} = \bar{u}_n, S_n^{[0:t]} = \bar{s}_n)}. \end{aligned} \tag{3.43}$$

Based on the proposed scheme, note that each $S_n^{[t']}$, $t' \in \{1, \ldots, t\}$ term is given by $S_n^{[t']} = S_n^{[t'-1]} + \bar{U}_n^{[t']}$, which makes the two sets $S_n^{[0:t]}$ and $\{S_n^{[0]}, \bar{U}_n^{[1:t]}\}$ statistically equivalent. Moreover, since each $\bar{U}_n^{[t']}$, $t' \in \{1, \ldots, t\}$ is a deterministic function of $U_n^{[t']}$ and $Q_n^{[t']}$, the two sets $\{S_n^{[0:t]}, Q_n^{[1:t]}, U_n^{[1:t]}\}$ and $\{S_n^{[0]}, Q_n^{[1:t]}, U_n^{[1:t]}\}$ are statistically equivalent as well. Since all realizations of the terms in the set $\{S_n^{[0]}, Q_n^{[1:t]}, U_n^{[1:t]}\}$ are random noise terms based on the construction of the proposed scheme, they are

independent of the updating submodel indices $\theta^{[1:t]}$. Therefore,

$$P(\theta^{[1:t]} = \bar{m} | Q_n^{[1:t]} = \bar{r}_n, U_n^{[1:t]} = \bar{u}_n, S_n^{[0:t]} = \bar{s}_n)$$
$$= \frac{P(Q_n^{[1:t]} = \bar{r}_n, U_n^{[1:t]} = \bar{u}_n, S_n^{[0:t]} = \bar{s}_n) P(\theta^{[1:t]} = \bar{m})}{P(Q_n^{[1:t]} = \bar{r}_n, U_n^{[1:t]} = \bar{u}_n, S_n^{[0:t]} = \bar{s}_n)} \tag{3.44}$$
$$= P(\theta^{[1:t]} = \bar{m}), \tag{3.45}$$

which results in the privacy condition in (3.7).

Privacy of the values of updates: For any $\tilde{q} \in \mathbb{F}_q^{\ell t}$, $\bar{u}_n \in \mathbb{F}_q^t$, $\bar{r}_n \in \mathbb{F}_q^{M\ell t}$ and $\bar{s}_n \in \mathbb{F}_q^{M\ell(t+1)}$,

$$P(\Delta_\theta^{[1:t]} = \tilde{q} | Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)$$
$$= \frac{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n | \Delta_\theta^{[1:t]} = \tilde{q}) P(\Delta_\theta^{[1:t]} = \tilde{q})}{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)}. \tag{3.46}$$

As before, all U_n, Q_n and S_n values are random noise terms and are independent of $\Delta_\theta^{[1:t]}$ from Shannon's one time pad theorem. Therefore,

$$P(\Delta_\theta^{[1:t]} = \tilde{q} | Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)$$
$$= \frac{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n) P(\Delta_\theta^{[1:t]} = \tilde{q})}{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)} \tag{3.47}$$
$$= P(\Delta_\theta^{[1:t]} = \tilde{q}), \tag{3.48}$$

which proves the condition in (3.8).

Security of the stored submodels: For any $\bar{w}, \bar{s}_n \in \mathbb{F}_q^{M\ell(t+1)}$, $\bar{u}_n \in \mathbb{F}_q^t$ and

$\bar{r}_n \in \mathbb{F}_q^{M\ell t}$,

$$P(W_{[1:M]}^{[0:t]} = \bar{w} | Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)$$
$$= \frac{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n | W_{[1:M]}^{[0:t]} = \bar{w}) P(W_{[1:M]}^{[0:t]} = \bar{w})}{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)}. \quad (3.49)$$

Based on the same reasoning as before, and since the databases are non-colluding, all U_n, Q_n and S_n values are independent of $W_{1:M}^{[0:t]}$ from Shannon's one time pad theorem. Therefore,

$$P(W_{1:M}^{[0:t]} = \bar{w} | Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)$$
$$= \frac{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n) P(W_{1:M}^{[0:t]} = \bar{w})}{P(Q_n^{[1:t]} = \bar{r}_n, S_n^{[0:t]} = \bar{s}_n, U_n^{[1:t]} = \bar{u}_n)} \quad (3.50)$$
$$= P(W_{1:M}^{[0:t]} = \bar{w}), \quad (3.51)$$

which proves the condition in (3.9).

3.5 Conclusions

In this chapter, we considered the problem of PRUW in relation to FSL, where a user reads, updates and writes back to a submodel of interest, without revealing its index or the values of the updates to the databases storing the submodels. We provided a basic PRUW scheme that performs private FSL by only downloading and uploading twice as many bits as the size of a submodel, while guaranteeing the information-theoretic privacy of the updating submodel index and the values

of the updates. One of the main future directions of PRUW in relation to FSL is to obtain the fundamental limits on the performance metrics. Since the reading phase of private FSL with non-colluding databases is identical to the problem of X-secure T-private information retrieval with $X = T = 1$, there exists a lower bound on the download cost [97]. However, in this work, we show that the total communication cost (reading+writing) can be reduced by incurring a higher reading cost to allow for lower writing costs. Within the scope of cross subspace alignment, we showed that the minimum total communication cost is achieved when the reading and writing costs are symmetric, and when the data and noise subspaces within the N dimensional space occupy approximately $\frac{N}{2}$ dimensions each. Therefore, combining the properties of the reading and writing phases is the main challenge in deriving the converse results. Apart from the converse results, multi-user PRUW, weakly private read-update-write and the presence of adversaries and eavesdroppers in PRUW are among the other immediate future directions.

3.6 Appendix

3.6.1 Proof of Lemma 3.1

Proof: We begin the proof by considering the left term in (3.23) of Lemma 3.1. Note that,

$$\frac{U_n}{f_k - \alpha_n} = \frac{\sum_{i=1}^{\ell} \tilde{\Delta}_{\theta,i} \prod_{j=1, j\neq i}^{\ell} (f_j - \alpha_n) + \prod_{j=1}^{\ell} (f_j - \alpha_n) \sum_{i=0}^{T_3-1} \alpha_n^i Z_i}{f_k - \alpha_n} \tag{3.52}$$

$$= \frac{\tilde{\Delta}_{\theta,k} \prod_{j=1,j\neq k}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n} + \frac{\sum_{i=1,i\neq k}^{\ell} \tilde{\Delta}_{\theta,i} \prod_{j=1,j\neq i}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n}$$
$$+ \frac{\prod_{j=1}^{\ell}(f_j-\alpha_n)(Z_0+\alpha_n Z_1+\ldots+\alpha_n^{T_3-1}Z_{T_3-1})}{f_k-\alpha_n}. \tag{3.53}$$

Now consider,

$$\frac{\prod_{j=1,j\neq k}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n}$$
$$= \frac{(f_1-f_k+f_k-\alpha_n)}{f_k-\alpha_n} \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) \tag{3.54}$$
$$= \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)\frac{\prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n} \tag{3.55}$$
$$= \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)\frac{(f_2-f_k+f_k-\alpha_n)}{f_k-\alpha_n}\prod_{j=3,j\neq k}^{\ell}(f_j-\alpha_n) \tag{3.56}$$
$$= \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)\prod_{j=3,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)(f_2-f_k)\frac{\prod_{j=3,j\neq k}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n} \tag{3.57}$$

$$\vdots$$

$$= \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)\prod_{j=3,j\neq k}^{\ell}(f_j-\alpha_n) + \ldots + \prod_{i=1}^{k-2}(f_i-f_k)\prod_{j=k,j\neq k}^{\ell}(f_j-\alpha_n)$$
$$+ \prod_{i=1}^{k-1}(f_i-f_k)\frac{\prod_{j=k,j\neq k}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n} \tag{3.58}$$
$$= \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)\prod_{j=3,j\neq k}^{\ell}(f_j-\alpha_n) + \ldots + \prod_{i=1}^{k-2}(f_i-f_k)\prod_{j=k+1}^{\ell}(f_j-\alpha_n)$$
$$+ \prod_{i=1}^{k-1}(f_i-f_k)\prod_{j=k+2}^{\ell}(f_j-\alpha_n) + \prod_{i=1,i\neq k}^{k+1}(f_i-f_k)\frac{\prod_{j=k+2}^{\ell}(f_j-\alpha_n)}{f_k-\alpha_n} \tag{3.59}$$

$$\vdots$$

$$= \prod_{j=2,j\neq k}^{\ell}(f_j-\alpha_n) + (f_1-f_k)\prod_{j=3,j\neq k}^{\ell}(f_j-\alpha_n) + \ldots + \prod_{i=1}^{k-1}(f_i-f_k)\prod_{j=k+2}^{\ell}(f_j-\alpha_n)$$

$$+ \prod_{i=1,i\neq k}^{k+1} (f_i - f_k) \prod_{j=k+3}^{\ell} (f_j - \alpha_n) + \dots$$

$$+ \prod_{i=1,i\neq k}^{\ell-2} (f_i - f_k) \frac{(f_{\ell-1} - f_k + f_k - \alpha_n)(f_\ell - \alpha_n)}{f_k - \alpha_n} \tag{3.60}$$

$$= \prod_{j=2,j\neq k}^{\ell} (f_j - \alpha_n) + (f_1 - f_k) \prod_{j=3,j\neq k}^{\ell} (f_j - \alpha_n) + \dots + \prod_{i=1}^{k-1} (f_i - f_k) \prod_{j=k+2}^{\ell} (f_j - \alpha_n)$$

$$+ \prod_{i=1,i\neq k}^{k+1} (f_i - f_k) \prod_{j=k+3}^{\ell} (f_j - \alpha_n) + \dots + (f_\ell - \alpha_n) \prod_{i=1,i\neq k}^{\ell-2} (f_i - f_k)$$

$$+ \prod_{i=1,i\neq k}^{\ell-1} (f_i - f_k) \frac{(f_\ell - f_k + f_k - \alpha_n)}{f_k - \alpha_n} \tag{3.61}$$

$$= \prod_{j=2,j\neq k}^{\ell} (f_j - \alpha_n) + (f_1 - f_k) \prod_{j=3,j\neq k}^{\ell} (f_j - \alpha_n) + \dots + \prod_{i=1}^{k-1} (f_i - f_k) \prod_{j=k+2}^{\ell} (f_j - \alpha_n)$$

$$+ \prod_{i=1,i\neq k}^{k+1} (f_i - f_k) \prod_{j=k+3}^{\ell} (f_j - \alpha_n) + \dots$$

$$+ (f_\ell - \alpha_n) \prod_{i=1,i\neq k}^{\ell-2} (f_i - f_k) + \prod_{i=1,i\neq k}^{\ell-1} (f_i - f_k) + \frac{\prod_{i=1,i\neq k}^{\ell} (f_i - f_k)}{f_k - \alpha_n} \tag{3.62}$$

$$= P_{\alpha_n}(\ell - 2) + \frac{\prod_{i=1,i\neq k}^{\ell} (f_i - f_k)}{f_k - \alpha_n}, \tag{3.63}$$

where $P_{\alpha_n}(\ell - 2)$ is a polynomial in α_n of degree $\ell - 2$. Therefore, from (3.53),

$$\frac{U_n}{f_k - \alpha_n} = \tilde{\Delta}_{\theta,k} \left(P_{\alpha_n}(\ell - 2) + \frac{\prod_{i=1,i\neq k}^{\ell} (f_i - f_k)}{f_k - \alpha_n} \right) + P_{\alpha_n}(\ell + T_3 - 2), \tag{3.64}$$

since the second and third terms of (3.53) result in a polynomial in α_n of degree $\ell + T_3 - 2$. Therefore,

$$\frac{U_n}{f_k - \alpha_n} = \frac{1}{f_k - \alpha_n} \Delta_{\theta,k} + P_{\alpha_n}(\ell + T_3 - 2). \tag{3.65}$$

■

3.6.2 Proof of Lemma 3.2

Proof: We begin the proof by considering the left term in (3.24) of Lemma 3.2. Note that,

$$\left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r-\alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r-f_k)}\right)\frac{1}{f_k-\alpha_n}=\frac{1}{f_k-\alpha_n}\left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r-f_k+f_k-\alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r-f_k)}\right) \tag{3.66}$$

$$=\frac{1}{f_k-\alpha_n}\prod_{r\in\mathcal{F}}\left(1+\frac{f_k-\alpha_n}{\alpha_r-f_k}\right) \tag{3.67}$$

$$=\frac{1}{f_k-\alpha_n}+P_{\alpha_n}(|\mathcal{F}|-1), \tag{3.68}$$

which completes the proof. ■

CHAPTER 4

Private Federated Submodel Learning (FSL) with Top r Sparsification

4.1 Introduction

In this chapter we consider the problem of PRUW in FSL with top r sparsification. In FSL with top r sparsification, the users download and upload only selected fractions of parameters and updates in the reading and writing phases, respectively, to reduce the communication cost. However, revealing the positions of these selected parameters and updates compromises the privacy of the user, in addition to the updating submodel index and the values of the updates. In this chapter, we present how information-theoretic privacy of the users local data can be guaranteed using PRUW in FSL with top r sparsification. We propose a novel scheme which privately reads from and writes to arbitrary parameters of any given submodel, without revealing the submodel index, values of the updates, or the positions of the sparse updates/parameters, to databases. The proposed scheme achieves significantly lower reading and writing costs compared to what is achieved without sparsification.

4.2 Problem Formulation

We consider N non-colluding databases storing M independent submodels, each having P subpackets. At a given time instance t, a given user reads, updates and writes one of the M submodels, while not revealing any information about the updated submodel index or the values of updates to any of the databases. The submodels, queries and updates consist of symbols from a large enough finite field $\mathbb{F}_q$.

In the PRUW process in FSL, users keep reading from and writing to required submodels in an iterative manner. With top r sparsification, each user only writes to a selected r fraction of subpackets of the updating submodel, that contains the most significant r fraction of updates.[1] [2] This significantly reduces the writing cost. Therefore, a given user who reads the same submodel at time $t+1$ only has to download the union of each r fraction of subpackets updated by all users at time t. Let the cardinality of this union be Pr', where $0 \leq r' \leq 1$. This reflects sparsification in the downlink with a rate of r'. For cases where Pr' is significantly large, i.e., with large number of users with non-overlapping sparse updates at time t, there are downlink sparsification protocols such as [80] that limit the value of Pr' in order to reduce the communication cost. Precisely, in this work, we assume

[1] In the update stage (model training) users typically work in continuous fields (real numbers) and make the least significant $1-r$ of the updates equal to zero (i.e., not update) based on the concept of top r sparsification in learning. All updates are converted to symbols in $\mathbb{F}_q$ and sent to the databases. We assume that the zeros in the continuous field are converted to zeros in the finite field.

[2] We assume that all parameters in the most significant r fraction of subpackets have non-zero updates.

that each user only updates Pr subpackets that correspond to the most significant r fraction of updates in the writing phase, and only downloads Pr' subpackets sent by the databases in the reading phase, of the required submodel. The values of r and r' are fixed and determined before the FSL process starts.

The reduction in the communication cost of the PRUW process with sparsification results from communicating only a selected set of updates (parameters) and their positions to the databases (users) in the writing (reading) phase. However, this leaks information about the most and least significant updates of the user. Therefore, to perform top r sparsification in private FSL to reduce the communication cost, the basic PRUW scheme needs to be modified in order to satisfy information-theoretic privacy of the updating submodel index and the values/positions of the sparse updates. Similar to the problem setting of basic PRUW in Section 3.2, the user sends queries to databases to download the required submodel in the reading phase, which are deterministic functions of the required submodel index and random noise generated. In this case the user will only download a selected set of Pr' subpackets, determined by the downlink sparsification protocol at the databases. In the writing phase, the user sends information on the values and positions of the Pr sparse updates, which are deterministic functions of the generated updates and random noise.

The system model is shown in Figure 4.1, which is the same as the model of basic PRUW, with the explicit indication of a coordinator. The coordinator exists in the basic PRUW also, where it is used to initialize the storage with identical random noise terms in all databases in the basic PRUW. In PRUW with sparsification, it is

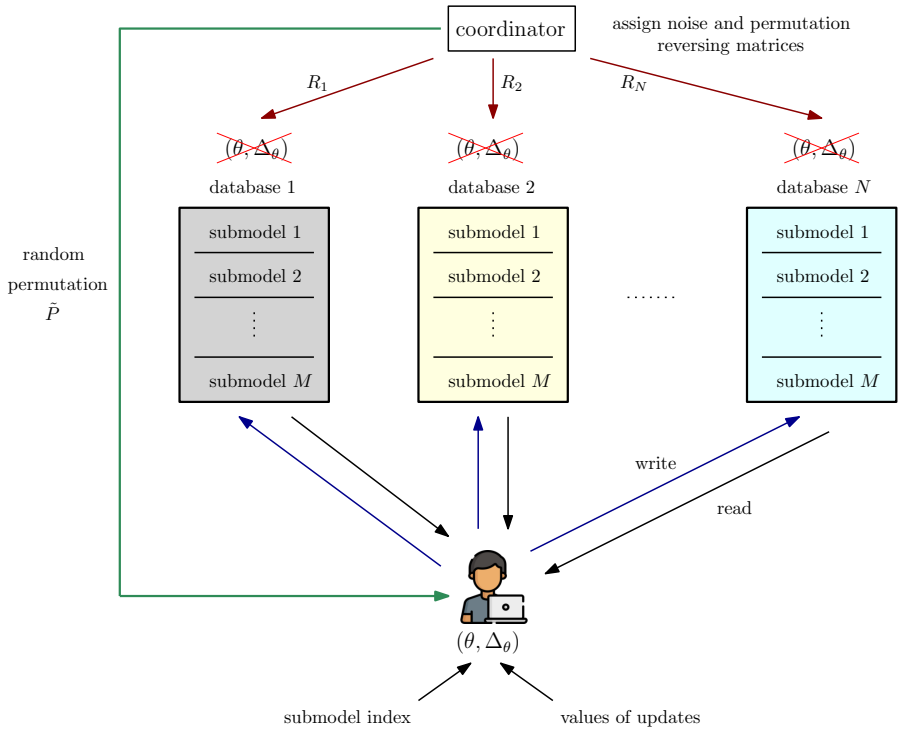


Figure 4.1: PRUW with top r sparsification: system model.

also utilized in guaranteeing the privacy of the indices of sparse updates.

The three components in PRUW with sparsification that need to be kept private are: 1) index of the submodel updated by each user, 2) values of the sparse updates, and 3) indices (positions) of the sparse updates. The formal descriptions of the privacy and security constraints are given below. The constraints are presented from the perspective of a single user at time t, even though multiple independent users update the model simultaneously.

Privacy of the submodel index: At any given time t, no information on the index of the submodel being updated, $\theta^{[t]}$, is allowed to leak to any of the databases with all the information received by the user, i.e., for each database n, $n \in \{1, \ldots, N\}$,

$$I(\theta^{[t]}; Q_n^{[t]}, Y_n^{[t]}, S_n^{[t]}) = 0, \quad t \in \mathbb{N}, \tag{4.1}$$

where $Q_n^{[t]}$ and $Y_n^{[t]}$ are the queries and all the uploads (information on the sparse updates and their positions) sent by the user to database n in the reading and writing phases, and $S_n^{[t]}$ is the content of database n, at time t.[3]

Privacy of the values of updates: At any given time t, no information on the values of updates is allowed to leak to any of the databases with all the information received by the user, i.e., for each database n, $n \in \{1, \dots, N\}$,

$$I(\Delta_\theta^{[t]}; Q_n^{[t]}, Y_n^{[t]}, S_n^{[t]}) = 0, \quad t \in \mathbb{N}, \tag{4.2}$$

where $\Delta_\theta^{[t]}$ is the update (with $(P - Pr)\ell$ zero and $Pr\ell$ non-zero updates, where ℓ is the subpacketization) of submodel $\theta^{[t]}$ generated by the user at time t.[4]

Security of submodels: No information on the parameters of submodels at time t is allowed to leak to any of the databases, i.e., for each database n, $n \in \{1, \dots, N\}$,

$$I(W_{1:M}^{[t]}; S_n^{[t]}, Q_n^{[t]}, Y_n^{[t]}) = 0, \quad t \in \mathbb{Z}_0^+, \tag{4.3}$$

where $W_{1:M}^{[t]}$ represents the parameters of submodels 1 to M at time t.

Correctness in the reading phase: The user should be able to correctly decode the sparse set of Pr' subpackets (denoted by J) of the required submodel, determined by the downlink sparsification protocol, from the answers received in the reading

[3]A detailed explanation on the privacy/security constraints and how they compare with the constraints of basic PRUW is provided in Remark 4.3.

[4]Note that the privacy of both values and positions of the sparse updates is considered in the constraint in (4.2) as $\Delta_\theta^{[t]}$ contains both zero and non-zero updates of which the values are not revealed.

phase, i.e.,

$$H(W_{\theta,J}^{[t-1]}|Q_{1:N}^{[t]}, A_{1:N}^{[t]}, \theta^{[t]}) = 0, \quad t \in \mathbb{N}, \tag{4.4}$$

where $W_{\theta,J}^{[t-1]}$ is the set of subpackets in set J of submodel W_θ at time $t-1$ and $A_n^{[t]}$ is the answer from database n at time t.

Correctness in the writing phase: Let $\theta^{[t]}$ be the updating submodel index and J' be the set of most significant Pr subpackets of $W_{\theta^{[t]}}$ updated by a given user at time t. Then, the subpacket s of submodel m at time t given by $W_m^{[t]}(s)$ is correctly updated as,

$$W_m^{[t]}(s) = \begin{cases} W_m^{[t-1]}(s) + \Delta_m^{[t]}(s), & \text{if } m = \theta^{[t]} \text{ and } s \in J' \\ W_m^{[t-1]}(s), & \text{if } m \neq \theta^{[t]} \text{ or } s \notin J' \end{cases}, \tag{4.5}$$

where $\Delta_m^{[t]}(s)$ is the corresponding update of $W_m^{[t-1]}(s)$. The reading and writing costs are defined the same as in Section 3.2.

4.3 Main Result

In this section, we provide the achievable reading and writing costs of the scheme proposed to perform top r sparsification in FSL, while guaranteeing information-theoretic privacy of the updating submodel index and the values of the updates (which includes the indices of sparse updates). The key component of the proposed scheme is a novel permutation technique, which requires the databases to store

certain *noise-added permutation reversing matrices*. We propose two cases of the scheme based on the structure and the size of the *noise-added permutation reversing matrices*. Theorem 4.1 summarizes the results of the two cases.

Theorem 4.1 *In a private FSL setting with N databases, M submodels (each of size L), P subpackets in each submodel, and r and r' sparsification rates in the uplink and downlink, respectively, the following reading and writing costs are achievable with the corresponding sizes of the noise-added permutation reversing matrices. The reading and writing costs are,*

$$C_R = \frac{4r' + \frac{4}{N}(1+r')\log_q P}{1-\frac{2}{N}} \tag{4.6}$$

$$C_W = \frac{4r(1+\log_q P)}{1-\frac{2}{N}}, \tag{4.7}$$

with noise-added permutation reversing matrices of size $O\left(\frac{L^2}{N^2}\right)$ and,

$$C_R = \frac{2r' + \frac{2}{N}(1+r')\log_q P}{1-\frac{4}{N}} \tag{4.8}$$

$$C_W = \frac{2r(1+\log_q P)}{1-\frac{4}{N}}, \tag{4.9}$$

with noise-added permutation reversing matrices of size $O(L^2)$.

Remark 4.1 *If sparsification is not considered in the PRUW process, the lowest achievable reading and writing costs are given by $C_R = C_W = \frac{2}{1-\frac{2}{N}}$; see Theorem 3.1. Therefore, sparsification with smaller values of r and r' results in significantly reduced communication costs as shown in Theorem 4.1.*

Remark 4.2 *The reading and writing costs double (approximately) as the size of the noise-added permutation reversing matrices reduces from* $O(L^2)$ *to* $O\left(\frac{L^2}{N^2}\right)$.

Remark 4.3 *PRUW in FSL with top r sparsification require additional information from the user, compared to basic PRUW described in Chapter 3, to privately indicate the selected positions of the sparse updates. The privacy and security constraints defined for PRUW with top r sparsification in Section 4.2 are not as strong as the corresponding constraints in basic PRUW due to the extra information required by the users, that results in significantly reduced communication costs via sparsification. As defined in [94], a database (in a non-colluding setting) that has access to all past storages, queries and information on updates/positions is called an internal adversary, and a database that only has access to the current storage, queries and information on updates/positions is called an external adversary. The basic PRUW scheme presented in Chapter 3 is protected from internal adversaries, while the top r sparsification scheme is only safe from external adversaries. The comparison of the privacy and security constraints in the two schemes is as follows.*

The privacy constraint on the submodel index guaranteed in basic PRUW, i.e., (3.7) *can be equivalently stated as* $I(\theta^{[1:t]}; Q_n^{[1:t]}, U_n^{[1;t]}, S_n^{[0]}) = 0$ *as each* $S_n^{[t']}$, $t' \in \{1, \ldots, t\}$ *term can be written as* $S_n^{[t']} = S_n^{[t'-1]} + \bar{U}_n^{[t']}$,[5] *which makes the two sets* $\{S_n^{[0:t]}\}$ *and* $\{S_n^{[0]}, \bar{U}_n^{[1:t]}\}$ *statistically equivalent. Moreover, the incremental update* $\bar{U}_n^{[t']}$ *is a deterministic function of all the information received by the user, i.e.,* $Q_n^{[t']}$ *and* $U_n^{[t']}$, *which makes the two sets* $\{Q_n^{[1:t]}, U_n^{[1:t]}, S_n^{[0:t]}\}$ *and* $\{Q_n^{[1:t]}, U_n^{[1:t]}, S_n^{[0]}\}$

[5] $\bar{U}_n^{[t]}$ is the incremental update at time t. This is explained in Section 3.4.

statistically equivalent as well. This results in the equivalent form of (3.7) *above. It can be shown that the scheme proposed in Section 4.4 for top r sparsification satisfies the privacy constraint* $I(\theta^{[1:t]}; Q_n^{[1:t]}, \hat{U}_n^{[1;t]}, k_n^{[t]}, S_n^{[0]}) = 0$*, where* $\hat{U}_n^{[1:t]}$ *and* $k_n^{[t]}$ *are the quantities uploaded to convey information on the values and positions of the sparse updates, respectively, at corresponding time instances. Note that this constraint is the same as the equivalent form of* (3.7)*, i.e.,* $I(\theta^{[1:t]}; Q_n^{[1:t]}, U_n^{[1;t]}, S_n^{[0]}) = 0$*, with the additional piece of information collected by the user on the positions of the sparse updates at time t, denoted by* $k_n^{[t]}$*. Therefore, the top r sparsification scheme presented in Section 4.4, which is based on the basic PRUW scheme in Section 3.4 is able to satisfy the stronger privacy constraint of basic PRUW even with the additional information on the positions of sparse updates at time t. However, the top r sparsification scheme cannot guarantee the stronger privacy constraint if the information on the positions of sparse updates at all time instances* $\{0, 1, \ldots, t\}$ *is available to the databases. Due to the mismatch between the roles of each database as an internal adversary on queries/values of updates and an external adversary on the positions of sparse updates, we have defined the privacy/security constraints of Section 4.2 (top r sparsification) such that the submodel index, values/positions of sparse updates and submodel values are protected against an external adversary, to make the definitions uniform.*[6]

The top r sparsification scheme is unable to guarantee the stronger privacy constraint $I(\theta^{[1:t]}; Q_n^{[1:t]}, \hat{U}_n^{[1;t]}, k_n^{[1:t]}, S_n^{[1:t]}) = 0$ *(with information on sparse positions at all time instances) because the scheme uses a permutation technique that uses the*

[6]The analysis presented for $\theta^{[1:t]}$ above is applicable for $\Delta_\theta^{[1:t]}$ as well.

same permutation at all time instances, which leaks information about the indices of the sparse updates, when the databases have the permuted information at multiple time instances. This is based on the fact that the most significant Pr subpacket indices that the user chooses in the writing phase at different time instances are correlated. In cases where it is reasonable to assume that the Pr subpackets chosen by a given user at time t is independent of the same quantity at time t' for all $t \neq t'$, the top r sparsification scheme can also achieve the stronger privacy constraint $I(\theta^{[1:t]}; Q_n^{[1:t]}, \hat{U}_n^{[1:t]}, k_n^{[1:t]}, S_n^{[1:t]}) = 0$.

4.4 Proposed Scheme

The scheme is similar to what is presented in Section 3.4 with the additional component of sparse uploads and downloads. In the writing (reading) phase of the scheme in Section 3.4, the updates (values) of all parameters in a given subpacket are combined into a single bit. Thus, a user sends (receives) P bits per database, where P is the number of subpackets in a submodel. In this section, using similar concepts as in Section 3.4, the user only downloads and uploads $Pr' \ll P$ and $Pr \ll P$ bits corresponding to the respective sparse subpackets in the reading and writing phases, respectively, which significantly reduces the communication cost. However, revealing the indices of the subpackets with no update (all zeros) in the writing phase leaks privacy, as the values of those updates (zero) are directly known by the databases.[7] Therefore, to send the indices of the sparse updates privately to the databases in

[7]The sparse set of subpackets in the downlink is determined by the databases with no additional information from the users. Therefore, privacy leakage from the sparse subpacket indices only occurs in the writing phase.

the process of top r sparsification, we use a permutation technique, which is the key component of the proposed scheme. The basic idea of this technique is to *add noise* to the sparse subpacket indices, to hide the real indices from the databases. Note that basic PRUW adds noise to storage, queries and updates, while PRUW with sparsification *adds noise* to the sparse subpacket indices, in addition to the storage, queries and updates. This is analogous to the case with *normal* and *timing* channels, where the *normal* channels add noise to the values while the *timing* channels add noise to the timings. Basic PRUW is analogous to a normal channel while PRUW with sparsification is analogous to a channel that combines characteristics of both normal and timing channels.

The process of *adding noise* to the sparse subpacket indices is as follows. In the writing phase, each user sends a random set of indices corresponding to the Pr sparse subpackets (with non-zero updates) instead of sending the real indices. This random set of indices is generated by the users based on a specific random permutation of all subpacket indices, which is not known by the databases. However, in order to guarantee the correctness of the writing process, the permutation needs to be reversed, and the databases should be able to place the received updates at the correct positions. This is accomplished by the use of *noise-added permutation reversing matrices*, stored at the databases. These permutation reversing matrices rearrange the permuted indices of the sparse subpackets received by the users in the correct order in such a way that the databases do not learn the underlying permutation or the real indices of the sparse updates. The *noise-added permutation reversing matrices* convert the *noise* in the sparse subpacket indices (timing channel)

into added noise in the incremental update calculation (normal channel). These extra noise terms in the incremental update calculation require extra noise terms to be added to storage, for the correctness of the writing phase, which adversely affects the efficiency of the process. However, the fact that the process is carried out only on r fraction of the original number of subpackets makes the overall process significantly efficient in communication cost.

The random selection and assignment of the permutation (to users) and the *noise-added permutation reversing matrices* (to databases) are performed by the same coordinator that assigns similar noise terms to all databases at the initialization stage of basic PRUW. Based on the structure and the size of the *noise-added permutation reversing matrices* stored at each database, we have two cases for the scheme, which result in two different total communication costs. Cases 1 and 2 correspond to *noise-added permutation reversing matrices* of sizes $O\left(\frac{L^2}{N^2}\right)$ and $O(L^2)$, respectively. The general scheme for case 1 is described in detail next, along with the respective modifications for case 2.[8]

[8]An example setting for PRUW with top r sparsification is provided in Section 4.4.2 for both cases 1 and 2. The reader can skip to Section 4.4.2 to get an overview of the proposed scheme.

4.4.1 General Scheme

Storage and initialization: The storage of a single subpacket in database n is,

$$S_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n) \sum_{i=0}^{x} \alpha_n^i Z_{1,i}^{[1]} \\ \vdots \\ W_{M,1} + (f_1 - \alpha_n) \sum_{i=0}^{x} \alpha_n^i Z_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} W_{1,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{x} \alpha_n^i Z_{1,i}^{[\ell]} \\ \vdots \\ W_{M,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{x} \alpha_n^i Z_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix}, \tag{4.10}$$

for each $n \in \{1, \ldots, N\}$, where ℓ is the subpacketization, $W_{i,j}$ is the jth bit of the given subpacket of the ith submodel W_i, $Z_{i,j}^{[k]}$ is the $(j+1)$st noise term for the kth bit of W_i, and $\{f_i\}_{i=1}^{\ell}$, $\{\alpha_n\}_{n=1}^{N}$ are globally known distinct constants chosen from $\mathbb{F}_q$, such that each α_n and $f_i - \alpha_n$ for all $i \in \{1, \ldots, \ell\}$ and $n \in \{1, \ldots, N\}$ are coprime with q. The degree of the noise polynomial in storage (value of x) for cases 1 and 2 are $x = 2\ell$ and $x = \ell + 1$, respectively.

In PRUW, at time $t = 0$, it should be ensured that all noise terms in storage are the same in all databases. This is handled by the coordinator in Figure 4.1. We make use of this coordinator again in PRUW with top r sparsification as follows. In the reading and writing phases, the user only reads and writes parameters/updates corresponding to a subset of subpackets ($\ll P$) without revealing their true indices. The coordinator is used to privately shuffle the true non-zero subpacket indices as

explained next.

At the beginning of the FSL system design, $t = 0$, the coordinator picks a random permutation of indices $\{1, \ldots, P\}$ out of all $P!$ options, denoted by $\tilde{P}$, where P is the number of subpackets. The coordinator sends $\tilde{P}$ to all users involved in the PRUW process. Then, the coordinator sends the corresponding noise-added permutation reversing matrix to database n, $n \in \{1, \ldots, N\}$, given by R_n, whose explicit forms are given below for the two cases. Each user sends the sparse updates to databases in the form (update, position), based on the order specified by $\tilde{P}$, and the databases can reverse the permutations using R_n, without knowing the permutation explicitly.

Case 1: The noise-added permutation reversing matrix is given by,

$$R_n = R + \prod_{i=1}^{\ell} (f_i - \alpha_n) \bar{Z}, \tag{4.11}$$

where R is the permutation reversing matrix and $\bar{Z}$ is a random noise matrix, both of size $P \times P$.[9] For example, for a case where $P = 3$, the matrix R for a random permutation given by $\tilde{P} = \{2, 3, 1\}$ is given by,

$$R = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.12}$$

For each database, R_n is a random noise matrix from Shannon's one time pad

[9]Since $P = \frac{L}{\ell}$ and $\ell = O(N)$, R_n is of $O(P^2)$ which is $O\left(\frac{L^2}{N^2}\right)$.

theorem, from which nothing can be learned about the random permutation $\tilde{P}$. The matrix R_n is fixed at database n at all time instances.

Case 2: The noise-added permutation reversing matrix is given by,

$$R_n = \tilde{R}_n + \tilde{Z}, \tag{4.13}$$

where $\tilde{R}_n$ is the permutation reversing matrix as in case 1 (i.e., R) with all its entries multiplied (element-wise) by the diagonal matrix,

$$\Gamma_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} & 0 & \dots & 0 \\ 0 & \frac{1}{f_2-\alpha_n} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{1}{f_\ell-\alpha_n} \end{bmatrix}. \tag{4.14}$$

Therefore, R_n is of size $P\ell \times P\ell = L \times L$. $\tilde{Z}$ is a random noise matrix of the same size. For the same example with $P = 3$ and $\tilde{P} = \{2, 3, 1\}$, the matrix $\tilde{R}_n$ is given by,

$$\tilde{R}_n = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix}. \tag{4.15}$$

Reading phase: The process of reading (downlink) a subset of parameters of a given submodel without revealing the submodel index or the parameter indices

within the submodel to databases is explained in this section.[10] In the proposed scheme, all communications between the users and databases take place only in terms of the permuted subpacket indices. The users at time $t-1$ send the permuted indices of the sparse subpackets to databases in the writing phase, and the databases work only with these permuted indices of all users to identify the sparse set of subpackets for the next downlink, and send the permuted indices of the selected set of Pr' sparse subpackets to all users at time t. Precisely, let $\tilde{V}$ be the set of permuted indices of the Pr' subpackets chosen by the databases (e.g., union of permuted indices received by all users at time $t-1$) at time t, to be sent to the users in the reading phase. One designated database sends $\tilde{V}$ to each user at time t, from which the users find the real indices of the subpackets in $\tilde{V}$, using the known permutation $\tilde{P}$, received by the coordinator at the initialization stage, i.e., the real indices V corresponding to the permuted set $\tilde{V}$ is given by $V(i) = \tilde{P}(\tilde{V}(i))$, $i \in \{1, \ldots, Pr'\}$. The next steps of the reading phase at time t are as follows. Note that the following steps are identical in both cases. However, the equations given next correspond to case 1, followed by the respective calculations of case 2, separately after the calculations of case 1.

1. The user sends a query to each database n, $n \in \{1, \ldots, N\}$ to privately specify

[10]The privacy constraints of the problem only imply the privacy of the submodel index in the reading phase. However, the privacy constraints applicable to the writing phase in the previous iteration imply the privacy of the sparse subpacket indices of the current reading phase.

the required submodel W_θ given by,

$$Q_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) + \tilde{Z}_1 \\ \frac{1}{f_2-\alpha_n} e_M(\theta) + \tilde{Z}_2 \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) + \tilde{Z}_\ell \end{bmatrix}, \tag{4.16}$$

where $e_M(\theta)$ is the all zeros vector of size $M \times 1$ with a 1 at the θth position and $\tilde{Z}_i$ are random noise vectors of the same size.

2. In order to send the non-permuted version of the ith, $i \in \{1, \ldots, |\tilde{V}|\}$, sparse subpacket (i.e., $V(i) = \tilde{P}(\tilde{V}(i))$) from the set $\tilde{V}$, database n picks the column $\tilde{V}(i)$ of the permutation reversing matrix R_n given in (4.11) indicated by $R_n(:, \tilde{V}(i))$ and calculates the corresponding query given by,

$$Q_n^{[V(i)]} = \begin{bmatrix} R_n(1, \tilde{V}(i))Q_n \\ \vdots \\ R_n(P, \tilde{V}(i))Q_n \end{bmatrix} \tag{4.17}$$

$$= \begin{bmatrix} (R(1, \tilde{V}(i)) + \prod_{i=1}^{\ell}(f_i - \alpha_n)\bar{Z}(1, \tilde{V}(i)))Q_n \\ \vdots \\ (R(P, \tilde{V}(i)) + \prod_{i=1}^{\ell}(f_i - \alpha_n)\bar{Z}(P, \tilde{V}(i)))Q_n \end{bmatrix} \tag{4.18}$$

$$= \begin{bmatrix} 1_{\{V(i)=1\}} \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) \end{bmatrix} + P_{\alpha_n}(\ell) \\ \vdots \\ 1_{\{V(i)=P\}} \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) \end{bmatrix} + P_{\alpha_n}(\ell) \end{bmatrix}, \tag{4.19}$$

where $P_{\alpha_n}(\ell)$ are noise vectors consisting of polynomials in α_n of degree ℓ.

3. Then, the user downloads (non-permuted) subpacket $V(i) = \tilde{P}(\tilde{V}(i))$, $i \in \{1, \ldots, |\tilde{V}|\}$ of the required submodel using the answers received by the N databases given by,

$$A_n^{[V(i)]} = S_n^T Q_n^{[V(i)]} \tag{4.20}$$

$$= \frac{1}{f_1 - \alpha_n} W_{\theta,1}^{[V(i)]} + \ldots + \frac{1}{f_\ell - \alpha_n} W_{\theta,\ell}^{[V(i)]} + P_{\alpha_n}(\ell + x + 1), \tag{4.21}$$

from which the ℓ bits of subpacket $V(i)$, $i \in \{1, \ldots, |\tilde{V}|\}$ can be obtained from the N answers, given that $N = \ell + \ell + x + 2 = 4\ell + 2$ is satisfied. Thus, the subpacketization is $\ell = \frac{N-2}{4}$, and the reading cost is,

$$C_R = \frac{P \log_q P + |\tilde{V}|(N + \log_q P)}{L} \tag{4.22}$$

$$= \frac{P \log_q P + Pr'(N + \log_q P)}{P \times \frac{N-2}{4}} \tag{4.23}$$

$$= \frac{4r' + \frac{4}{N}(1+r')\log_q P}{1 - \frac{2}{N}}, \tag{4.24}$$

where r', $0 \leq r' \leq 1$ is the sparsification rate in the downlink given by $|\tilde{V}| = Pr'$.

Calculations of case 2: The steps described above for case 1 are the same for case 2 as well, with the following modifications in the equations. The query sent by the user to database n, $n \in \{1, \ldots, N\}$ in step 1 is given by,

$$Q_n = \begin{bmatrix} \hat{Q}_1 = e_M(\theta) + (f_1 - \alpha_n)\tilde{Z}_1 \\ \hat{Q}_2 = e_M(\theta) + (f_2 - \alpha_n)\tilde{Z}_2 \\ \vdots \\ \hat{Q}_\ell = e_M(\theta) + (f_\ell - \alpha_n)\tilde{Z}_\ell \end{bmatrix}, \tag{4.25}$$

with the same notation. Then, in step 2, to download the (non-permuted) subpacket $V(i) = \tilde{P}(\tilde{V}(i))$ each database n uses the following procedure. Denote the $P\ell \times \ell$ sized submatrix of R_n (in (4.13)) that includes the first ℓ columns of R_n by $R_n^{[1]}$, and the submatrix that includes the second ℓ columns of R_n by $R_n^{[2]}$, and so on, i.e., $R_n^{[s]} = R_n(:, (s-1)\ell + 1 : s\ell)$. Now, to download subpacket $V(i)$, database n picks $R_n^{[\tilde{V}(i)]}$, computes the sum of the columns in $R_n^{[\tilde{V}(i)]}$ as,

$$\hat{R}_n^{[\tilde{V}(i)]} = \sum_{j=1}^{\ell} R_n^{[\tilde{V}(i)]}(:, j) = \sum_{j=1}^{\ell} R_n(:, (\tilde{V}(i) - 1)\ell + j), \tag{4.26}$$

and calculates the corresponding query as,

$$Q_n^{[V(i)]} = \begin{bmatrix} \begin{bmatrix} \hat{R}_n^{[\tilde{V}(i)]}(1)\hat{Q}_1 \\ \hat{R}_n^{[\tilde{V}(i)]}(2)\hat{Q}_2 \\ \vdots \\ \hat{R}_n^{[\tilde{V}(i)]}(\ell)\hat{Q}_\ell \end{bmatrix} \\ \begin{bmatrix} \hat{R}_n^{[\tilde{V}(i)]}(\ell+1)\hat{Q}_1 \\ \hat{R}_n^{[\tilde{V}(i)]}(\ell+2)\hat{Q}_2 \\ \vdots \\ \hat{R}_n^{[\tilde{V}(i)]}(2\ell)\hat{Q}_\ell \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \hat{R}_n^{[\tilde{V}(i)]}((P-1)\ell+1)\hat{Q}_1 \\ \hat{R}_n^{[\tilde{V}(i)]}((P-1)\ell+2)\hat{Q}_2 \\ \vdots \\ \hat{R}_n^{[\tilde{V}(i)]}(P\ell)\hat{Q}_\ell \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1_{\{V(i)=1\}} \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) \end{bmatrix} + P_{\alpha_n}(1) \\ 1_{\{V(i)=2\}} \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) \end{bmatrix} + P_{\alpha_n}(1) \\ \vdots \\ 1_{\{V(i)=P\}} \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} e_M(\theta) \end{bmatrix} + P_{\alpha_n}(1) \end{bmatrix}, \tag{4.27}$$

where $P_{\alpha_n}(1)$ is vector polynomial in α_n of degree 1 of size $M\ell \times 1$. Then, in step 3, database n sends the answers to the queries in the same way as,

$$A_n^{[V(i)]} = S_n^T Q_n^{[V(i)]} \tag{4.28}$$

$$= \frac{1}{f_1-\alpha_n} W_{\theta,1}^{[V(i)]} + \ldots + \frac{1}{f_\ell-\alpha_n} W_{\theta,\ell}^{[V(i)]} + P_{\alpha_n}(x+2), \tag{4.29}$$

where $W_{i,j}^{[k]}$ is the jth bit of submodel i in subpacket k. The ℓ bits of W_θ in subpacket

$V(i)$ are obtained when $N = \ell + x + 3 = 2\ell + 4$ is satisfied, which gives the subpacketization of case 2 as $\ell = \frac{N-4}{2}$, that results in the reading cost given by,

$$C_R = \frac{P \log_q P + |\tilde{V}|(N + \log_q P)}{L} \tag{4.30}$$

$$= \frac{P \log_q P + Pr'(N + \log_q P)}{P \times \frac{N-4}{2}} \tag{4.31}$$

$$= \frac{2r' + \frac{2}{N}(1 + r') \log_q P}{1 - \frac{4}{N}}, \tag{4.32}$$

with the same notation used for case 1.

Writing phase: Similar to the presentation of the reading phase, we describe the general scheme that is valid for both cases, along with the equations relevant to case 1, and provide the explicit equations corresponding to case 2 at the end. The writing phase of the PRUW scheme with top r sparsification consists of the following steps.

1. The user generates combined updates (one bit per subpacket) of the non-zero subpackets and has zero as the combined update of the rest of the $P(1-r)$ subpackets. The update of subpacket s for database n is given by,[11]

$$U_n(s) = \begin{cases} 0, & s \in B^c, \\ \sum_{i=1}^{\ell} \tilde{\Delta}_{\theta,i}^{[s]} \prod_{j=1, j\neq i}^{\ell} (f_j - \alpha_n) + \prod_{j=1}^{\ell} (f_j - \alpha_n) Z_s, & s \in B, \end{cases} \tag{4.33}$$

where B is the set of subpacket indices with non-zero updates, Z_s is a random noise bit and $\tilde{\Delta}_{\theta,i}^{[s]} = \frac{\Delta_{\theta,i}^{[s]}}{\prod_{j=1, j\neq i}^{\ell} (f_j - f_i)}$ with $\Delta_{\theta,i}^{[s]}$ being the update for the ith bit

[11] A permuted version of these updates is sent to the databases.

of subpacket s of W_θ.

2. The user permutes the updates of subpackets using $\tilde{P}$. The permuted combined updates are given by,

$$\hat{U}_n(i) = U_n(\tilde{P}(i)), \quad i = 1, \ldots, P. \tag{4.34}$$

3. Then, the user sends the following (update, position) pairs to each database n,

$$Y_n^{[j]} = (\hat{U}_n^{[j]}, k^{[j]}), \quad j = 1, \ldots, Pr, \tag{4.35}$$

where $\hat{U}_n^{[j]}$ is the jth non-zero entry in the vector $\hat{U}_n$ in (4.34), and $k^{[j]}$ is its index, i.e., the position of $\hat{U}_n^{[j]}$ in the vector $\hat{U}_n$.

4. Based on the received (update, position) pairs, each database constructs an update vector $\hat{V}_n$ of size $P \times 1$ with $\hat{U}_n^{[j]}$ placed as the $k^{[j]}$th entry,

$$\hat{V}_n = \sum_{j=1}^{Pr} \hat{U}_n^{[j]} e_P(k^{[j]}) = \hat{U}_n. \tag{4.36}$$

5. $\hat{V}_n$ in (4.36) contains the combined updates of the form (4.33) arranged in a random permutation given by $\tilde{P}$. The databases are unable to determine the true indices of the subpackets since $\tilde{P}$ is not known by the databases. However, for correctness in the writing phase, the updates in $\hat{V}_n$ must be rearranged in the correct order. This is done with the noise-added permutation reversing

matrix given in (4.11) as,

$$T_n = R_n \hat{V}_n = R\hat{V}_n + \prod_{i=1}^{\ell}(f_i - \alpha_n)P_{\alpha_n}(\ell), \tag{4.37}$$

where $P_{\alpha_n}(\ell)$ is a $P \times 1$ vector containing noise polynomials in α_n of degree ℓ, $R\hat{V}_n$ contains all updates of all subpackets (including zeros) in the correct order, while $\prod_{i=1}^{\ell}(f_i - \alpha_n)P_{\alpha_n}(\ell)$ contains random noise, that hides the indices of the zero update subpackets.

6. The incremental update is calculated in the same way as described in Section 3.4 in each subpacket as,

$$\bar{U}_n(s) = D_n \times T_n(s) \times Q_n \tag{4.38}$$

$$= D_n \times U_n(s) \times Q_n + D_n \times P_{\alpha_n}(2\ell) \tag{4.39}$$

$$= \begin{cases} \begin{bmatrix} \Delta_{\theta,1}^{[s]} e_M(\theta) \\ \vdots \\ \Delta_{\theta,\ell}^{[s]} e_M(\theta) \end{bmatrix} + \begin{bmatrix} (f_1 - \alpha_n)P_{\alpha_n}(2\ell) \\ \vdots \\ (f_\ell - \alpha_n)P_{\alpha_n}(2\ell) \end{bmatrix}, & s \in B, \\ \begin{bmatrix} (f_1 - \alpha_n)P_{\alpha_n}(2\ell) \\ \vdots \\ (f_\ell - \alpha_n)P_{\alpha_n}(2\ell) \end{bmatrix}, & s \in B^c, \end{cases} \tag{4.40}$$

where $P_{\alpha_n}(2\ell)$ here are noise vectors of size $M\ell \times 1$ in (4.39) and $M \times 1$ in (4.40) with polynomials in α_n of degree 2ℓ and D_n is the scaling matrix given

by,

$$D_n = \begin{bmatrix} (f_1 - \alpha_n)I_M & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & (f_\ell - \alpha_n)I_M \end{bmatrix}, \tag{4.41}$$

for all n. $\bar{U}_n(s)$ is in the same format as the storage in (4.10) with $x = 2\ell$ (case 1) and hence can be added to the existing storage to obtain the updated storage, i.e.,

$$S_n^{[t]}(s) = S_n^{[t-1]}(s) + \bar{U}_n(s), \quad s = 1, \dots, P. \tag{4.42}$$

Note that the degree ℓ noise polynomials in α_n (noise matrix) in the noise-added permutation reversing matrix in (4.11) introduces ℓ extra noise terms in the incremental update calculation, compared to the basic PRUW scheme in which the incremental update has a noise polynomial in α_n of degree ℓ. In other words, the permutation technique requires ℓ dimensions from the N dimensional space which reduces the number of dimensions left for data downloads and uploads to guarantee the privacy of the sparsification process. The writing cost of the scheme is given by,[12]

$$C_W = \frac{PrN(1 + \log_q P)}{L} \tag{4.43}$$

[12]Note that the *upload cost* of the query vector from the reading phase, which is of size $M\ell \times 1$ and is not considered in the writing cost calculation since $\frac{M\ell}{L}$ is negligible.

$$= \frac{PrN(1+\log_q P)}{P \times \frac{N-2}{4}} \tag{4.44}$$

$$= \frac{4r(1+\log_q P)}{1-\frac{2}{N}}. \tag{4.45}$$

Calculations of case 2: Steps 1-4 in the general scheme are valid for case 2 with the same equations. In step 5, the updates of permuted subpackets $\hat{V}_n$ are privately arranged in the correct order as follows. Using the noise added permutation reversing matrix R_n in (4.13), database n, $n \in \{1, \ldots, N\}$ calculates,

$$T_n = R_n \times [\hat{V}_n(1)1_\ell, \hat{V}_n(2)1_\ell, \ldots \hat{V}_n(P)1_\ell]^T \tag{4.46}$$

$$= (\tilde{R}_n + \tilde{Z})[\hat{V}_n(1)1_\ell, \hat{V}_n(2)1_\ell, \ldots \hat{V}_n(P)1_\ell]^T \tag{4.47}$$

$$= \begin{bmatrix} U_n(1) \begin{bmatrix} \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \end{bmatrix} \\ U_n(2) \begin{bmatrix} \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \end{bmatrix} \\ \vdots \\ U_n(P) \begin{bmatrix} \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \end{bmatrix} \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} \begin{bmatrix} \frac{\Delta^{[1]}_{\theta,1}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta^{[1]}_{\theta,\ell}}{f_\ell-\alpha_n} \end{bmatrix} \\ \begin{bmatrix} \frac{\Delta^{[2]}_{\theta,1}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta^{[2]}_{\theta,\ell}}{f_\ell-\alpha_n} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{\Delta^{[P]}_{\theta,1}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta^{[P]}_{\theta,\ell}}{f_\ell-\alpha_n} \end{bmatrix} \end{bmatrix} + P_{\alpha_n}(\ell), \tag{4.48}$$

where 1_ℓ is an all ones vector of size $\ell \times 1$ and $P_{\alpha_n}(\ell)$ here is a vector polynomial

in α_n of degree ℓ of size $P\ell \times 1$. Note that many $U_n(i)$s in the above calculation are zero due to sparsification. The last equality is derived from the application of Lemma 3.1 on expressions of the form $\frac{U_n(i)}{f_j - \alpha_n}$. Recall that $\Delta^{[i]}_{\theta,j} = 0$, $j \in \{1, \ldots, \ell\}$ for all subpackets i, that are not within the Pr selected subpackets with non-zero updates.

Now that the updates are privately arranged in the correct order, it remains only to place the updates at the intended submodel in the storage. Note that the updates of the first subpacket are in the first ℓ rows of T_n, the updates of the second subpacket are in the next ℓ rows of T_n, and so on. Therefore, we divide T_n, based on its correspondence to subpackets as,

$$T_n^{[s]} = T_n((s-1)\ell + 1 : s\ell), \tag{4.49}$$

for $s \in \{1, \ldots, P\}$. With this initialization, for step 6, each database calculates the incremental update of subpacket s, $s \in \{1, \ldots, P\}$ using the the query in the reading phase (4.25) as,

$$\bar{U}_n(s) = D_n \times \begin{bmatrix} T_n^{[s]}(1)\hat{Q}_1 \\ \vdots \\ T_n^{[s]}(\ell)\hat{Q}_\ell \end{bmatrix} \tag{4.50}$$

$$= D_n \times \begin{bmatrix} \left(\frac{\Delta^{[s]}_{\theta,1}}{f_1 - \alpha_n} + P_{\alpha_n}(\ell)\right)(e_M(\theta) + (f_1 - \alpha_n)\tilde{Z}_1) \\ \vdots \\ \left(\frac{\Delta^{[s]}_{\theta,\ell}}{f_\ell - \alpha_n} + P_{\alpha_n}(\ell)\right)(e_M(\theta) + (f_\ell - \alpha_n)\tilde{Z}_1) \end{bmatrix} \tag{4.51}$$

$$= \begin{bmatrix} \Delta_{\theta,1}^{[s]} e_M(\theta) + (f_1 - \alpha_n) P_{\alpha_n}(\ell+1) \\ \vdots \\ \Delta_{\theta,\ell}^{[s]} e_M(\theta) + (f_\ell - \alpha_n) P_{\alpha_n}(\ell+1) \end{bmatrix}, \tag{4.52}$$

with the same notation used in case 1. Since the incremental update is in the same form as the storage in (4.10) with $x = \ell + 1$, $\bar{U}_n(s)$ for $s \in \{1, \ldots, P\}$ is added to the existing storage to obtain the updated version similar to case 1. The resulting writing cost is given by,

$$C_W = \frac{PrN(1 + \log_q P)}{L} \tag{4.53}$$

$$= \frac{PrN(1 + \log_q P)}{P \times \frac{N-4}{2}} \tag{4.54}$$

$$= \frac{2r(1 + \log_q P)}{1 - \frac{4}{N}}. \tag{4.55}$$

Remark 4.4 *This problem can also be solved by considering a classical FSL setting without sparsification with P submodels, i.e., $M = P$, and by using the private FSL scheme in Section 3.4 to update the sparse Pr submodels. However, in this case the normalized cost of sending the queries Q_n given by $\frac{NM\ell}{L} = \frac{NP\ell}{L} = N$ is large, and cannot be neglected.*

4.4.2 Example

Assume that there are $N = 10$ databases containing M submodels, each with $P = 5$ subpackets. The coordinator first picks a random permutation of $\{1, \ldots, 5\}$ out of the 5! options available. Let the realization of the permutation be $\tilde{P} = \{2, 5, 1, 3, 4\}$.

Case 1: The subpacketization is $\ell = \frac{N-2}{4} = 2$ and the storage of database n consists of the model given by,[13]

$$S_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{1,i}^{[1]} \\ \vdots \\ W_{M,1} + (f_1 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{M,i}^{[1]} \end{bmatrix} \\ \begin{bmatrix} W_{1,2} + (f_2 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{1,i}^{[2]} \\ \vdots \\ W_{M,2} + (f_2 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{M,i}^{[2]} \end{bmatrix} \end{bmatrix}, \tag{4.56}$$

since the degree of the noise polynomial is $2\ell = 4$. The permutation reversing matrix is given by,

$$R_n = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} + \prod_{i=1}^{2} (f_i - \alpha_n) \bar{Z}, \tag{4.57}$$

where $\bar{Z}$ is a random noise matrix of size 5×5. The coordinator places matrix R_n at database n at the beginning of the process and sends $\tilde{P}$ to each user. Assume that a given user wants to update submodel θ at time t. In the reading phase, the user only downloads the sparse set of subpackets indicated by the permuted set of indices $\tilde{V} = \{2, 3\}$, which is determined by the databases. One designated database sends

[13]Here we have only presented the storage of a single subpacket.

these permuted indices to each of the users at time t. Then, the user obtains the real indices of the subpackets in $\tilde{V}$, using $V(i) = \tilde{P}(\tilde{V}(i))$ for $i = 1, 2$, i.e., $V = \{5, 1\}$. The user sends the query specifying the requirement of submodel θ given by,

$$Q_n = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} e_M(\theta) + \tilde{Z}_1 \\ \frac{1}{f_2 - \alpha_n} e_M(\theta) + \tilde{Z}_2 \end{bmatrix} \tag{4.58}$$

to database n. Then, each database privately calculates the non-permuted query vector for each subpacket $V(i)$ using the noise added permutation reversing matrix and the query received. The query for subpacket $V(1) = 5$ is,

$$Q_n^{[5]} = \begin{bmatrix} R_n(1, \tilde{V}(1)) Q_n \\ \vdots \\ R_n(P, \tilde{V}(1)) Q_n \end{bmatrix} = \begin{bmatrix} 0_{2M} \\ 0_{2M} \\ 0_{2M} \\ 0_{2M} \\ Q_n \end{bmatrix} + P_{\alpha_n}(2) \tag{4.59}$$

where $P_{\alpha_n}(2)$ is a vector of size $10M \times 1$ consisting of polynomials in α_n of degree 2 and 0_{2M} is the all zeros vector of size $2M \times 1$. Then, the answer from database n corresponding to subpacket $V(1) = 5$ is given by,

$$A_n^{[5]} = S_n^T Q_n^{[5]} \tag{4.60}$$

$$= \frac{1}{f_1 - \alpha_n} W_{\theta,1}^{[5]} + \frac{1}{f_2 - \alpha_n} W_{\theta,2}^{[5]} + P_{\alpha_n}(3 \times 2 + 1), \tag{4.61}$$

from which the 2 bits of subpacket 5 of submodel θ can be correctly obtained by

using the $N = 10$ answers from the ten databases. Similarly, the user can obtain subpacket 1 of W_θ by picking column $\tilde{V}(2) = 3$ of R_n in (4.57) in the calculation of (4.59) and following the same process.

Once the user downloads and trains W_θ, the user generates the r fraction of subpackets with non-zero updates. Let the subpacket indices with non-zero updates be 1 and 4. The noisy updates generated by the user to be sent to database n according to (4.33) is given by $U_n = [U_n(1), 0, 0, U_n(4), 0]^T$ in the correct order. The user then permutes U_n based on the given permutation $\tilde{P}$, i.e., $\hat{U}_n(i) = U_n(\tilde{P}(i))$ for $i = \{1, \ldots, 5\}$,

$$\hat{U}_n = [0, 0, U_n(1), 0, U_n(4)]^T. \tag{4.62}$$

The user sends the values and the positions of the non-zero updates as $(U_n(1), 3)$ and $(U_n(4), 5)$ based on the permuted order. Each database receives these pairs and reconstructs (4.62),

$$\hat{V}_n = U_n(1)e_5(3) + U_n(4)e_5(5) = \hat{U}_n. \tag{4.63}$$

To rearrange the updates back in the correct order privately, database n multiplies $\hat{V}_n$ by the permutation reversing matrix,

$$T_n = R_n \times \hat{V}_n \tag{4.64}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \hat{V}_n + \prod_{i=1}^{2}(f_i - \alpha_n)\bar{Z} \times \hat{V}_n \tag{4.65}$$

$$= [U_n(1), 0, 0, U_n(4), 0]^T + \prod_{i=1}^{2}(f_i - \alpha_n)P_{\alpha_n}(2), \tag{4.66}$$

since $U_n(1)$ and $U_n(4)$ are of the form $\sum_{i=1}^{2} \tilde{\Delta}_{\theta,i} \prod_{j=1,j\neq i}^{2}(f_j - \alpha_n) + \prod_{j=1}^{2}(f_j - \alpha_n)Z = P_{\alpha_n}(2)$. The incremental update of subpacket s, is calculated by,

$$\bar{U}_n(s) = D_n \times T_n(s) \times Q_n \tag{4.67}$$

$$= \begin{cases} \begin{bmatrix} \Delta_{1,1}^{[s]} e_M(\theta) \\ \Delta_{1,2}^{[s]} e_M(\theta) \end{bmatrix} + \begin{bmatrix} (f_1 - \alpha_n)P_{\alpha_n}(4) \\ (f_2 - \alpha_n)P_{\alpha_n}(4) \end{bmatrix}, & s = 1, 4 \\ \begin{bmatrix} (f_1 - \alpha_n)P_{\alpha_n}(4) \\ (f_2 - \alpha_n)P_{\alpha_n}(4) \end{bmatrix}, & s = 2, 3, 5 \end{cases} \tag{4.68}$$

using Lemma 3.1, where $P_{\alpha_n}(4)$ are vectors of size $M \times 1$ consisting of noise polynomials in α_n of degree 4. Since the incremental update is in the same format as the storage in (4.56), the existing storage can be updated as $S_n^{[t]}(s) = S_n^{[t-1]}(s) + \bar{U}_n(s)$ for $s = 1, \ldots, 5$, where $S_n^{[t]}(s)$ is the storage of subpacket s in (4.56) at time t.

Case 2: For this case, the subpacketization is $\ell = \frac{N-4}{2} = 3$ and the storage of

the model is given by,

$$S_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{1,i}^{[1]} \\ \vdots \\ W_{M,1} + (f_1 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{M,i}^{[1]} \end{bmatrix} \\ \begin{bmatrix} W_{1,2} + (f_2 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{1,i}^{[2]} \\ \vdots \\ W_{M,2} + (f_2 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{M,i}^{[2]} \end{bmatrix} \\ \begin{bmatrix} W_{1,3} + (f_3 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{1,i}^{[3]} \\ \vdots \\ W_{M,3} + (f_3 - \alpha_n) \sum_{i=0}^{4} \alpha_n^i Z_{M,i}^{[3]} \end{bmatrix} \end{bmatrix}, \tag{4.69}$$

since the degree of the noise polynomial $x = \ell + 1 = 4$. The permutation reversing matrix stored in database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & \Gamma_n & 0_{3\times3} & 0_{3\times3} \\ \Gamma_n & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \Gamma_n & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \Gamma_n \\ 0_{3\times3} & \Gamma_n & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} + \tilde{Z}, \tag{4.70}$$

where $\Gamma_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} & 0 & 0 \\ 0 & \frac{1}{f_2-\alpha_n} & 0 \\ 0 & 0 & \frac{1}{f_3-\alpha_n} \end{bmatrix}$, and $0_{3\times3}$ is the all zeros matrix of size 3×3.

For the same example where users need to read the permuted subpackets $\tilde{V} = \{2, 3\}$, a designated database sends $\tilde{V}$ to each user, from which the user obtains the non-permuted subpacket indices $V = \{5, 1\}$ using $\tilde{P}$. The user sends the following query to specify the required submodel index θ,

$$Q_n = \begin{bmatrix} \hat{Q}_1 = e_M(\theta) + (f_1 - \alpha_n)\tilde{Z}_1 \\ \hat{Q}_2 = e_M(\theta) + (f_2 - \alpha_n)\tilde{Z}_2 \\ \hat{Q}_3 = e_M(\theta) + (f_3 - \alpha_n)\tilde{Z}_3 \end{bmatrix}. \tag{4.71}$$

To read subpacket $V(1) = 5$, database n first computes the sum of the $\ell = 3$ columns of the $\tilde{V}(1) = 2$nd submatrix of R_n given by,

$$\hat{R}_n^{[\tilde{V}(1)]} = \hat{R}_n^{[2]} = \sum_{j=1}^{3} R_n(:, 3+i) = \begin{bmatrix} 0_3 \\ 0_3 \\ 0_3 \\ 0_3 \\ \frac{1}{f_1 - \alpha_n} \\ \frac{1}{f_2 - \alpha_n} \\ \frac{1}{f_3 - \alpha_n} \end{bmatrix} + \hat{Z}, \tag{4.72}$$

where 0_3 is the all zeros vector of size 3×1, $\hat{Z}$ is a random vector of size 15×1.

Then, each database computes the specific query for $V(1) = 5$ given by,

$$Q_n^{[5]} = \begin{bmatrix} \begin{bmatrix} \hat{R}_n^{[2]}(1)\hat{Q}_1 \\ \hat{R}_n^{[2]}(2)\hat{Q}_2 \\ \hat{R}_n^{[2]}(3)\hat{Q}_3 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \hat{R}_n^{[2]}(13)\hat{Q}_1 \\ \hat{R}_n^{[2]}(14)\hat{Q}_2 \\ \hat{R}_n^{[2]}(15)\hat{Q}_3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \times \hat{Q}_1 \\ 0 \times \hat{Q}_1 \\ 0 \times \hat{Q}_3 \\ \vdots \\ 0 \times \hat{Q}_1 \\ 0 \times \hat{Q}_1 \\ 0 \times \hat{Q}_3 \\ \frac{1}{f_1-\alpha_n}\hat{Q}_1 \\ \frac{1}{f_2-\alpha_n}\hat{Q}_2 \\ \frac{1}{f_3-\alpha_n}\hat{Q}_3 \end{bmatrix} + P_{\alpha_n}(1) = \begin{bmatrix} 0_{3M} \\ 0_{3M} \\ 0_{3M} \\ 0_{3M} \\ \frac{1}{f_1-\alpha_n}e_M(\theta) \\ \frac{1}{f_2-\alpha_n}e_M(\theta) \\ \frac{1}{f_3-\alpha_n}e_M(\theta) \end{bmatrix} + P_{\alpha_n}(1), \quad (4.73)$$

where the polynomial vectors $P_{\alpha_n}(1)$ are resulted by the multiplications of the form $\hat{Z}_i\hat{Q}_j$ and by the residual terms of the calculations of the form $\frac{1}{f_i-\alpha_n}\hat{Q}_j$. Note that the two $P_{\alpha_n}(1)$ vectors in (4.73) are not the same, and they are both some random vector polynomials in α_n of degree 1 of size $15M \times 1$. Each database n, $n \in \{1, \ldots, N\}$ then sends the answers to this query given by,

$$A_n^{[5]} = S_n^T Q_n^{[5]} \quad (4.74)$$

$$= \frac{1}{f_1-\alpha_n}W_{\theta,1}^{[5]} + \frac{1}{f_2-\alpha_n}W_{\theta,2}^{[5]} + \frac{1}{f_3-\alpha_n}W_{\theta,3}^{[5]} + P_{\alpha_n}(6), \quad (4.75)$$

from which the three bits of subpacket 5 can be obtained since $N = 3 + 6 + 1 = 10$.

For the same example considered in case 1, the user sends the two updates

corresponding to subpackets 1 and 4, along with the permuted positions, from which the databases compute $\hat{V}_n = [0, 0, U_n(1), 0, U_n(4)]^T$ given in (4.63), where $U_n(1)$ and $U_n(4)$ are of the form $\sum_{i=1}^{3} \tilde{\Delta}_{\theta,i} \prod_{j=1,j\neq i}^{3}(f_j - \alpha_n) + \prod_{j=1}^{3}(f_j - \alpha_n)Z = P_{\alpha_n}(3)$. Then, database n, $n \in \{1, \ldots, N\}$ rearranges the updates in the correct order as,

$$T_n = R_n \times \begin{bmatrix} \hat{V}_n(1)1_3 \\ \vdots \\ \hat{V}_n(5)1_3 \end{bmatrix} \tag{4.76}$$

$$= \left(\begin{bmatrix} 0_{3\times3} & 0_{3\times3} & \Gamma & 0_{3\times3} & 0_{3\times3} \\ \Gamma & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \Gamma & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \Gamma \\ 0_{3\times3} & \Gamma & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} + \bar{Z} \right) \begin{bmatrix} 0_{3\times1} \\ 0_{3\times1} \\ U_n(1)1_3 \\ 0_{3\times1} \\ U_n(4)1_3 \end{bmatrix} \tag{4.77}$$

$$= \begin{bmatrix} U_n(1) \begin{bmatrix} \frac{1}{f_1-\alpha_n} \\ \frac{1}{f_2-\alpha_n} \\ \frac{1}{f_3-\alpha_n} \end{bmatrix} \\ 0_{3\times1} \\ 0_{3\times1} \\ U_n(4) \begin{bmatrix} \frac{1}{f_1-\alpha_n} \\ \frac{1}{f_2-\alpha_n} \\ \frac{1}{f_3-\alpha_n} \end{bmatrix} \\ 0_{3\times1} \end{bmatrix} + P_{\alpha_n}(3) = \begin{bmatrix} \begin{bmatrix} \frac{\Delta_{\theta,1}^{[1]}}{f_1-\alpha_n} \\ \frac{\Delta_{\theta,2}^{[1]}}{f_2-\alpha_n} \\ \frac{\Delta_{\theta,3}^{[1]}}{f_3-\alpha_n} \end{bmatrix} \\ 0_{3\times1} \\ 0_{3\times1} \\ \begin{bmatrix} \frac{\Delta_{\theta,1}^{[4]}}{f_1-\alpha_n} \\ \frac{\Delta_{\theta,2}^{[4]}}{f_2-\alpha_n} \\ \frac{\Delta_{\theta,3}^{[4]}}{f_3-\alpha_n} \end{bmatrix} \\ 0_{3\times1} \end{bmatrix} + P_{\alpha_n}(3), \tag{4.78}$$

where the last equality is obtained by using Lemma 3.1. Since the subpacketization is $\ell = 3$, we divide T_n into blocks of 3 elements each (subpackets) as shown in (4.49). For example, $T_n^{[1]} = \left[\frac{\Delta_{\theta,1}^{[1]}}{f_1-\alpha_n}, \frac{\Delta_{\theta,2}^{[1]}}{f_2-\alpha_n}, \frac{\Delta_{\theta,3}^{[1]}}{f_3-\alpha_n}\right]^T + P_{\alpha_n}(3)$ and $T_n^{[2]} = P_{\alpha_n}(3)$, where $P_{\alpha_n}(3)$ is a vector polynomial of size 3×1. Then, as an example, the incremental update of the first subpacket is calculated as,

$$\bar{U}_n(1) = D_n \times \begin{bmatrix} T_n^{[1]}(1)\hat{Q}_1 \\ T_n^{[1]}(2)\hat{Q}_2 \\ T_n^{[1]}(3)\hat{Q}_3 \end{bmatrix} \tag{4.79}$$

$$= D_n \times \begin{bmatrix} \left(\frac{\Delta_{\theta,1}^{[1]}}{f_1-\alpha_n} + P_{\alpha_n}(3)\right)(e_M(\theta) + (f_1-\alpha_n)\tilde{Z}_1) \\ \left(\frac{\Delta_{\theta,2}^{[1]}}{f_2-\alpha_n} + P_{\alpha_n}(3)\right)(e_M(\theta) + (f_2-\alpha_n)\tilde{Z}_2) \\ \left(\frac{\Delta_{\theta,3}^{[1]}}{f_3-\alpha_n} + P_{\alpha_n}(3)\right)(e_M(\theta) + (f_3-\alpha_n)\tilde{Z}_3) \end{bmatrix} \tag{4.80}$$

$$= \begin{bmatrix} \Delta_{\theta,1}^{[1]} e_M(\theta) + (f_1 - \alpha_n)P_\alpha(4) \\ \Delta_{\theta,2}^{[1]} e_M(\theta) + (f_2 - \alpha_n)P_\alpha(4) \\ \Delta_{\theta,3}^{[1]} e_M(\theta) + (f_3 - \alpha_n)P_\alpha(4) \end{bmatrix}, \tag{4.81}$$

where $P_{\alpha_n}(4)$ is a vector polynomial in α_n of degree 4, of size $M \times 1$. The above incremental update is directly added to the first subpacket of the existing storage in (4.69) since both are of the same format.

4.4.3 Proof of Privacy

Privacy of the submodel index: The uploads of the user in the writing phase of the proposed scheme is given by $Y_n = (\hat{U}_n, k)$, (see (4.35)) where $\hat{U}_n$ are the val-

ues of sparse updates and k are the corresponding permuted positions. For any $m \in \{1, \ldots, M\}$ and arbitrary realizations of storage, queries, updates and permuted positions $(\bar{s}_n, \bar{r}_n, \bar{u}_n, \bar{k})$, consider the following aposteriori probability from the perspective of an individual database,

$$
\begin{aligned}
&P(\theta^{[t]} = m | S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}) \\
&\quad = \frac{P(\theta^{[t]} = m, S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})}{P(S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})} && (4.82) \\
&\quad = \frac{P(\theta^{[t]} = m, k^{[t]} = \bar{k}) P(S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n)}{P(S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n) P(k^{[t]} = \bar{k})} && (4.83) \\
&\quad = \frac{P(k^{[t]} = \bar{k} | \theta^{[t]} = m) P(\theta^{[t]} = m)}{P(k^{[t]} = \bar{k})}, && (4.84)
\end{aligned}
$$

where (4.83) is due to the fact that $S_n^{[t]}$, $Q_n^{[t]}$ and $\hat{U}_n^{[t]}$ are random noise terms that are independent of $\theta^{[t]}$ and $k^{[t]}$.[14] For all realizations of updates at time t, i.e., $\Delta_\theta^{[t]}$, denoted by δ and permutations $\tilde{P}$, denoted by $\tilde{p}$,

$$
\begin{aligned}
P(k^{[t]} = \bar{k} | \theta^{[t]} = m) &= \sum_{\delta} \sum_{\tilde{p}} P(k^{[t]} = \bar{k}, \tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta | \theta^{[t]} = m) && (4.85) \\
&= \sum_{\delta} \sum_{\tilde{p}} P(k^{[t]} = \bar{k} | \tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta, \theta^{[t]} = m) \\
&\quad \times P(\tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta | \theta^{[t]} = m) && (4.86) \\
&= \sum_{\delta} P(\Delta_\theta^{[t]} = \delta | \theta^{[t]} = m) \sum_{\tilde{p}} 1_{\{\tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta, k^{[t]} = \bar{k}\}} P(\tilde{P} = \tilde{p}) && (4.87) \\
&= \frac{(Pr)!(P - Pr)!}{P!} && (4.88)
\end{aligned}
$$

[14]Note that $S_n^{[t]} = S_n^{[0]} + \sum_{t'=1}^{t} \bar{U}_n^{[t']}$ is random noise, based on the random noise component in $S_n^{[0]}$ (added to submodels at time $t = 0$), from Shannon's one time pad theorem.

$$= \frac{1}{\binom{P}{Pr}}, \tag{4.89}$$

where (4.87) is from the fact that the randomly selected permutation $\tilde{P}$ is independent of the updating submodel index and the values of updates. Moreover,

$$P(k^{[t]} = \bar{k}) = \sum_{m=1}^{M} P(k^{[t]} = \bar{k}|\theta^{[t]} = m)P(\theta^{[t]} = m) \tag{4.90}$$

$$= \frac{1}{\binom{P}{Pr}} \sum_{m=1}^{M} P(\theta^{[t]} = m) = \frac{1}{\binom{P}{Pr}}. \tag{4.91}$$

Therefore, from (4.84),

$$P(\theta^{[t]} = m|S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})$$

$$= \frac{P(k^{[t]} = \bar{k}|\theta^{[t]} = m)P(\theta^{[t]} = m)}{P(k^{[t]} = \bar{k})} \tag{4.92}$$

$$= \frac{\frac{1}{\binom{P}{Pr}} P(\theta^{[t]} = m)}{\frac{1}{\binom{P}{Pr}}} \tag{4.93}$$

$$= P(\theta^{[t]} = m), \tag{4.94}$$

which proves (4.1).

Privacy of the values of updates: For any set of updates of submodel $\theta^{[t]}$ given by, $\tilde{q} \in \mathbb{F}_q^L$ and arbitrary realizations of storage, queries, updates and positions $(\bar{s}_n, \bar{r}_n, \bar{u}_n, \bar{k})$, consider the following aposteriori probability from the perspective of an individual database,

$$P(\Delta_\theta^{[t]} = \tilde{q}|S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})$$

$$= \frac{P(\Delta_\theta^{[t]} = \tilde{q}, S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})}{P(S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})} \tag{4.95}$$

$$= \frac{P(\Delta_\theta^{[t]} = \tilde{q}, k^{[t]} = \bar{k}) P(S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n)}{P(S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n) P(k^{[t]} = \bar{k})}, \tag{4.96}$$

as $S_n^{[t]}$, $Q_n^{[t]}$ and $\hat{U}_n^{[t]}$ are random noise terms that are independent of the values/positions of sparse updates and the submodel index. Therefore,

$$P(\Delta_\theta^{[t]} = \tilde{q} | S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}) = \frac{P(k^{[t]} = \bar{k} | \Delta_\theta^{[t]} = \tilde{q}) P(\Delta_\theta^{[t]} = \tilde{q})}{P(k^{[t]} = \bar{k})}, \tag{4.97}$$

Note that for all possible realizations of permutations $\tilde{p}$,

$$P(k^{[t]} = \bar{k} | \Delta_\theta^{[t]} = \tilde{q}) = \sum_{\tilde{p}} P(k^{[t]} = \bar{k}, \tilde{P} = \tilde{p} | \Delta_\theta^{[t]} = \tilde{q}) \tag{4.98}$$

$$= \sum_{\tilde{p}} P(k^{[t]} = \bar{k} | \tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \tilde{q}) P(\tilde{P} = \tilde{p} | \Delta_\theta^{[t]} = \tilde{q}) \tag{4.99}$$

$$= \sum_{\tilde{p}} 1_{\{k^{[t]} = \bar{k}, \Delta_\theta^{[t]} = \tilde{q}, \tilde{P} = \tilde{p}\}} P(\tilde{P} = \tilde{p}) \tag{4.100}$$

$$= \frac{(Pr)!(P - Pr)!}{P!} \tag{4.101}$$

$$= \frac{1}{\binom{P}{Pr}}, \tag{4.102}$$

where (4.100) is due to the fact that the permutation is independently and randomly selected, irrespective of the values of updates. Therefore, from (4.97),

$$P(\Delta_\theta^{[t]} = \tilde{q} | S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}) = \frac{\frac{1}{\binom{P}{Pr}} P(\Delta_\theta^{[t]} = \tilde{q})}{\frac{1}{\binom{P}{Pr}}} \tag{4.103}$$

$$= P(\Delta_\theta^{[t]} = \tilde{q}), \tag{4.104}$$

which proves (4.2).

Security of the stored submodels: For any $\bar{w} \in \mathbb{F}_q^{M\ell}$ and arbitrary realizations of storage, queries, updates and positions $(\bar{s}_n, \bar{r}_n, \bar{u}_n, \bar{k})$,

$$\begin{aligned} P(W_{[1:M]}^{[t]} = \bar{w} | Q_n^{[t]} = \bar{r}_n, S_n^{[t]} = \bar{s}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}) \\ = \frac{P(Q_n^{[t]} = \bar{r}_n, S_n^{[t]} = \bar{s}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}, W_{[1:M]}^{[t]} = \bar{w})}{P(Q_n^{[t]} = \bar{r}_n, S_n^{[t]} = \bar{s}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k})} \quad (4.105) \\ = \frac{P(Q_n^{[t]} = \bar{r}_n, S_n^{[t]} = \bar{s}_n, \hat{U}_n^{[t]} = \bar{u}_n) P(W_{1:M}^{[t]} = \bar{w}, k^{[t]} = \bar{k})}{P(Q_n^{[t]} = \bar{r}_n, S_n^{[t]} = \bar{s}_n, \hat{U}_n^{[t]} = \bar{u}_n) P(k^{[t]} = \bar{k})} \quad (4.106) \end{aligned}$$

as $\hat{U}_n^{[t]}$, $Q_n^{[t]}$ and $S_n^{[t]}$ values are random noise terms that do not depend on the submodel values or sparse update positions. Therefore,

$$\begin{aligned} P(W_{1:M}^{[t]} = \bar{w} | Q_n^{[t]} = \bar{r}_n, S_n^{[t]} = \bar{s}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}) \\ = \frac{P(k^{[t]} = \bar{k} | W_{1:M}^{[t]} = \bar{w}) P(W_{1:M}^{[t]} = \bar{w})}{P(k^{[t]} = \bar{k})}. \end{aligned} \tag{4.107}$$

For all realizations of updates δ and permutations $\tilde{p}$,

$$P(k^{[t]} = \bar{k} | W_{1:M}^{[t]} = \bar{w}) = \sum_\delta \sum_{\tilde{p}} P(k^{[t]} = \bar{k}, \tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta | W_{1:M}^{[t]} = \bar{w}) \tag{4.108}$$

$$\begin{aligned} &= \sum_\delta \sum_{\tilde{p}} P(k^{[t]} = \bar{k} | \tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta, W_{1:M}^{[t]} = \bar{w}) \\ &\quad \times P(\tilde{P} = \tilde{p}, \Delta_\theta^{[t]} = \delta | W_{1:M}^{[t]} = \bar{w}) \end{aligned} \tag{4.109}$$

$$= \sum_{\delta} P(\Delta_{\theta}^{[t]} = \delta | W_{1:M}^{[t]} = \bar{w})$$

$$\times \sum_{\tilde{p}} 1_{\{\tilde{P}=\tilde{p}, \Delta_{\theta}^{[t]}=\delta, k^{[t]}=\bar{k}\}} P(\tilde{P} = \tilde{p}) \tag{4.110}$$

$$= \frac{(Pr)!(P-Pr)!}{P!} \tag{4.111}$$

$$= \frac{1}{\binom{P}{Pr}}. \tag{4.112}$$

Therefore, from (4.107) and (4.91),

$$P(W_{1:M}^{[t]} = \bar{w} | S_n^{[t]} = \bar{s}_n, Q_n^{[t]} = \bar{r}_n, \hat{U}_n^{[t]} = \bar{u}_n, k^{[t]} = \bar{k}) = \frac{\frac{1}{\binom{P}{Pr}} P(W_{1:M}^{[t]} = \bar{w})}{\frac{1}{\binom{P}{Pr}}} \tag{4.113}$$

$$= P(W_{1:M}^{[t]} = \bar{w}) \tag{4.114}$$

which proves the condition in (4.3).

4.5 Conclusions

In this chapter, we considered the problem of PRUW in FSL with top r sparsification, where only a selected number of parameters and updates are read and written in the reading and writing phases, respectively, in the FSL process. These parameters/updates are selected based on their significance. Therefore, in order to guarantee the information-theoretic privacy of the users participating in the FSL process, the updating submodel index, values of the sparse updates/parameters, and the positions of the sparse updates/parameters must be kept private. This is an extension of the basic PRUW problem considered in Chapter 3. To satisfy the

additional privacy constraint on the positions of the sparse updates/parameters, we introduced a permutation technique, which is based on certain properties of Lagrange polynomials. This permutation technique however requires *noise-added permutation reversing matrices* to be stored in databases. Based on the structure and size of these matrices, the proposed scheme is able to achieve asymptotic (large N) normalized reading and writing costs as low as $2r$ or $4r$, where r is the sparsification rate, which is typically around 10^{-2} to 10^{-3}. The main drawback of the proposed methods is the additional storage cost incurred by the large *noise-added permutation reversing matrices*. We focus on this issue and provide solutions in the context of FL in Chapter 6 .

CHAPTER 5

Private Federated Submodel Learning (FSL) with Random Sparsification

5.1 Introduction

In this Chapter, we investigate the problem of private FSL with random sparsification, which is also formulated as a rate-distortion trade-off in PRUW. In this work, we study how the communication cost of PRUW in FSL can be reduced by performing random sparsification, where pre-determined amounts of randomly chosen parameters and updates are not downloaded and uploaded in the reading and writing phases, respectively. This process introduces some amount of distortion in the two phases since a pre-determined amount of downloads and uploads are made zero (not communicated) irrespective of their real values. We study the behavior of the communication cost with the level of distortion (random sparsification rate) allowed. Our results characterize the rate-distortion trade-off in PRUW, and provide an achievable scheme that works under any given distortion budget.

5.2 Problem Formulation

We consider the basic PRUW setting described in Section 3.2 with N non-colluding databases storing M independent submodels $\{W_1, \ldots, W_M\}$ of size L, each containing random symbols from $\mathbb{F}_q$. At each time instance t, a user updates an arbitrary submodel without revealing its index or the values of updates. Pre-determined amounts of distortion (random sparsification rates in the uplink and downlink) are allowed in the reading and writing phases ($\tilde{D}_r$ and $\tilde{D}_w$, respectively), in order to reduce the communication cost.

Distortion in the reading phase: A distortion of no more than $\tilde{D}_r$ is allowed in the reading phase, i.e., $D_r \leq \tilde{D}_r$, with

$$D_r = \frac{1}{L} \sum_{i=1}^{L} 1_{W_{\theta,i} \neq \hat{W}_{\theta,i}} \tag{5.1}$$

where $W_{\theta,i}$, $\hat{W}_{\theta,i}$ are the actual and downloaded versions of the ith bit of the required submodel W_θ.

Distortion in the writing phase: A distortion of no more than $\tilde{D}_w$ is allowed in the writing phase, i.e., $D_w \leq \tilde{D}_w$, with

$$D_w = \frac{1}{L} \sum_{i=1}^{L} 1_{\Delta_{\theta,i} \neq \hat{\Delta}_{\theta,i}} \tag{5.2}$$

where $\Delta_{\theta,i}$ and $\hat{\Delta}_{\theta,i}$ are the actual and uploaded versions of the ith bit of the update to the required submodel.

The goal of this work is to find schemes that result in the lowest total communication cost under given distortion budgets in the reading and writing phases in the PRUW setting, i.e., a rate-distortion trade-off in PRUW. The privacy constraints on the updating submodel index and the values of updates as well as the security constraint on the submodels are the same as (3.7), (3.8) and (3.9), respectively. The correctness conditions are defined as follows.

Correctness in the reading phase: The user should be able to correctly decode the sparse set of parameters (denoted by G) of the required submodel W_θ from the answers received in the reading phase, i.e.,

$$H(W_{\theta,G}^{[t-1]}|Q_{1:N}^{[t]}, A_{1:N}^{[t]}, \theta^{[t]}) = 0, \quad t \in \mathbb{N}, \tag{5.3}$$

where $W_{\theta,G}^{[t-1]}$ is the set of parameters in set G of submodel W_θ before updating, $Q_n^{[t]}$ is the query sent to database n at time t, $A_n^{[t]}$ is the corresponding answer and $\theta^{[t]}$ is the updating submodel index at time t.[1]

Correctness in the writing phase: Let G' be the sparse set of parameters with non-zero updates of W_θ in the writing phase. Then, the ith parameter of submodel

[1]The correctness condition in (5.3) states that the set of random parameters that the user decides to download from the required submodel in the reading phase, denoted by G, must be correctly downloaded without any uncertainty. The entropy of the entire submodel $W_\theta^{[t-1]}$, given all queries and answers is not zero, as the user only downloads parts of it, and the rest account for distortion. However, the parameters that the user randomly chooses to download must be downloaded with zero ambiguity.

m at time t, $t \in \mathbb{N}$ given by $W_{m,i}^{[t]}$ is correctly updated as,

$$W_{m,i}^{[t]} = \begin{cases} W_{m,i}^{[t-1]} + \Delta_{m,i}^{[t]}, & \text{if } m = \theta^{[t]} \text{ and } i \in G' \\ W_{m,i}^{[t-1]}, & \text{if } m \neq \theta^{[t]} \text{ or } i \notin G' \end{cases}, \tag{5.4}$$

where $\Delta_{m,i}^{[t]}$ is the corresponding update of $W_{m,i}^{[t-1]}$.

In the reading phase, users privately send queries to download a randomly selected set of parameters of the required submodel, and in the writing phase, users privately send updates to be added to a randomly selected set of parameters of the existing submodels while ensuring the distortions resulted by sparse downloads and uploads in the two phases are within the allowed budgets ($\tilde{D}_r$, $\tilde{D}_w$). The reading, writing and total costs are defined the same as in Section 3.2. [2]

5.3 Main Result

Theorem 5.1 *For a PRUW setting with N non-colluding databases containing M independent submodels, where $\tilde{D}_r$ and $\tilde{D}_w$ amounts of distortion are allowed in the reading and writing phases, respectively, the following reading and writing costs are*

[2]Note that PRUW with top r sparsification considered in Chapter 4 also results in incomplete downloads/uploads, as only a selected set of subpackets are downloaded/updated. However, these parameters and updates are carefully chosen based on their significance to improve the accuracy. It has been shown in certain cases that top r sparsification outperforms non-sparse distributed learning [80, 86, 98]. Therefore, we do not consider the ignored subpackets in the reading and writing phases in top r sparsification as distortion. However, in random sparsification, since the selected parameters/updates are chosen randomly, we treat the ignored parameters/updates as distortion, to characterize the rate-distortion trade-off in PRUW.

achievable,

$$C_R = \begin{cases} \frac{2}{1-\frac{2}{N}}(1-\tilde{D}_r), & \textit{even } N \\ \frac{2-\frac{2}{N}}{1-\frac{3}{N}}(1-\tilde{D}_r), & \textit{odd } N, \ \ \tilde{D}_r < \tilde{D}_w \\ \frac{2}{1-\frac{3}{N}}(1-\tilde{D}_r), & \textit{odd } N, \ \ \tilde{D}_r \geq \tilde{D}_w \end{cases}, \tag{5.5}$$

$$C_W = \begin{cases} \frac{2}{1-\frac{2}{N}}(1-\tilde{D}_w), & \textit{even } N \\ \frac{2}{1-\frac{3}{N}}(1-\tilde{D}_w), & \textit{odd } N, \ \ \tilde{D}_r < \tilde{D}_w \\ \frac{2-\frac{2}{N}}{1-\frac{3}{N}}(1-\tilde{D}_w), & \textit{odd } N, \ \ \tilde{D}_r \geq \tilde{D}_w \end{cases}. \tag{5.6}$$

Remark 5.1 *The total communication cost decreases linearly with the increasing amounts of distortion allowed in the reading and writing phases, i.e., the rate-distortion characterization is linear.*

Remark 5.2 *From the perspective of random sparsification,* $1-\tilde{D}_r$ *and* $1-\tilde{D}_w$ *are the sparsification rates in the reading and writing phases, respectively, as* $\tilde{D}_r$ *and* $\tilde{D}_w$ *fractions of parameters and updates that are not downloaded and updated. The reading and writing costs in Theorem 5.1 are essentially the reading and writing costs of basic PRUW, scaled by the sparsification rate.*

5.4 Proposed Scheme

The proposed scheme is an extension of the scheme presented in Section 3.4. The scheme in Section 3.4 considers $\lfloor \frac{N}{2} \rfloor - 1$ bits of the required submodel at a time (called subpacketization) and reads from and writes to $\lfloor \frac{N}{2} \rfloor - 1$ bits using a single bit in

each of the reading and writing phases with no error. In this section, we consider larger subpackets with more bits, i.e., $\ell \geq \lfloor \frac{N}{2} \rfloor - 1$, and correctly read from/write to only $\lfloor \frac{N}{2} \rfloor - 1$ selected bits in each subpacket using single bits in the two phases. The rest of the $\ell - \lfloor \frac{N}{2} \rfloor + 1$ bits in each subpacket account for the distortion in each phase, which is maintained under the allowed distortion budgets. The privacy of the updating submodel index as well as the values of updates is preserved in this scheme, while also not revealing the indices of the distorted uploads/downloads.

The proposed scheme consists of the following three tasks: 1) calculating the optimum reading and writing subpacketizations ℓ_r^* and ℓ_w^* based on the given distortion budgets $\tilde{D}_r$ and $\tilde{D}_w$, 2) specifying the scheme, i.e., storage, reading/writing queries and single bit updates, for given values of ℓ_r^* and ℓ_w^*, and 3) in cases where the subpacketizations calculated in task 1 are non-integers, the model is divided into two sections and two different integer-valued subpacketizations are assigned to the two sections in such a way that the resulting distortion is within the given budgets. Then, task 2 is performed in each of the two sections.

For task 2, note that the scheme in Section 3.4 allocates distinct constants f_i, $i \in \{1, \ldots, \ell\}$ to the ith bit of each subpacket in all submodels (see (3.14)) in the storage, which makes it possible to combine all parameters/updates in a given subpacket to a single bit in a way that the parameters/updates can be correctly and privately decomposed. However, in this scheme, since there may be two subpacketizations in the two phases (reading and writing), we need to ensure that each subpacket in both phases consists of bits with distinct associated f_is. In order to do this, we associate distinct f_is with each consecutive $\max\{\ell_r^*, \ell_w^*\}$ bits in a cyclic

manner so that each subpacket in both phases have distinct f_is. The proposed scheme is explained in detail next, along with an example.

The scheme is defined on a single subpacket in each of the two phases, and is applied repeatedly on all subpackets. Since the number of bits correctly downloaded/updated remains constant at $\lfloor \frac{N}{2} \rfloor - 1$ for a given N, the distortion in a subpacket of size ℓ is $\frac{\ell - \lfloor \frac{N}{2} \rfloor + 1}{\ell}$. Note that this agrees with the definitions in (5.1) and (5.2) since the same distortion is resulted by all subpackets.[3] Therefore, the optimum subpacketizations in the two phases, ℓ_r^* and ℓ_w^*, are functions of $\tilde{D}_r$, $\tilde{D}_w$ and N, and will be calculated in Section 5.4.3. First, we describe the general scheme for any given ℓ_r^* and ℓ_w^*. The scheme is studied under two cases, 1) $y = \ell_w^* > \ell_r^*$, and 2) $y = \ell_r^* \geq \ell_w^*$.

[3]Here, we assume that the integer-valued subpacketization ℓ is uniform throughout the storage, i.e., task 3 is not applicable. The extension to non-uniform subpacketizations (two different subpacketizations as in task 3) is derived from the same concept and is described in detail in Section 5.4.3.

5.4.1 Case 1: $y = \ell_w^* > \ell_r^*$

Storage and initialization: The storage of $y = \max\{\ell_r^*, \ell_w^*\} = \ell_w^*$ bits of all submodels in database n, $n \in \{1, \ldots, N\}$ is given by,

$$S_n = \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_{1,1} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{1,j}^{[1]} \\ \vdots \\ \frac{1}{f_1-\alpha_n} W_{M,1} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{M,j}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_y-\alpha_n} W_{1,y} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{1,j}^{[y]} \\ \vdots \\ \frac{1}{f_y-\alpha_n} W_{M,y} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{M,j}^{[y]} \end{bmatrix} \end{bmatrix}, \tag{5.7}$$

where $W_{i,j}$ is the jth bit of submodel i, $Z_{i,j}^{[k]}$s are random noise terms and $\{f_i\}_{i=1}^{y}$, $\{\alpha_n\}_{n=1}^{N}$ are globally known distinct constants from $\mathbb{F}_q$, such that each α_n and $f_i - \alpha_n$ for all $i \in \{1, \ldots, \ell\}$ and $n \in \{1, \ldots, N\}$ are coprime with q.

Reading phase: In this case, the user considers subpackets of size ℓ_r^* and only downloads $\lfloor \frac{N}{2} \rfloor - 1$ bits of each subpacket. Note that each consecutive $y = \ell_w^*$ bits in storage are associated with distinct f_is, which makes each consecutive set of ℓ_r^* (reading subpacket size) f_is distinct as well (since $\ell_r^* \leq \ell_w^*$). However, not all reading subpackets have the same f_i allocated to their ith bit due to the definition of the storage structure (cyclic allocation of ℓ_w^* distinct values of f_i). Therefore, we cannot define the reading query on a single subpacket and use it repeatedly, since the reading queries depend on f_is. Thus, we define $\gamma_r = \frac{\mathrm{lcm}\{\ell_r^*, \ell_w^*\}}{\ell_r^*}$ queries to read

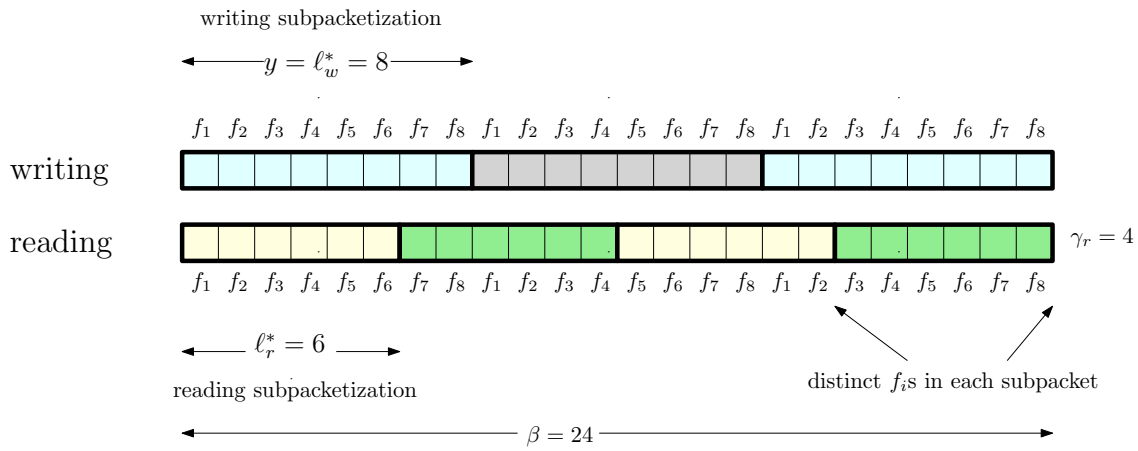


Figure 5.1: An example setting for case 1.

any γ_r consecutive subpackets. Note that the *super subpacket* which consists of any γ_r consecutive reading subpackets have the same set of f_is that occur in a cyclic manner in the storage. Therefore, the γ_r queries can be defined once on a *super subpacket*, and can be used repeatedly throughout the process. An example setting is given in Figure 5.1, where the reading and writing subpacketizations are given by $\ell_r^* = 6$, $\ell_w^* = 8$ and the storage structure repeats at every $y = 8$ bits. Each square in Figure 5.1 corresponds to a single bit of all submodels associated with the corresponding value of f_i. It shows three consecutive storage/writing subpackets on the top row. The same set of bits are viewed as $\gamma_r = \frac{\text{lcm}\{6,8\}}{6} = 4$ reading subpackets, each of size $\ell_r^* = 6$ in the bottom row. Note that each reading subpacket contains distinct f_is, which are not the same across the four subpackets. However, it is clear that the structure of the *super subpacket* which contains the four regular subpackets keeps repeating with the same set of f_is in order. The reading phase has the following steps.

The user sends the following queries to database n to obtain each of the arbi-

trary sets of $\lfloor \frac{N}{2} \rfloor - 1$ bits of each subpacket in each set of $\gamma_r = \frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{\ell_r^*}$ consecutive, non-overlapping subpackets. Let $J_r^{[s]}$ be the set of $\lfloor \frac{N}{2} \rfloor - 1$ randomly chosen parameter indices that are read correctly from subpacket s for $s \in \{1, \ldots, \gamma_r\}$. The query to download subpacket s is,

$$Q_n(s) = \begin{bmatrix} e_M(\theta) 1_{\{1 \in J_r^{[s]}\}} + (f_{g((s-1)\ell_r^*+1)} - \alpha_n) \tilde{Z}_{s,1} \\ \vdots \\ e_M(\theta) 1_{\{\ell_r^* \in J_r^{[s]}\}} + (f_{g(s\ell_r^*)} - \alpha_n) \tilde{Z}_{s,\ell_r^*} \end{bmatrix}, \tag{5.8}$$

and the corresponding subpacket s is,

$$S_n(s) = \begin{bmatrix} \begin{bmatrix} \frac{1}{f_{g((s-1)\ell_r^*+1)} - \alpha_n} W_{1,1}^{[s]} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{1,j}^{[1]}(s) \\ \vdots \\ \frac{1}{f_{g((s-1)\ell_r^*+1)} - \alpha_n} W_{M,1}^{[s]} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{M,j}^{[1]}(s) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_{g(s\ell_r^*)} - \alpha_n} W_{1,\ell_r^*}^{[s]} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{1,j}^{[y]}(s) \\ \vdots \\ \frac{1}{f_{g(s\ell_r^*)} - \alpha_n} W_{M,\ell_r^*}^{[s]} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor - 1} \alpha_n^j Z_{M,j}^{[y]}(s) \end{bmatrix} \end{bmatrix}, \tag{5.9}$$

where $e_M(\theta)$ is the all zeros vector of size $M \times 1$ with a 1 at the θth position, $\tilde{Z}_{i,j}$s are random noise vectors of size $M \times 1$ and the function $g(\cdot)$ is defined as,

$$g(x) = \begin{cases} x \bmod y, & \text{if } x \bmod y \neq 0 \\ y, & \text{if } x \bmod y = 0 \end{cases} \tag{5.10}$$

Note that the *super subpacket* $S_n = [S_n^{[1]}, \ldots, S_n^{[\gamma_r]}]^T$ is the concatenation of $\frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{y}$ blocks of the form (5.7). The γ_r answers received by database n, $n \in \{1, \ldots, N\}$, are given by,

$$A_n(s) = S_n(s)^T Q_n(s) \tag{5.11}$$

$$= \sum_{i=1}^{\ell_r^*} \left(\frac{1}{f_{g((s-1)\ell_r^*+i)} - \alpha_n} W_{\theta,i}^{[s]} \right) 1_{\{i \in J_r^{[s]}\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor) \tag{5.12}$$

for each $s \in \{1, \ldots, \gamma_r\}$, where $P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor)$ is a polynomial in α_n of degree $\lfloor \frac{N}{2} \rfloor$. Since $|J_r^{[s]}| = \lfloor \frac{N}{2} \rfloor - 1$ for each $s \in \{1, \ldots, \gamma_r\}$, the required $\lfloor \frac{N}{2} \rfloor - 1$ bits of each of the γ_r subpackets can be correctly retrieved from $2\lfloor \frac{N}{2} \rfloor$ answers of the form (5.12) (corresponding to $2\lfloor \frac{N}{2} \rfloor$ databases). Note that when N is odd, the user has to download answers from only $N-1$ databases, since $N-1$ equations of the form (5.12) with distinct α_ns suffice to solve for the $\lfloor \frac{N}{2} \rfloor - 1$ parameters of the required submodel when N is odd. The resulting reading cost of the first case is given by,

$$C_R^{[1]} = \begin{cases} \frac{\gamma_r \times N}{\gamma_r \times \ell_r^*} = \frac{N}{\ell_r^*}, & \text{even } N, \\ \frac{\gamma_r \times (N-1)}{\gamma_r \times \ell_r^*} = \frac{N-1}{\ell_r^*}, & \text{odd } N. \end{cases} \tag{5.13}$$

For a better understanding of the reading phase, we present the queries and answers corresponding to the example in Figure 5.1 next. Assume that $N = 6$ for this example and the set of $\lfloor \frac{N}{2} \rfloor - 1 = 2$ parameter indices that are read correctly from the second subpacket (out of $\gamma_r = 4$ subpackets) is given by $J_r^{[2]} = \{2, 5\}$.

Then, the query corresponding to the second subpacket is given by,

$$Q_n(2) = \begin{bmatrix} & (f_7 - \alpha_n)\tilde{Z}_{2,1} \\ e_M(\theta)+ & (f_8 - \alpha_n)\tilde{Z}_{2,2} \\ & (f_1 - \alpha_n)\tilde{Z}_{2,3} \\ & (f_2 - \alpha_n)\tilde{Z}_{2,4} \\ e_M(\theta)+ & (f_3 - \alpha_n)\tilde{Z}_{2,5} \\ & (f_4 - \alpha_n)\tilde{Z}_{2,6} \end{bmatrix}, \tag{5.14}$$

which is used to obtain the 2nd and 5th elements of the second reading subpacket given by,

$$S_n(2) = \begin{bmatrix} \frac{1}{f_7-\alpha_n} W^{[2]}_{\cdot,1} + \sum_{j=0}^{2} \alpha_n^j Z^{[1]}_{\cdot,j}(2) \\ \frac{1}{f_8-\alpha_n} W^{[2]}_{\cdot,2} + \sum_{j=0}^{2} \alpha_n^j Z^{[2]}_{\cdot,j}(2) \\ \frac{1}{f_1-\alpha_n} W^{[2]}_{\cdot,3} + \sum_{j=0}^{2} \alpha_n^j Z^{[3]}_{\cdot,j}(2) \\ \frac{1}{f_2-\alpha_n} W^{[2]}_{\cdot,4} + \sum_{j=0}^{2} \alpha_n^j Z^{[4]}_{\cdot,j}(2) \\ \frac{1}{f_3-\alpha_n} W^{[2]}_{\cdot,5} + \sum_{j=0}^{2} \alpha_n^j Z^{[5]}_{\cdot,j}(2) \\ \frac{1}{f_4-\alpha_n} W^{[2]}_{\cdot,6} + \sum_{j=0}^{2} \alpha_n^j Z^{[6]}_{\cdot,j}(2) \end{bmatrix}, \tag{5.15}$$

where $W^{[2]}_{\cdot,i} = [W^{[2]}_{1,i}, \ldots, W^{[2]}_{M,i}]^T$ and $Z^{[i]}_{\cdot,j}(2) = [Z^{[i]}_{1,j}(2), \ldots, Z^{[i]}_{M,j}(2)]^T$. Then, the answer from database n, $n \in \{1, \ldots, 6\}$ for this specific subpacket ($s = 2$) is given by,

$$A_n(2) = S_n(2)^T Q_n(2) \tag{5.16}$$

$$=\frac{1}{f_8-\alpha_n}W_{\theta,2}^{[2]}+\frac{1}{f_3-\alpha_n}W_{\theta,5}^{[2]}+P_{\alpha_n}(3), \tag{5.17}$$

where $P_{\alpha_n}(3)$ is a polynomial in α_n of degree 3. The user can then find $W_{\theta,2}^{[2]}$ and $W_{\theta,5}^{[2]}$ by solving,

$$\begin{bmatrix} A_1(2) \\ \vdots \\ A_6(2) \end{bmatrix} = \begin{bmatrix} \frac{1}{f_8-\alpha_1} & \frac{1}{f_3-\alpha_1} & 1 & \alpha_1 & \alpha_1^2 & \alpha_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_8-\alpha_6} & \frac{1}{f_3-\alpha_6} & 1 & \alpha_6 & \alpha_6^2 & \alpha_6^3 \end{bmatrix} \begin{bmatrix} W_{\theta,2}^{[2]} \\ W_{\theta,5}^{[2]} \\ R_0 \\ R_1 \\ R_2 \\ R_3 \end{bmatrix}. \tag{5.18}$$

Writing phase: Since the subpacketization in the writing phase is y, which is the same as the period of the cyclic structure of the storage in (5.7), a single writing query, specifying the submodel index and the correctly updated bit indices, defined on a single subpacket suffices to repeatedly update all subpackets, as the f_is in all subpackets are identical. The writing query sent to database n, $n \in \{1, \ldots, N\}$, is,

$$\tilde{Q}_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} e_M(\theta) 1_{\{1\in J_w\}} + \hat{Z}_1 \\ \vdots \\ \frac{1}{f_y-\alpha_n} e_M(\theta) 1_{\{y\in J_w\}} + \hat{Z}_y \end{bmatrix}, \tag{5.19}$$

where J_w is the set of indices of the $\lfloor \frac{N}{2} \rfloor - 1$ parameters of each subpacket that are updated correctly and $\hat{Z}$s are random noise vectors of size $M \times 1$. Since $\tilde{Q}_n$ is sent

only once, the same set of J_w indices will be correctly updated in all subpackets. The user then sends a single bit combined update for each subpacket of the form (5.7) given by,

$$U_n = \sum_{i \in J_w} \tilde{\Delta}_{\theta,i} \prod_{j \in J_w, j \neq i} (f_j - \alpha_n) + \prod_{j \in J_w} (f_j - \alpha_n) Z, \tag{5.20}$$

for each $n \in \{1, \ldots, N\}$ where $\tilde{\Delta}_{\theta,i} = \frac{\Delta_{\theta,i}}{\prod_{j \in J_w, j \neq i}(f_j - f_i)}$ with $\Delta_{\theta,i}$ being the update of the ith parameter of submodel θ (of the subpacket considered) and Z is a random noise bit. Each database then calculates the incremental update as,

$$\tilde{U}_n = U_n \times \tilde{Q}_n \tag{5.21}$$

$$= \begin{bmatrix} \frac{\Delta_{\theta,1}}{f_1 - \alpha_n} e_M(\theta) 1_{\{1 \in J_w\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor - 1) \\ \vdots \\ \frac{\Delta_{\theta,y}}{f_y - \alpha_n} e_M(\theta) 1_{\{y \in J_w\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor - 1) \end{bmatrix}, \tag{5.22}$$

where $P_{\alpha_n}(\cdot)$ is a polynomial in α_n of degree in parenthesis,[4] and (5.22) is obtained from (5.21) by applying Lemma 3.1. Since the incremental update in (5.22) is in the same form as the storage in (5.7), (5.22) is directly added to the existing storage to obtain the updated submodel as,

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n^{[t]}, \tag{5.23}$$

for each $n \in \{1, \ldots, N\}$ for both even and odd N. The writing cost of case 1 is

[4]Note that all $P_{\alpha_n}(\cdot)$ are not the same and each polynomial is resulted by the combination of all unwanted terms (noise subspace) resulting from the decomposition of combined updates.

given by,

$$C_W^{[1]} = \frac{N}{\ell_w^*}. \tag{5.24}$$

5.4.2 Case 2: $y = \ell_r^* \geq \ell_w^*$

Storage and initialization: The storage of $y = \max\{\ell_r^*, \ell_w^*\} = \ell_r^*$ bits of all submodels in database n, $n \in \{1, \ldots, N\}$ is given by,

$$S_n = \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_{1,1} + \sum_{j=0}^{\lceil \frac{N}{2} \rceil - 1} \alpha_n^j Z_{1,j}^{[1]} \\ \vdots \\ \frac{1}{f_1-\alpha_n} W_{M,1} + \sum_{j=0}^{\lceil \frac{N}{2} \rceil - 1} \alpha_n^j Z_{M,j}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_y-\alpha_n} W_{1,y} + \sum_{j=0}^{\lceil \frac{N}{2} \rceil - 1} \alpha_n^j Z_{1,j}^{[y]} \\ \vdots \\ \frac{1}{f_y-\alpha_n} W_{M,y} + \sum_{j=0}^{\lceil \frac{N}{2} \rceil - 1} \alpha_n^j Z_{M,j}^{[y]} \end{bmatrix} \end{bmatrix}, \tag{5.25}$$

where $W_{i,j}$ is the jth bit of submodel i and the Zs are random noise terms.

Reading phase: In the reading phase, each user correctly downloads $\lfloor \frac{N}{2} \rfloor - 1$ bits from each subpacket while not downloading the rest of the $\ell_r^* - \lfloor \frac{N}{2} \rfloor + 1$ bits. The user randomly picks the $\lfloor \frac{N}{2} \rfloor - 1$ bits within the subpacket that are downloaded correctly and prepares the query to be sent to database n as follows. Let J_r be the

set of indices of the $\lfloor \frac{N}{2} \rfloor - 1$ bits that need to be downloaded correctly. Then,

$$Q_n = \begin{bmatrix} e_M(\theta) 1_{\{1 \in J_r\}} + (f_1 - \alpha_n)\tilde{Z}_1 \\ \vdots \\ e_M(\theta) 1_{\{y \in J_r\}} + (f_y - \alpha_n)\tilde{Z}_y \end{bmatrix}, \tag{5.26}$$

where $\tilde{Z}$ are random noise vectors of size $M \times 1$. The answer of database n is,

$$A_n = S_n^T Q_n = \sum_{i=1}^{y} \left(\frac{1}{f_i - \alpha_n} W_{\theta,i} \right) 1_{\{i \in J_r\}} + P_{\alpha_n}(\lceil \frac{N}{2} \rceil). \tag{5.27}$$

Since $|J_r| = \lfloor \frac{N}{2} \rfloor - 1$, the user required $\lfloor \frac{N}{2} \rfloor - 1$ bits of W_θ can be correctly downloaded using the answers received by the N databases. The resulting reading cost of case 2 is given by,

$$C_R^{[2]} = \frac{N}{\ell_r^*}. \tag{5.28}$$

Writing phase: In the writing phase, the user considers subpackets of size ℓ_w^* and only updates $\lfloor \frac{N}{2} \rfloor - 1$ out of the ℓ_w^* bits correctly, while making the updates of the rest of the $\ell_w^* - \lfloor \frac{N}{2} \rfloor + 1$ bits zero. The $\lfloor \frac{N}{2} \rfloor - 1$ bits that are correctly updated are chosen randomly. The following steps describe the writing process when $\ell_w^* \leq \ell_r^* = y$.

1. A general writing query that specifies the submodel to which the update should be added, along with the positions of the $\lfloor \frac{N}{2} \rfloor - 1$ non-zero updates in each subpacket is sent first. The same query from the reading phase (5.26) can be used if the subpacketization and the indices of the correct $\lfloor \frac{N}{2} \rfloor - 1$ bits

within the subpacket are the same in both phases. However, for the strict case $\ell_r^* > \ell_w^*$, we need a new general query $\tilde{Q}_n$ for the writing phase. $\tilde{Q}_n$ consists of $\gamma_w = \frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{\ell_w^*}$ sub-queries, where each sub-query corresponds to a single subpacket of size ℓ_w^*. These sub-queries are required since the storage structure of these γ_w subpackets is not identical, which calls for γ_w different queries, customized for each subpacket. An example setting for case 2 is given in Figure 5.2, where $y = \ell_r^* = 6$ in the storage given in (5.25), and $\ell_w^* = 4$, which results in distinct sets of associated f_is in every $\gamma_w = \frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{\ell_w^*} = 3$ consecutive writing subpackets of size ℓ_w^*. However, the *super subpacket* containing these $\gamma_w = \frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{\ell_w^*} = 3$ regular subpackets keep repeating with the same set of associated f_is. Therefore, we only send the $\gamma_w = \frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{\ell_w^*} = 3$ sub-queries of $\tilde{Q}_n$ once to each database, which will be repeatedly used throughout the writing process. The general writing scheme that writes to each of the γ_w consecutive subpackets is described in the next steps.

2. Let $J_w^{[s]}$ be the set of indices of the correctly updated $\lfloor \frac{N}{2} \rfloor - 1$ parameters in subpacket s for $s \in \{1, \ldots, \gamma_w\}$. Then, the sub-query s, $s \in \{1, \ldots, \gamma_w\}$ of the writing query for database n is given by,

$$\tilde{Q}_n(s) = \begin{bmatrix} \frac{1}{f_{g((s-1)\ell_w^*+1)-\alpha_n}} e_M(\theta) 1_{\{1 \in J_w^{[s]}\}} + \hat{Z}_{s,1} \\ \vdots \\ \frac{1}{f_{g(s\ell_w^*)-\alpha_n}} e_M(\theta) 1_{\{\ell_w^* \in J_w^{[s]}\}} + \hat{Z}_{s,\ell_w^*} \end{bmatrix}, \tag{5.29}$$

where $\hat{Z}$ are random noise vectors of size $M \times 1$ and the function $g(\cdot)$ is

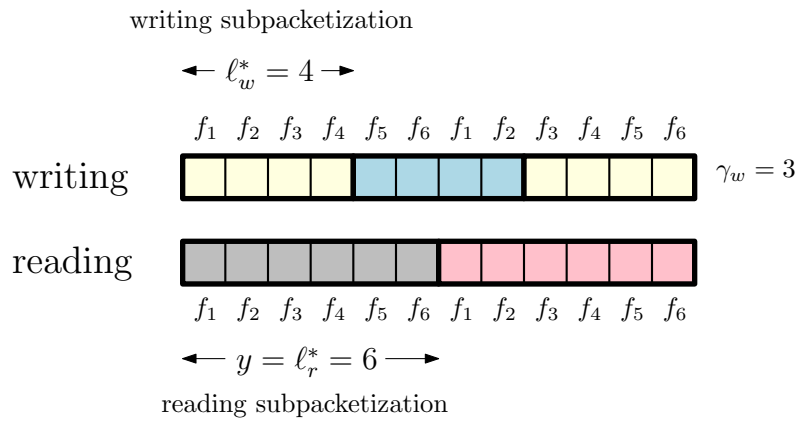


Figure 5.2: An example setting for case 2.

defined as (5.10). For the example considered in Figure 5.2, the sub-query corresponding to subpacket 2 if $J_w^{[2]} = \{1, 3\}$ is given by,

$$\tilde{Q}_n(2) = \begin{bmatrix} \frac{1}{f_5 - \alpha_n} e_M(\theta) + & \hat{Z}_{2,1} \\ & \hat{Z}_{2,2} \\ \frac{1}{f_1 - \alpha_n} e_M(\theta) + & \hat{Z}_{2,3} \\ & \hat{Z}_{2,4} \end{bmatrix}. \tag{5.30}$$

Note that the values of f_i in each individual section of $\tilde{Q}_n$ are distinct due to $\ell_w^* \leq y$ (in the example, the first section has $f_i = \{1, 2, 3, 4\}$ and the second has $f_i = \{5, 6, 1, 2\}$ and so on). This makes it possible for the user to send a single combined update bit (combining the updates of the $\lfloor \frac{N}{2} \rfloor - 1$ non-zero updates in each subpacket) to each individual subpacket as described in Section 3.4. The query $\tilde{Q}_n$ (consisting of γ_w sub-queries) will only be sent once to each database. Therefore, the indices of the non-zero updates $J_w^{[s]}$, $s \in \{1, \ldots, \gamma_w\}$ will be fixed at each consecutive non-overlapping group of γ_w subpackets.

3. Next, the user sends a single combined update bit corresponding to each subpacket. The γ_w combined updates sent to database n, $n \in \{1, \ldots, N\}$ corresponding to a given set of γ_w consecutive subpackets is given by,

$$U_n(s) = \sum_{i \in J_w^{[s]}} \tilde{\Delta}_{\theta,i}^{[s]} \prod_{j \in J_w^{[s]}, j \neq i} (f_{g((s-1)\ell_w^* + j)} - \alpha_n) + \prod_{j \in J_w^{[s]}} (f_{g((s-1)\ell_w^* + j)} - \alpha_n) Z_s, \tag{5.31}$$

for each subpacket $s \in \{1, \ldots, \gamma_w\}$, where $\tilde{\Delta}_{\theta,i}^{[s]} = \frac{\Delta_{\theta,i}^{[s]}}{\prod_{j \in J_w^{[s]}, j \neq i} (f_{g((s-1)\ell_w^* + j)} - f_{g((s-1)\ell_w^* + i)})}$ with $\Delta_{\theta,i}^{[s]}$ being the update of the ith parameter of subpacket s of submodel θ and Z_s are random noise bits. Note that each $U_n(s)$ is a polynomial in α_n of degree $\lfloor \frac{N}{2} \rfloor - 1$. For the example in Figure 5.2, the combined update corresponding to subpacket 2 with $J_w^{[2]} = \{1, 3\}$ is given by,

$$U_n(2) = \tilde{\Delta}_{\theta,1}^{[2]} (f_1 - \alpha_n) + \tilde{\Delta}_{\theta,3}^{[2]} (f_5 - \alpha_n) + (f_1 - \alpha_n)(f_5 - \alpha_n) Z_2, \tag{5.32}$$

where $\tilde{\Delta}_{\theta,1}^{[2]} = \frac{\Delta_{\theta,1}^{[2]}}{f_1 - f_5}$ and $\tilde{\Delta}_{\theta,3}^{[2]} = \frac{\Delta_{\theta,3}^{[2]}}{f_5 - f_1}$.

4. Each database then calculates the incremental update of each subpacket as follows. The incremental update of subpacket s, $s \in \{1, \ldots, \gamma_w\}$ is given by,

$$\tilde{U}_n(s) = \begin{cases} U_n(s) \times \tilde{Q}_n(s), & \text{even } N \\ \tilde{\Omega}_n(s) \times U_n(s) \times \tilde{Q}_n(s), & \text{odd } N \end{cases} \tag{5.33}$$

where,

$$\tilde{\Omega}_n(s) = \begin{bmatrix} \frac{\alpha_r-\alpha_n}{\alpha_r-f_{g((s-1)\ell_w^*+1)}} I_M & 0 & \dots & 0 \\ 0 & \frac{\alpha_r-\alpha_n}{\alpha_r-f_{g((s-1)\ell_w^*+2)}} I_M & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{\alpha_r-\alpha_n}{\alpha_r-f_{g(s\ell_w^*)}} I_M \end{bmatrix}. \quad (5.34)$$

Then, from (5.33),

$$\tilde{U}_n(s) = \begin{cases} \begin{bmatrix} \frac{\Delta_{\theta,1}^{[s]}}{f_{g((s-1)\ell_w^*+1)}-\alpha_n} e_M(\theta) 1_{\{1\in J_w^{[s]}\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor - 1) \\ \vdots \\ \frac{\Delta_{\theta,\ell_w^*}^{[s]}}{f_{g(s\ell_w^*)}-\alpha_n} e_M(\theta) 1_{\{\ell_w^*\in J_w^{[s]}\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor - 1) \end{bmatrix}, & \text{even } N \\ \begin{bmatrix} \frac{\Delta_{\theta,1}^{[s]}}{f_{g((s-1)\ell_w^*+1)}-\alpha_n} e_M(\theta) 1_{\{1\in J_w^{[s]}\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor) \\ \vdots \\ \frac{\Delta_{\theta,\ell_w^*}^{[s]}}{f_{g(s\ell_w^*)}-\alpha_n} e_M(\theta) 1_{\{\ell_w^*\in J_w^{[s]}\}} + P_{\alpha_n}(\lfloor \frac{N}{2} \rfloor) \end{bmatrix}, & \text{odd } N \end{cases} \quad (5.35)$$

where r is a randomly chosen database out of the N databases for odd N. Note that when N is odd, the user can reduce the writing cost by not sending the combined updates to database r, since $\tilde{U}_r(s) = 0$ for all s. The convention for the updates of each $i \notin J_w^{[s]}$ is $\Delta_{\theta,i}^{[s]} = 0$. Lemmas 3.1 and 3.2 are used to obtain (5.35) from (5.33). Note that the concatenation of all γ_w incremental updates of the form (5.35) is in the same format as the concatenation of $\eta = \frac{\text{lcm}\{\ell_r^*, \ell_w^*\}}{y}$ reading subpackets (storage in (5.25)) since $g(\gamma_w \ell_w^*) = g(\text{lcm}\{\ell_r^*, \ell_w^*\}) = y$,

and therefore, can be added to the corresponding subpackets to obtain their updated versions, i.e.,

$$[S_n^{[t]}(1), \ldots, S_n^{[t]}(\eta)]^T = [S_n^{[t-1]}(1), \ldots, S_n^{[t-1]}(\eta)]^T + [\tilde{U}_n(1), \ldots, \tilde{U}_n(\gamma_w)]^T, \tag{5.36}$$

where $[S_n^{[t]}(1), \ldots, S_n^{[t]}(\eta)]^T$ contains η consecutive S_ns of the form given in (5.25).

The writing cost of case 2 is given by,

$$C_W^{[2]} = \begin{cases} \frac{\gamma_w \times N}{\gamma_w \times \ell_w^*} = \frac{N}{\ell_w^*}, & \text{even } N, \\ \frac{\gamma_w \times (N-1)}{\gamma_w \times \ell_w^*} = \frac{N-1}{\ell_w^*}, & \text{odd } N. \end{cases} \tag{5.37}$$

Remark 5.3 *For even N, both cases achieve reading and writing costs given by $\frac{N}{\ell_r^*}$ and $\frac{N}{\ell_w^*}$, respectively. However, when N is odd, it is possible to achieve either a lower reading cost ($\frac{N-1}{\ell_r^*}$) with fewer noise terms in storage ($\lfloor \frac{N}{2} \rfloor - 1$), or a lower writing cost ($\frac{N-1}{\ell_w^*}$) with an extra noise term in storage ($\lceil \frac{N}{2} \rceil - 1$), with case 1 and case 2, respectively. In particular, when N is odd, the total costs for the two options are given by $\frac{N-1}{\ell_r^*} + \frac{N}{\ell_w^*} = \frac{N(\ell_r^* + \ell_w^*)}{\ell_r^* \ell_w^*} - \frac{1}{\ell_r^*}$ and $\frac{N}{\ell_r^*} + \frac{N-1}{\ell_w^*} = \frac{N(\ell_r^* + \ell_w^*)}{\ell_r^* \ell_w^*} - \frac{1}{\ell_w^*}$, respectively. This justifies the extra noise term in storage for case 2 when N is odd.*

Remark 5.4 *Note that the cost of sending Q_n and $\tilde{Q}_n$ is not considered in the above writing cost since they are sent only once to each database in the entire PRUW process (not per subpacket) and the maximum combined cost of Q_n and $\tilde{Q}_n$ given by*

$\frac{M}{L}(lcm\{\ell_r^*, \ell_w^*\} + \max\{\ell_r^*, \ell_w^*\})$ *is negligible since* L *is very large.*

5.4.3 Calculation of Optimum ℓ_r^* and ℓ_w^* for Given $(\tilde{D}_r, \tilde{D}_w)$

In order to minimize the total communication cost, the user correctly reads from and writes to only $\lfloor \frac{N}{2} \rfloor - 1$ out of each of the ℓ_r^* and ℓ_w^* bits in reading and writing phases, respectively. This results in an error that needs to be kept within the given distortion budgets of $\tilde{D}_r$ and $\tilde{D}_w$. Note that $\min C_R + \min C_W \leq \min C_R + C_W$. In this section, we find the subpacketizations in the reading and writing phases (ℓ_r^*, ℓ_w^*) that achieve $\min C_R + \min C_W$ while being compatible with the proposed scheme. Note that each reading/writing cost in both cases is of the form $\frac{N}{\ell}$ or $\frac{N-1}{\ell}$, where ℓ is the respective subpacketization. Since only $\lfloor \frac{N}{2} \rfloor - 1$ bits in a subpacket are read/written correctly, the subpacketization in general can be written as,

$$\ell = \lfloor \frac{N}{2} \rfloor - 1 + i \tag{5.38}$$

for some $i \in \mathbb{Z}_0^+$. Therefore, the reading/writing costs of both cases are of the form $\frac{N}{\lfloor \frac{N}{2} \rfloor - 1 + i}$ or $\frac{N-1}{\lfloor \frac{N}{2} \rfloor - 1 + i}$ for some $i \in \mathbb{Z}_0^+$, both decreasing in i. For a subpacketization of the form $\ell = \lfloor \frac{N}{2} \rfloor - 1 + i$ (irrespective of reading or writing), the resulting distortion is given by,

$$D = \frac{i}{\lfloor \frac{N}{2} \rfloor - 1 + i}, \tag{5.39}$$

if the same subpacketization is considered throughout the storage. Since the resulting distortion must satisfy $D \leq \tilde{D}$,[5] an upper bound on i is derived as,

$$i \leq \frac{\tilde{D}}{1-\tilde{D}}\left(\lfloor\frac{N}{2}\rfloor - 1\right). \tag{5.40}$$

Therefore, for given distortion budgets in the reading and writing phases ($\tilde{D}_r$,$\tilde{D}_w$), the optimum values of i are given by,

$$i_r^* = \frac{\tilde{D}_r}{1-\tilde{D}_r}\left(\lfloor\frac{N}{2}\rfloor - 1\right) \tag{5.41}$$

$$i_w^* = \frac{\tilde{D}_w}{1-\tilde{D}_w}\left(\lfloor\frac{N}{2}\rfloor - 1\right), \tag{5.42}$$

which determine the optimum subpacketizations from (5.38). For cases where $i_r^* \notin \mathbb{Z}_0^+$ or $i_w^* \notin \mathbb{Z}_0^+$, we divide all submodels into two sections, assign two separate integer-subpacketizations that guarantee the distortion budget, and apply the scheme on the two sections independently, which achieves the minimum costs in (5.5)-(5.6), after using an optimum ratio for the subsection lengths. To find the optimum ratio, we solve the following optimization problem. Let λ_i be the fraction of each submodel with subpacketization $\ell_i = \lfloor\frac{N}{2}\rfloor - 1 + i$ for some $i = \eta_1, \eta_2 \in \mathbb{Z}_0^+$. In this calculation, we drop the r and w subscripts which indicate the phase (reading/writing), since the calculation is the same for both phases.[6] The given $\tilde{D}_r$ and $\tilde{D}_w$ must be substituted for $\tilde{D}$ in the following calculation to obtain the specific

[5]Here, $\tilde{D}$ refers to $\tilde{D}_r$ or $\tilde{D}_w$, based on the phase the subpacketization is defined for.

[6]Note that we focus on minimizing each individual cost (reading/writing cost) at a time since $\min C_R + \min C_W \leq \min C_R + C_W$.

results for the reading and writing phases, respectively. The optimum subpacketizations are obtained by solving,[7]

$$\begin{aligned}
\min \quad & \sum_{i=\eta_1,\eta_2} \lambda_i \frac{N}{\lfloor \frac{N}{2} \rfloor - 1 + i} \\
\text{s.t.} \quad & \sum_{i=\eta_1,\eta_2} \lambda_i \frac{i}{\lfloor \frac{N}{2} \rfloor - 1 + i} \leq \tilde{D} \\
& \lambda_{\eta_1} + \lambda_{\eta_2} = 1 \\
& \lambda_{\eta_1}, \lambda_{\eta_2} \geq 0.
\end{aligned} \tag{5.43}$$

This problem has multiple solutions that give the same minimum total communication costs. As one of the solutions, consider $\eta_1 = 0$ and $\eta_2 = \eta$, where $\eta = \lceil \frac{\tilde{D}}{1-\tilde{D}}(\lfloor \frac{N}{2} \rfloor - 1) \rceil$,

$$\lambda_0 = 1 - \frac{\tilde{D}}{\eta}\left(\lfloor \frac{N}{2} \rfloor - 1 + \eta\right), \tag{5.44}$$

$$\lambda_\eta = \frac{\tilde{D}}{\eta}\left(\lfloor \frac{N}{2} \rfloor - 1 + \eta\right). \tag{5.45}$$

This gives a minimum cost of $C_{\min} = \frac{N}{\lfloor \frac{N}{2} \rfloor - 1}(1 - \tilde{D})$ (or $C_{\min} = \frac{N-1}{\lfloor \frac{N}{2} \rfloor - 1}(1 - \tilde{D})$) which match the terms in (5.5)-(5.6), with $\tilde{D} = \tilde{D}_r$ and $\tilde{D} = \tilde{D}_w$. The optimality of the solution to the optimization problem is obvious since the resulting total cost is the same as what is achieved by the optimum subpacketizations characterized by (5.41) and (5.42), with no segmentation of submodels.

Next, we present the explicit expressions of optimum subpacketizations, with

[7]Even though there are two types of reading and writing costs costs ($\frac{N}{\lfloor \frac{N}{2} \rfloor - 1 + i}$ and $\frac{N-1}{\lfloor \frac{N}{2} \rfloor - 1 + i}$), the optimization problem remains the same since the two costs are scaled versions of one another.

the optimum values of i obtained above. For a setting with given N, $\tilde{D}_r$ and $\tilde{D}_w$, the reading and writing costs given in (5.5)-(5.6) are achievable with corresponding subpacketizations given by,

$$\ell_r^* = \begin{cases} \lfloor \frac{N}{2} \rfloor - 1, & \text{for } \lambda_0^{[r]} \text{ of submodel,} \\ \lfloor \frac{N}{2} \rfloor - 1 + \lceil \frac{\tilde{D}_r \left(\lfloor \frac{N}{2} \rfloor - 1 \right)}{1 - \tilde{D}_r} \rceil, & \text{for } 1 - \lambda_0^{[r]} \text{ of submodel,} \end{cases} \tag{5.46}$$

and

$$\ell_w^* = \begin{cases} \lfloor \frac{N}{2} \rfloor - 1, & \text{for } \lambda_0^{[w]} \text{ of submodel,} \\ \lfloor \frac{N}{2} \rfloor - 1 + \lceil \frac{\tilde{D}_w \left(\lfloor \frac{N}{2} \rfloor - 1 \right)}{1 - \tilde{D}_w} \rceil, & \text{for } 1 - \lambda_0^{[w]} \text{ of submodel,} \end{cases} \tag{5.47}$$

where $\lambda_0^{[r]}$ and $\lambda_0^{[w]}$ are λ_0 in (5.44) with $\tilde{D}$ replaced by $\tilde{D}_r$ and $\tilde{D}_w$, respectively. Once the subpacketizations of both reading and writing phases are determined based on the given distortion budgets, each section of all submodels is assigned a *case*, based on the corresponding values of ℓ_r^* and ℓ_w^*, which determines the specific form of storage from either (5.7) or (5.25). An example setting is shown in Figure 5.3. Assume that the subpacketizations satisfy $\ell_1 < \ell_2 < \ell_3$, and therefore, for example the middle section which has a reading subpacketization of ℓ_2 and a writing subpacketization of ℓ_1 satisfying $\ell_1 < \ell_2$, belongs to case 2 by definition.

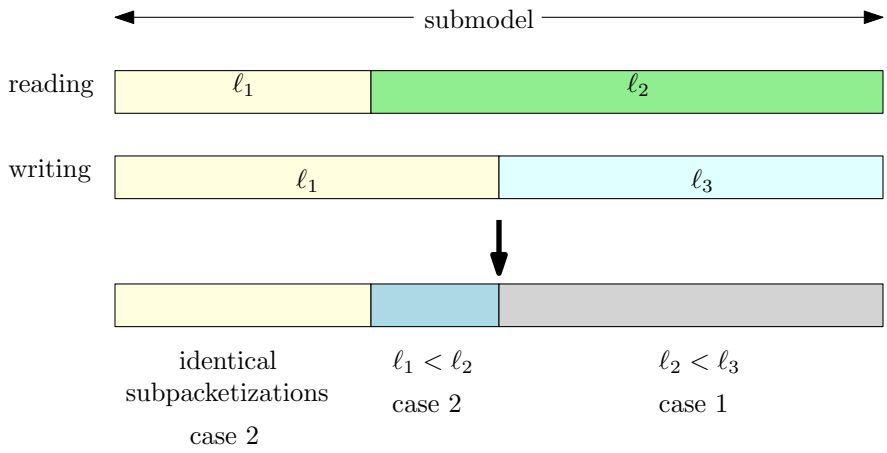


Figure 5.3: Storage of submodels

5.4.4 Proof of Privacy

The structures and sizes of the queries, updates and storage are determined at the initialization stage (when the subpacketizations are calculated and the storage is initialized), based on the given distortion budgets in the proposed scheme, and do not depend on each user's updating submodel index or the values of sparse updates. Moreover, the queries Q_n, updates U_n and storage S_n in this scheme are random noise terms that are independent of the values and positions of the sparse updates as well as the updating submodel index. Therefore, the proofs presented in Section 3.4.4 for the privacy of submodel index, privacy of values of updates and security of submodels are valid in this section as well.

5.5 Conclusions

In this chapter, we considered the problem of PRUW in FSL with random sparsification, where each user only reads and writes a randomly selected set of parameters

and updates in the learning process to reduce the communication cost. The problem is formulated in terms of a rate-distortion characterization as the unread/unwritten parameters and updates introduce a certain amount of distortion. As the main result of this work, we showed that a linear rate-distortion characterization is achievable, and proposed a scheme that minimizes the total communication cost (within the scope of CSA) for given amounts of distortion allowed in the reading and writing phases. The resulting asymptotic normalized reading and writing costs are both equal to $2r$, where $r = 1 - \tilde{D}$, where $\tilde{D}$ is the distortion allowed. Since a fraction of $\tilde{D}$ parameters of the entire submodel are not read/updated, the sparsification rate for this case is $r = 1 - \tilde{D}$. It is clear that random sparsification outperforms (or performs equally) top r sparsification in terms of the communication cost when similar sparsification rates are considered. However, random sparsification may not be as effective as top r sparsification since it does not capture the most significant variations of the gradients in the stochastic gradient descent (SGD) process, in relation to the underlying learning task. This may have an adverse effect on the model convergence time as well as on the accuracy of the trained model.

Private FSL with top r sparsification considered in Chapter 4 also results in incomplete downloads/uploads, as only a selected set of subpackets are downloaded/updated. However, these parameters and updates are carefully chosen based on their significance to improve the accuracy. It has been shown in certain cases that top r sparsification outperforms non-sparse distributed learning. Therefore, we do not consider the ignored subpackets in the reading and writing phases in top r sparsification as distortion. However, in random sparsification, since the selected

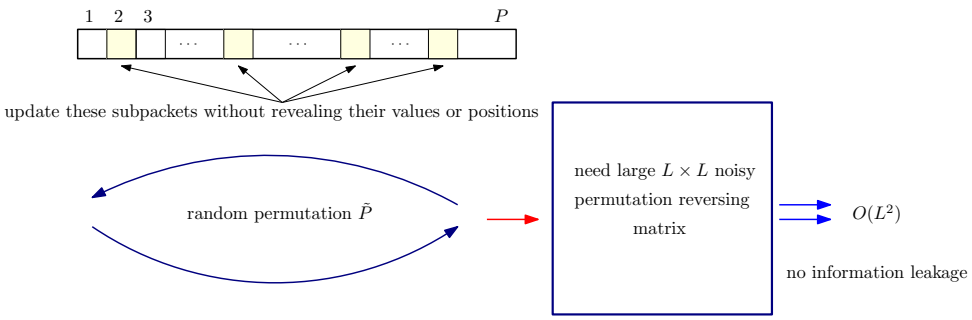
parameters/updates are chosen randomly, we treat the ignored parameters/updates as distortion, to characterize the rate-distortion trade-off in PRUW.

CHAPTER 6

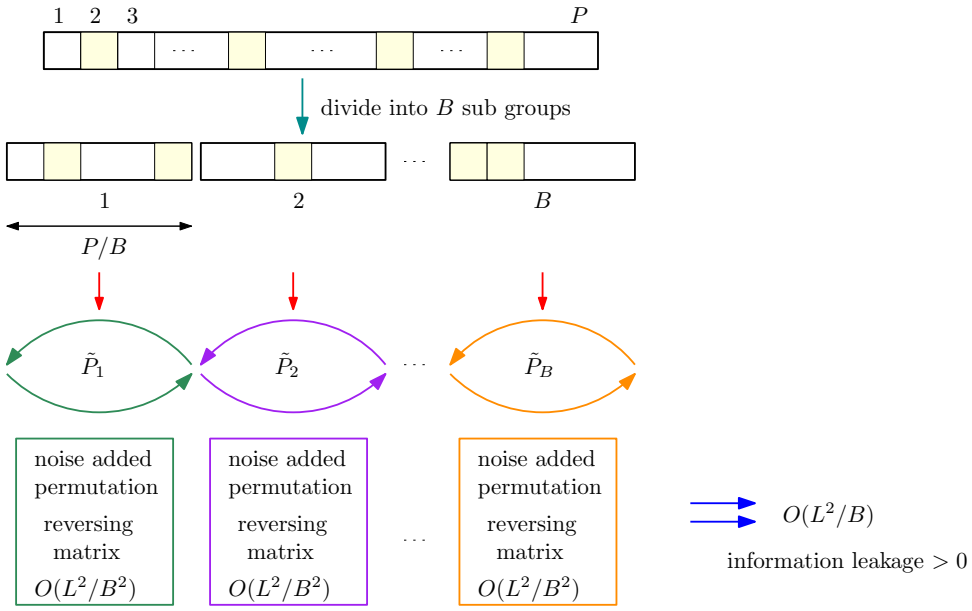
Weakly Private Read-Update-Write (PRUW) in Federated Learning (FL) with Top r Sparsification

6.1 Introduction

In this chapter, we consider the problem of PRUW in FL with top r sparsification. The main goal of this work is to develop schemes that perform the user-database communications in FL with top r sparsification while guaranteeing information-theoretic privacy of the values and the indices of the sparse updates/parameters. For this, we use the same permutation technique introduced in Chapter 4, which however incurs a significantly large storage cost in FL, compared to FSL. To that end, we propose schemes that reduce the storage cost at the expense of a given amount of information leakage. This is achieved by dividing the ML model into multiple segments and carrying out permutations within each segment. This is illustrated in Fig. 6.1. The number of segments is chosen based on the allowed amount of information leakage and the storage capacity of the databases. In general, this chapter presents the trade-off between the communication cost, storage complexity and information leakage in private FL with sparsification.



(a) Without segmentation.



(b) With segmentation.

Figure 6.1: Motivation for segmentation in permutation techniques: (a) Permutation of the entire model without segmentation. (b) Permutation within segments with segmentation.

6.2 Problem Formulation

We consider a FL setting in which a ML model consisting of L parameters belonging to P subpackets is stored in N non-colluding databases. The parameters take values from a large enough finite field $\mathbb{F}_q$. A given user at a given time t reads (downloads) the required parameters of the model from the databases, trains the model using

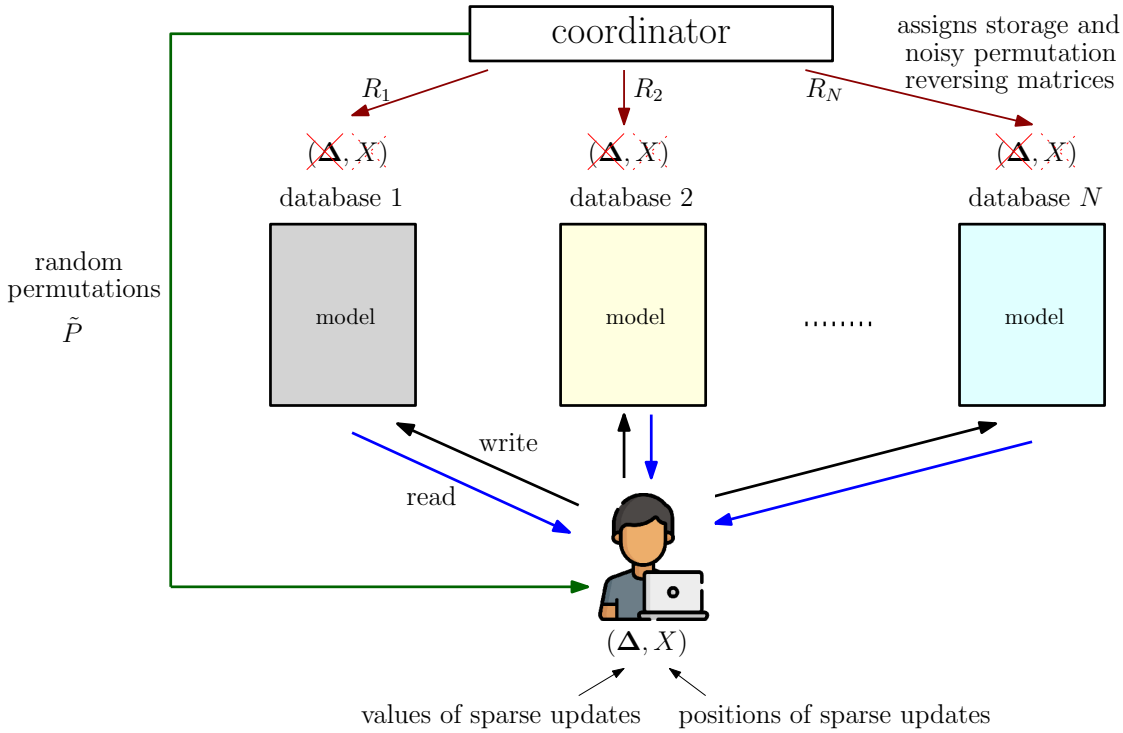


Figure 6.2: System model: A user reads (downloads), updates, writes (uploads) a ML model.

the user's local data, and writes (uploads) the most significant r fraction of updates back to all databases. In this work, we consider sparsification in both uplink and downlink, to reduce the communication cost. In particular, the sparsification rates of the reading (downlink) and writing (uplink) phases are given by r' and r, respectively. In other words, in the reading (download) phase, the users only download a selected set of Pr' subpackets determined by the databases.[1] Once the model is trained locally, each user only uploads the most significant Pr set of updates (corresponding subpackets) to the databases in the writing phase.[2] The system model is shown in Fig. 6.2, where the coordinator is used to initialize the process.

[1]These subpackets could be determined by the databases based on the sparse updates received at the previous time step, or by any other downlink sparsification protocol. For example, the databases can choose the most commonly updated Pr' subpackets in the writing phase of time $t-1$ to be sent to the users in the reading phase at time t.

[2]We assume that all values in the sparse set of Pr subpackets in the writing phase are non-zero.

Note that the users send no information to the databases in the reading phase. Therefore, no information about the user's local data is leaked to the databases in the reading phase. The users send the sparse updates and their positions (indices) to the databases in the writing phase to train the model. Information about the user's local data can be leaked to the databases from these updates and their indices.[3] In this work, we consider the following privacy guarantees on the values and the indices of the sparse updates.

Privacy of the values of sparse updates: No information on the values of the sparse updates is allowed to leak to any of the databases, i.e.,

$$I(\Delta_i^{[t]}; G_n^{[t]}) = 0, \quad n \in \{1, \ldots, N\}, \quad \forall i \tag{6.1}$$

where $\Delta_i^{[t]}$ is the value of the ith sparse (non-zero) update of a given user at time t and $G_n^{[t]}$ contains all the information sent by the user to database n at time t.

Privacy of the positions (indices) of sparse updates: The amount of information leaked on the indices of the sparse updates need to be maintained under a given privacy leakage budget ϵ, i.e.,

$$I(X^{[t]}; G_n^{[t]}) \leq \epsilon, \quad n \in \{1, \ldots, N\}, \tag{6.2}$$

where $X^{[t]}$ is the set of indices of the sparse subpackets updated by a given user at time t. The system model with the privacy constraints is shown in Fig. 6.2.

[3]The positions (indices) of the sparse updates leak information about the most and least significant parameters in the model for a given user, which may leak information about the user's local data.

A coordinator is used to initialize the FL process.[4] In addition to the privacy constraints, we require the following security and correctness conditions for the reliability of the scheme.

Security of the model: No information about the model parameters is allowed to leak to the databases, i.e.,

$$I(W^{[t]}; S_n^{[t]}) = 0, \quad n \in \{1, \ldots, N\}, \tag{6.3}$$

where $W^{[t]}$ is the ML model and $S_n^{[t]}$ is the data content in database n at time t.

Correctness in the reading phase: The user should be able to correctly decode the sparse set of subpackets (denoted by J) of the model, determined by the downlink sparsification protocol, from the downloads in the reading phase, i.e.,

$$H(W_J^{[t-1]} | A_{1:N}^{[t]}) = 0, \tag{6.4}$$

where $W_J^{[t-1]}$ is the set of subpackets in set J of the model W at time $t-1$ (before updating) and $A_n^{[t]}$ is the information downloaded from database n at time t.

Correctness in the writing phase: Let J' be the set of most significant Pr subpackets of the model, updated by a given user at time t. The model should be

[4]The coordinator is only available at the initialization stage, and will not be part of the system model once the FL process begins.

correctly updated as,

$$W_s^{[t]} = \begin{cases} W_s^{[t-1]} + \Delta_s^{[t]}, & \text{if } s \in J' \\ W_s^{[t-1]}, & \text{if } s \notin J' \end{cases}, \tag{6.5}$$

where $W_s^{[t-1]}$ is subpacket s of the model at time $t-1$ and $\Delta_s^{[t]}$ is the corresponding update of subpacket s at time t.

Reading and writing costs: The reading and writing costs are defined as $C_R = \frac{\mathcal{D}}{L}$ and $C_W = \frac{\mathcal{U}}{L}$, respectively, where $\mathcal{D}$ is the total number of symbols downloaded in the reading phase, $\mathcal{U}$ is the total number of symbols uploaded in the writing phase, and L is the size of the model. The total cost C_T is the sum of the reading and writing costs $C_T = C_R + C_W$.

Storage complexity: The storage complexity is quantified by the order of the total number of symbols stored in each database.

In this work, we propose schemes to perform FL with top r sparsification, that result in the minimum total communication costs and storage complexities, while satisfying all privacy, security and correctness conditions described above.

6.3 Main Result

Theorem 6.1 *Consider a FL model stored in N non-colluding databases, consisting of L parameters with values from a finite field $\mathbb{F}_q$, which are included in P subpackets. The model is divided into B segments of equal size ($1 \leq B < P$), such that each consecutive $\frac{P}{B}$ subpackets constitute each segment. Assume that the FL model*

is being updated by users at each time instance with uplink and downlink sparsification rates (top r sparsification) of r and r', respectively. Let $\hat{X}_i$ be the random variable representing the number of subpackets with sparse (non-zero) updates in the ith segment, uploaded by any given user, and let $(\tilde{X}_1, \ldots, \tilde{X}_B)$ be the general vector representing all distinct combinations of $(\hat{X}_1, \ldots, \hat{X}_B)$, irrespective of the segment index. Then, the reading/writing costs, storage complexities and amounts of information leakage presented in Table 6.1 are achievable in a single round of the FL process in the perspective of a single user.

case	reading cost	writing cost	storage complexity	information leakage
1	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{2}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{2}{N}}$	$O(\frac{L^2}{B})$	$H(\hat{X}_1, \ldots, \hat{X}_B)$
2	$\frac{3r'(1+\frac{\log_q P}{N})}{1-\frac{1}{N}}$	$\frac{3r(1+\log_q P)}{1-\frac{1}{N}}$	$O(\frac{L^2}{BN^2})$	$H(\hat{X}_1, \ldots, \hat{X}_B)$
3	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{4}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{4}{N}}$	$\max\{O(\frac{L^2}{B}), O(N^2B^2)\}$	$H(\tilde{X}_1, \ldots, \tilde{X}_B)$
4	$\frac{5r'(1+\frac{\log_q P}{N})}{1-\frac{1}{N}}$	$\frac{5r(1+\log_q P)}{1-\frac{1}{N}}$	$\max\{O(\frac{L^2}{N^2B}), O(B^2)\}$	$H(\tilde{X}_1, \ldots, \tilde{X}_B)$

Table 6.1: Achievable sets of communication costs, storage costs and information leakage.

Remark 6.1 *The information leakage in Table 6.1 corresponds to the amount of information leaked on the indices of the sparse updates.*[5] *For a given privacy leakage budget on the indices of the sparse updates given by ϵ, the optimum number of segments B can be calculated by minimizing the storage complexity, such that $H(\hat{X}_1, \ldots, \hat{X}_B) < \epsilon$ or $H(\tilde{X}_1, \ldots, \tilde{X}_B) < \epsilon$ is satisfied (based on the considered case). This is valid for all four cases.*

Remark 6.2 *When $B = 1$ (no segmentation present), $\hat{X}_1 = \tilde{X}_1 = Pr$ and the*

[5]Information theoretic privacy of the values of updates is guaranteed, as stated in the problem formulation.

corresponding information leakage is zero since Pr is fixed and $H(\hat{X}_1) = H(\tilde{X}_1) = 0$, i.e., the four schemes corresponding to the four cases achieve information theoretic privacy of the values and positions of the sparse updates while incurring the same communication costs stated in Table 6.1, when $B = 1$. However, in this case, the storage costs increase to either $O(L^2)$ or $O\left(\frac{L^2}{N^2}\right)$.

Remark 6.3 *$H(\hat{X}_1, \ldots, \hat{X}_B) > H(\tilde{X}_1, \ldots, \tilde{X}_B)$ since $H(\hat{X}_1, \ldots, \hat{X}_B)$ considers all possible values of $\hat{X}_i$, while $H(\tilde{X}_1, \ldots, \tilde{X}_B)$ only considers distinct sets of $\{\hat{X}_i\}_{i=1}^{B}$. For example, if $B = 2$, $H(\hat{X}_1, \hat{X}_2)$ considers both permutations $\{1, 2\}$ and $\{2, 1\}$ while $H(\tilde{X}_1, \tilde{X}_2)$ only takes one of them into account, i.e., the probabilities considered in $H(\tilde{X}_1, \ldots, \tilde{X}_B)$ are more dense and concentrated compared to that of $H(\hat{X}_1, \ldots, \hat{X}_B)$.*

Remark 6.4 *Cases 1-4 are achieved by schemes that utilize both CSA [97] and permutation techniques which are described in detail in Section 6.4. The schemes for cases 1 and 2 use a single round permutation technique (only within-segment permutations) while cases 3 and 4 use a two-round permutation technique (both within and inter-segment permutations) which reduces the information leakage further. Cases 3 and 4 are extensions of cases 1 and 2, respectively, with the additional permutation round. Cases 3 and 4 incur larger communication costs compared to cases 1 and 2, while resulting in lower amounts of information leakage.*

Remark 6.5 *The four cases (schemes) have different properties. Cases 1 and 3 result in the lowest communication costs at the expense of a larger storage complexity resulted by replicated storage and larger noisy permutation reversing matrices. Cases*

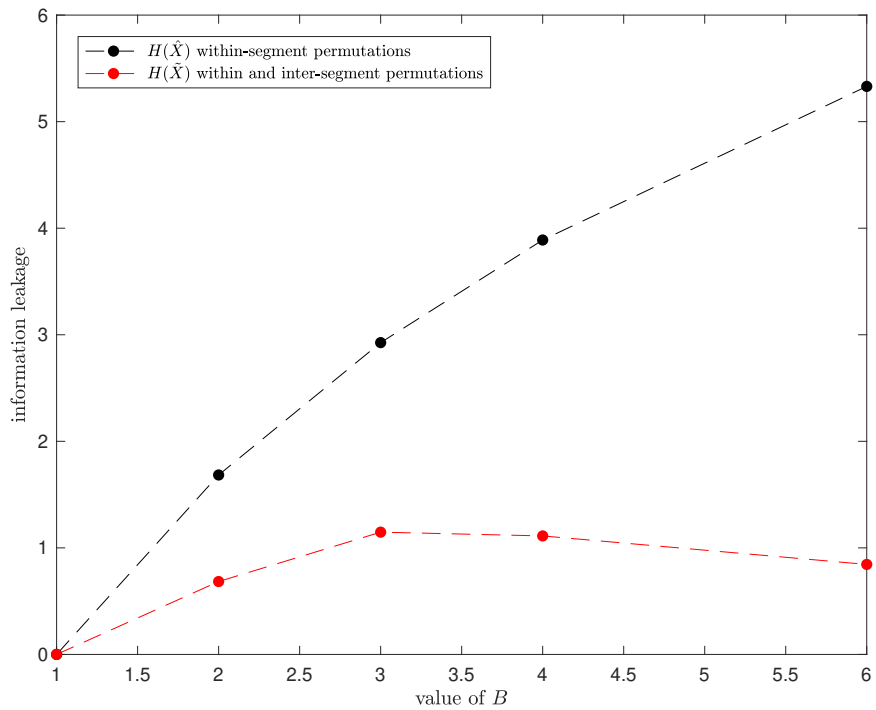


Figure 6.3: Information leakage of an example setting with $P = 12$ and different values of B.

2 and 4 use MDS coded storage and compact permutation reversing matrices, which reduces the storage complexity at the expense of larger communication costs.

Remark 6.6 *The communication cost does not depend on the number of segments B.*

Remark 6.7 *Consider an example setting with $P = 12$ subpackets divided into $B = 1, 2, 3, 4, 6$ segments. Assume that each subpacket is equally probable to be selected to the set of most significant $Pr = 3$ subpackets. The behavior of the information leakage for each value of B is shown in Fig. 6.3.*

6.4 Proposed Schemes

In this section, we present four schemes that perform private FL with top r sparsification, which have different properties. The schemes differ from each other based on their storage structure (MDS coded or uncoded) and the number of permutation stages.

6.4.1 General Schemes With Examples

In this section, we provide the proposed schemes for all four cases. In all four schemes, we divide the P subpackets into B non-overlapping equal-sized segments to control the storage cost and the information leakage. The parameter B is a variable that can be chosen based on the given privacy leakage budget and the limitations on the storage capacities. In this section, we present the general schemes for arbitrary values of B, P, r and r'. As a further illustration, we provide examples along with the general scheme for all four cases. In the two examples corresponding to cases 1 and 2, we assume the same setting with $P = 15$ subpackets (subpacketization ℓ), divided into $B = 3$ equal segments as shown in Fig. 6.4.

Case 1: Uncoded[6] storage and larger permutation reversing matrices are used in this case to reduce the communication cost, at the expense of a larger storage cost.

Initialization: A single subpacket (subpacket s) in case 1 is stored in database

[6]Even though the model parameters and noise symbols are combined together (coded form) in the storage in (6.6), each parameter is not combined with other parameters, resulting in uncoded storage.

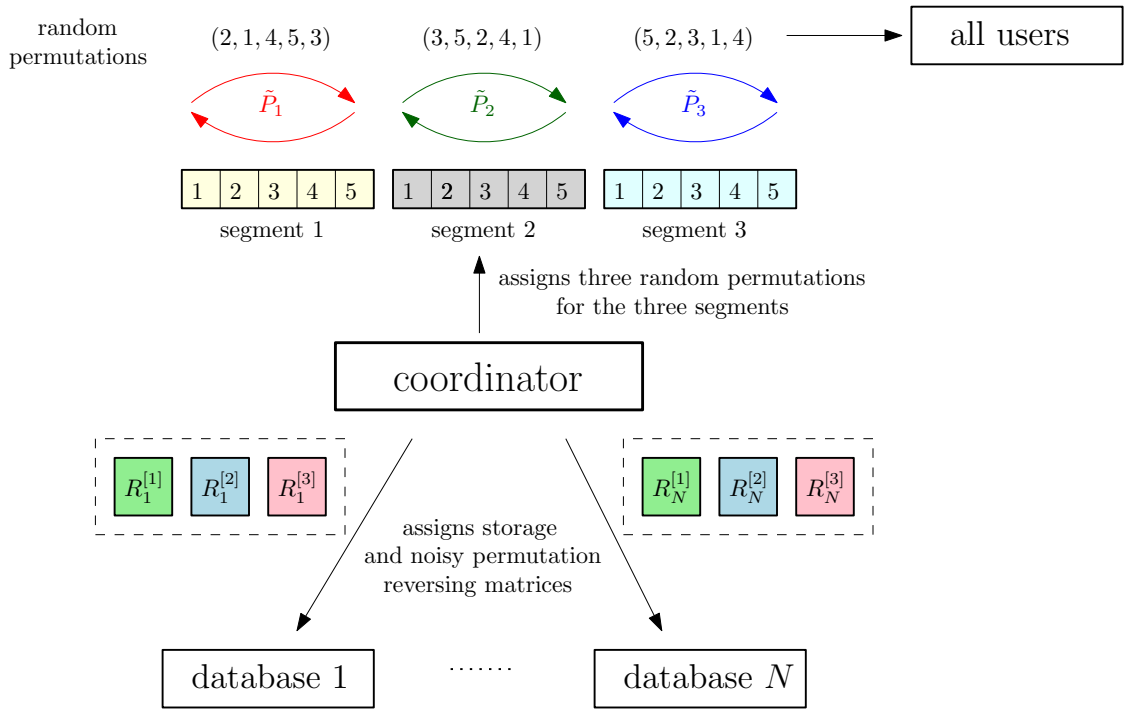


Figure 6.4: Initialization of the scheme for cases 1 and 2.

n, $n \in \{1, \ldots, N\}$ as,

$$S_n^{[s]} = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} W_1^{[s]} + \sum_{j=0}^{\ell} \alpha_n^j Z_{1,j}^{[s]} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} W_\ell^{[s]} + \sum_{j=0}^{\ell} \alpha_n^j Z_{\ell,j}^{[s]} \end{bmatrix}, \tag{6.6}$$

where $W_i^{[s]}$ is the ith parameter of subpacket s, $Z_{i,j}^{[s]}$ are random noise symbols and $\{f_i\}_{i=1}^{\ell}, \{\alpha_n\}_{n=1}^{N}$ are globally known distinct constants from $\mathbb{F}_q$. The subpackets in each segment are stacked one after the other in the order of subpacket 1 through subpacket $\frac{P}{B}$. At the initialization stage, the coordinator sends B ($B = 3$ for the example considered) randomly and independently chosen permutations of the $\frac{P}{B}$ ($\frac{P}{B} = 5$ for the example considered) subpackets in each of the B segments to all users. These permutations are denoted by $\tilde{P}_1, \ldots, \tilde{P}_B$. The coordinator also sends

the B corresponding noise added permutation reversing matrices given by,

$$R_n^{[i]} = (\tilde{R}^{[i]} \otimes \Gamma_n) + \tilde{Z}^{[i]}, \quad i = 1, \ldots, B, \tag{6.7}$$

to database n, $n \in \{1, \ldots, N\}$, as shown in Fig. 6.4, where $\tilde{R}^{[i]}$ is the permutation reversing matrix corresponding to the permutation $\tilde{P}_i$, Γ_n is the diagonal matrix given by,

$$\Gamma_n = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} & & \\ & \ddots & \\ & & \frac{1}{f_\ell - \alpha_n} \end{bmatrix}, \tag{6.8}$$

and $\tilde{Z}^{[i]}$ is a random noise matrix of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$. Based on the example considered, the permutation reversing matrix for database n, $n \in \{1, \ldots, N\}$ corresponding to the first segment (permutation: $\tilde{P}_1 = (2, 1, 4, 5, 3)$) is given by,

$$R_n^{[1]} = \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \otimes \Gamma_n \right) + \tilde{Z}^{[1]} = \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]}, \tag{6.9}$$

Similarly, for the second segment (permutation:$\tilde{P}_2 = (3, 5, 2, 4, 1)$), the permutation

reversing matrix for database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n^{[2]} = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[2]}. \tag{6.10}$$

The coordinator leaves the system once the storage, permutations and noise added permutation reversing matrices are initialized and the system is ready to begin the FL process. All subsequent communications take place only between individual users and databases in terms of permuted subpacket indices. The databases never learn the underlying permutations despite having access to the noise added permutation reversing matrices, since the added noise $\tilde{Z}^{[i]}$ makes the noisy matrices independent of the original permutation reversing matrix from Shannon's one time pad theorem.

Reading Phase: The databases decide the permuted indices of the Pr' sparse subpackets to be sent to the users at time t in the reading phase, based on the permuted subpacket indices received in the writing phase at time $t-1$. For example, the databases consider the permuted indices of the subpackets updated by all users at time $t-1$, and select the most popular Pr' of them to be sent to the users in the reading phase of time t. Note that the databases are unaware of the real indices of the sparse subpacket indices updated by users in the writing phase at each time

instance and only work with the permuted indices in both phases. We denote the permuted indices of the sparse subpackets to be sent to the users from segment j as $\tilde{V}_j$ for $j \in \{1, \ldots, B\}$. For this example, let the sparse set of permuted subpacket indices corresponding to the first segment be $\tilde{V}_1 = \{1, 3\}$.[7] One designated database sends these permuted indices of each segment to the users. The users then find the real indices, using the known permutations as $V_j(i) = \tilde{P}_j(\tilde{V}_j(i))$ for each sparse subpacket i in segment $j \in \{1, \ldots, B\}$. For this example (segment 1), the real set of indices is given by,

$$V_1(i) = \tilde{P}_1(\tilde{V}_1(i)), \quad i = 1, 2 \tag{6.11}$$

$$V_1 = \{2, 4\}. \tag{6.12}$$

In order to send the ith sparse subpacket of segment j, $\tilde{V}_j(i)$, each database n, $n \in \{1, \ldots, N\}$ generates the following query.

$$Q_n^{[\tilde{V}_j(i)]} = \sum_{k=1}^{\ell} R_n^{[j]}(:, (i-1)\ell + k). \tag{6.13}$$

For example, the query corresponding to the first sparse subpacket of the first seg-

[7] Two similar sets (with same or different cardinalities, such that the sum of all three cardinalities equals Pr') exist for segments 2 and 3 as well.

ment (i.e., $\tilde{V}_1(1) = 1$) is given by,

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, k) = \begin{bmatrix} 0_\ell \\ \frac{1}{f_1 - \alpha_n} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1, \tag{6.14}$$

where Z_1 is a random noise vector resulted by the noise component of $R_n^{[1]}$. Similarly, the query for the second sparse subpacket of segment 1 (i.e., $\tilde{V}_1(2) = 3$) is given by,

$$Q_n^{[\tilde{V}_1(2)]} = Q_n^{[3]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, 2\ell + k) = \begin{bmatrix} 0_\ell \\ 0_\ell \\ 0_\ell \\ \frac{1}{f_1 - \alpha_n} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} \\ 0_\ell \end{bmatrix} + Z_2. \tag{6.15}$$

Note that the reversal of the permutations is hidden from the databases by the random noise vectors Z_1 and Z_2. Then, database n, $n \in \{1, \ldots, N\}$ sends the answer corresponding to the ith sparse subpacket of segment j to all users by calculating

the dot product between the queries and the scaled storage as,

$$A_n^{[\tilde{V}_j(i)]} = (D_n \times S_n)^T Q_n^{[\tilde{V}_j(i)]}, \quad j \in \{1, \ldots, B\} \tag{6.16}$$

$$= \frac{1}{f_1 - \alpha_n} W_1^{[V_j(i)]} + \ldots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[V_j(i)]} + P_{\alpha_n}(\ell + 1), \tag{6.17}$$

where D_n is the diagonal matrix of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$ given by,

$$D_n = I_{\frac{P}{B}} \otimes \Gamma_n^{-1} = \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}, \tag{6.18}$$

where $I_{\frac{P}{B}}$ is the identity matrix of size $\frac{P}{B} \times \frac{P}{B}$ and $P_{\alpha_n}(\ell+1)$ is a polynomial in α_n of degree $\ell + 1$. For example, the answer of database n,$n \in \{1, \ldots, N\}$ corresponding to the first sparse subpacket of segment 1 (i.e., $\tilde{V}_1(1) = 1$) is given by,

$$A_n^{[\tilde{V}_1(1)]} = (D_n \times S_n)^T Q_n^{[\tilde{V}_1(1)]} \tag{6.19}$$

$$= \left(\begin{bmatrix} \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[1]} + \sum_{j=0}^{\ell} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[1]} + \sum_{j=0}^{\ell} \alpha_n^j I_{\ell,j} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[5]} + \sum_{j=0}^{\ell} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[5]} + \sum_{j=0}^{\ell} \alpha_n^j I_{\ell,j} \end{bmatrix} \end{bmatrix} \right)^T$$

$$\times \left(\begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1 \right) \tag{6.20}$$

$$= \frac{1}{f_1-\alpha_n} W_1^{[2]} + \ldots + \frac{1}{f_\ell-\alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell+1). \tag{6.21}$$

Now, the users obtain the parameters of real subpacket 2 of segment 1, (i.e., $V_1(1) = \tilde{P}_1(\tilde{V}_1(1)) = 2$) by solving,

$$\begin{bmatrix} A_1^{\tilde{V}_1(1)} \\ \vdots \\ A_N^{\tilde{V}_1(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \ldots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^{\ell+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \ldots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \ldots & \alpha_N^{\ell+1} \end{bmatrix} \begin{bmatrix} W_1^{[2]} \\ \vdots \\ W_\ell^{[2]} \\ R_0 \\ \vdots \\ R_{\ell+1} \end{bmatrix}, \tag{6.22}$$

where R_i are the coefficents of the polynomial $P_{\alpha_n}(\ell+1)$ in (6.21). Note that (6.22) (and the corresponding general set of equations in (6.17)) is solvable given that $N = 2\ell + 2$, which determines the subpacketization as $\ell = \frac{N-2}{2}$. The same procedure described above is carried out for all sparse subpackets in each of the

B segments. The resulting reading cost (including both data and permuted index downloads) is given by,

$$C_R = \frac{Pr'(\log_q \frac{P}{B} + \log_q B) + Pr'N}{L} = \frac{Pr'(\log_q P + N)}{P\frac{N-2}{2}} = \frac{2r'(1 + \frac{\log_q P}{N})}{1 - \frac{2}{N}}. \quad (6.23)$$

Writing Phase: In the writing phase, each user selects the Pr subpackets with the most significant updates and sends the corresponding noise added combined updates (single bit per subpacket) along with their permuted subpacket indices to each of the databases. The noise added combined update of real subpacket i of segment j (assuming this subpacket is among the Pr selected subpackets) sent to database n, $n \in \{1, \ldots, N\}$) is given by,

$$U_n^{[i,j]} = \sum_{k=1}^{\ell} \prod_{r=1, r\neq k}^{\ell} (f_r - \alpha_n)\tilde{\Delta}_k^{[i,j]} + \prod_{r=1}^{\ell}(f_r - \alpha_n)Z^{[i,j]}, \quad (6.24)$$

where $\tilde{\Delta}_k^{[i,j]} = \frac{\Delta_k^{[i,j]}}{\prod_{r=1,r\neq k}^{\ell}(f_r - f_k)}$ with $\Delta_k^{[i,j]}$ being the update of the kth bit of the sparse subpacket i of segment j and $Z^{[i,j]}$ is a random noise bit. To determine the permuted subpacket index of subpacket i of segment j, consider the permutation assigned for segment j (i.e., $\tilde{P}_j$) to be a one-to-one mapping from the set $\{1, \ldots, \frac{P}{B}\}$ to the set $\tilde{P}_j$ in the exact order. Then, the permuted subpacket index corresponding to subpacket i of segment j is given by,

$$Y^{[i,j]} = \tilde{P}_j^{-1}(i), \quad j \in \{1, \ldots, B\}. \quad (6.25)$$

Once the combined updates and permuted subpacket indices corresponding to all Pr chosen subpackets are computed, the user uploads the Pr (update, subpacket, segment) tuples to all databases.[8]

For example, assume that a given user wants to update the real subpackets 2 and 4 from segment 1, subpacket 2 from segment 2 and subpacket 5 from segment 3. Based on the permutations considered in this example, i.e., $\tilde{P}_1 = \{2, 1, 4, 5, 3\}$, $\tilde{P}_2 = \{3, 5, 2, 4, 1\}$ and $\tilde{P}_3 = \{5, 2, 3, 1, 4\}$, the user generates the combined updates $U_n^{[2,1]}$, $U_n^{[4,1]}$, $U_n^{[2,2]}$ and $U_n^{[5,3]}$ which are of the form (6.24). The corresponding permuted subpacket indices are given by,

$$Y^{[2,1]} = \tilde{P}_1^{-1}(2) = 1 \tag{6.26}$$

$$Y^{[4,1]} = \tilde{P}_1^{-1}(4) = 3 \tag{6.27}$$

$$Y^{[2,2]} = \tilde{P}_2^{-1}(2) = 3 \tag{6.28}$$

$$Y^{[5,3]} = \tilde{P}_3^{-1}(5) = 1. \tag{6.29}$$

Therefore, the two permuted (update, subpacket, segment) tuples corresponding to segment 1, sent by the user to database n are given by, $(U_n^{[2,1]}, 1, 1)$ and $(U_n^{[4,1]}, 3, 1)$. Similarly, the permuted (update, subpacket, segment) tuples corresponding to segments 2 and 3 are given by $(U_n^{[2,2]}, 3, 2)$ and $(U_n^{[5,3]}, 1, 3)$, respectively.[9] Once database n, $n \in \{1, \ldots, N\}$ receives the Pr (update, subpacket, segment) tuples, it creates

[8]Note that the 'subpacket' and 'segment' elements in the (update, subpacket, segment) refer to the permuted subpacket index and the real segment index, respectively.

[9]Note that there is no permutation in the segment index, and only the subpacket indices within each segment is being permuted.

the permuted update vectors $\hat{Y}_n^{[j]}$ for each segment $j \in \{1, \ldots, B\}$ given by,

$$\hat{Y}_n^{[j]} = \sum_{i=1}^{\frac{P}{B}} U_n^{[i,j]} e_{\frac{P}{B}}(Y^{[i,j]}), \tag{6.30}$$

where $e_{\frac{P}{B}}(Y^{[i,j]})$ is the all zeros vector of size $\frac{P}{B} \times 1$ with a 1 at the $Y^{[i,j]}$th position. Note that we consider $U_n^{[i,j]} = 0$ for those values of i, $i \in \{1, \ldots, \frac{P}{B}\}$ whose corresponding subpackets are not included in the set of Pr selected subpackets. In order to reverse the permutation privately, each database creates,

$$\hat{U}_n^{[j]} = \hat{Y}_n^{[j]} \otimes 1_\ell = [\hat{Y}_n^{[j]}(1) \cdot 1_\ell^T, \ldots, \hat{Y}_n^{[j]}(\frac{P}{B}) \cdot 1_\ell^T]^T, \tag{6.31}$$

for each segment j, where 1_ℓ is the all ones vector of size $\ell \times 1$. For the example considered, the $\hat{U}_n^{[j]}$ vectors for the three segments, generated by database n, $n \in \{1, \ldots, N\}$ based on the received information $(U_n^{[2,1]}, 1, 1)$, $(U_n^{[4,1]}, 3, 1)$,$(U_n^{[2,2]}, 3, 2)$ and $(U_n^{[5,3]}, 1, 3)$ are given by,

$$\hat{U}_n^{[1]} = \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[4,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix}, \quad \hat{U}_n^{[2]} = \begin{bmatrix} 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[2,2]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix}, \quad \hat{U}_n^{[3]} = \begin{bmatrix} U_n^{[5,3]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix}. \tag{6.32}$$

Next, the databases privately rearrange the updates in the real order and calculate the incremental updates of each segment as $\bar{U}_n^{[j]} = R_n^{[j]} \hat{U}_n^{[j]}$ for $j \in \{1, \ldots, B\}$, and

add it to the jth segment of the existing storage to obtain the updated storage. Consider the incremental update calculation of segment 1 in database n, $n \in \{1, \ldots, N\}$ for the example considered,

$$\bar{U}_n^{[1]} = R_n^{[1]} \hat{U}_n^{[1]} \tag{6.33}$$

$$= \left(\begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \bar{Z}_1 \right) \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[4,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \tag{6.34}$$

$$= \begin{bmatrix} 0_\ell \\ \frac{U_n^{[2,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{U_n^{[2,1]}}{f_\ell - \alpha_n} \\ 0_\ell \\ \frac{U_n^{[4,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{U_n^{[4,1]}}{f_\ell - \alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0_\ell \\ \frac{\Delta_1^{[2,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[2,1]}}{f_\ell - \alpha_n} \\ 0_\ell \\ \frac{\Delta_1^{[4,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[4,1]}}{f_\ell - \alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell), \tag{6.35}$$

where $P_{\alpha_n}(\ell)$ here is a vector of size $\frac{P\ell}{B}$ consisting of polynomial in α_n of degree ℓ, and the last equality is obtained by applying Lemma 3.1. The same process is carried out for the other two segments as well. Since the incremental update is in

the same form as the storage in (6.6), the storage of segment j, $j \in \{1, 2, 3\}$ at time t can be updated as,

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \quad n \in \{1, \ldots, N\}. \tag{6.36}$$

Note from (6.35) that for segment 1, the two real sparse subpackets 2 and 4 have been correctly updated, while ensuring that the rest of the subpackets remain the same, without revealing the real subpacket indices 2 and 4 to any of the databases. The resulting writing cost is given by,

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} = \frac{PrN(1 + \log_q P)}{P^{\frac{N-2}{2}}} = \frac{2r(1 + \log_q P)}{1 - \frac{2}{N}}. \tag{6.37}$$

The total storage complexity (including both data and the permutation reversing matrices) is given by $O(L) + O(\frac{L^2}{B^2} \times B) = O(\frac{L^2}{B})$. The information leakage is derived in Section 6.4.2.

Case 2: MDS coded storage and smaller permutation reversing matrices are used in this case to reduce the storage cost, at the expense of a larger communication cost. The information leakage is the same for both cases 1 and 2, since they both use only within-segment permutations. The same example considered for case 1 (shown in Fig. 6.4) is considered in this case as well.

Initialization: A single subpacket s in case 2 is stored in database n, $n \in$

$\{1, \ldots, N\}$ as,

$$S_n^{[s]} = \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[s]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[s]}. \tag{6.38}$$

Therefore, the storage of segment j, $j \in \{1, \ldots, B\}$ is given by,

$$S_n^{[j]} = \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,j]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[1,j]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[\frac{P}{B},j]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[\frac{P}{B},j]} \end{bmatrix}, \tag{6.39}$$

where $W_i^{[s,j]}$ is the ith parameter of subpacket s in segment j and $Z_i^{[s,j]}$ are random noise symbols. Note that $\frac{P}{B} = 5$ for the example considered. Similar to case 1, the coordinator initializes all noise terms in storage, assigns B permutations $\tilde{P}_i$, $i \in \{1, \ldots, B\}$ of the subpackets in each of the B segments and sends them to the users, and sends the corresponding B noise added permutation reversing matrices $R_n^{[i]}$, $i \in \{1, \ldots, B\}$ to database n, $n \in \{1, \ldots .N\}$, as shown in Fig. 6.4. The noise added permutation reversing matrices are of the form,

$$R_n^{[i]} = \bar{R}^{[i]} + \alpha_n^{\ell} \bar{Z}^{[i]}, \quad i \in \{1, \ldots, B\}, \tag{6.40}$$

where $\bar{R}^{[i]}$ is the permutation reversing matrix corresponding to the ith permutation $\tilde{P}_i$, and $\bar{Z}^{[i]}$ is a random noise matrix, both of size $\frac{P}{B} \times \frac{P}{B}$. The noise added permutation reversing matrices corresponding to the three segments, sent to database n, $n \in \{1, \ldots, N\}$ for the example in Fig. 6.4 are given by (recall: $\tilde{P}_1 = (2, 1, 4, 5, 3)$,

$\tilde{P}_2 = (3,5,2,4,1)$, $\tilde{P}_3 = (5,2,3,1,4))$,

$$R_n^{[1]} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]} \tag{6.41}$$

$$R_n^{[2]} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[2]} \tag{6.42}$$

$$R_n^{[3]} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[3]}. \tag{6.43}$$

As explained in case 1, the coordinator leaves the system once the FL process begins, and all subsequent communications take place only between the users and databases using permuted subpacket indices.

Reading Phase: As described in case 1, let $\tilde{V}_j$ be the set of permuted indices of the sparse subpackets chosen from segment j to be sent to the users for $j \in$

$\{1, \ldots, B\}$. For example, let $\tilde{V}_1 = \{1, 3\}$ be the permuted set of sparse subpackets of segment 1 that needs to be sent to the users at time t. One designated database sends the permuted subpacket indices of each segment (segment 1: $\tilde{V}_1 = \{1, 3\}$) to the users, from which the users identify the corresponding real sparse subpacket indices using the known permutations using $V_j(i) = \tilde{P}_j(\tilde{V}_j(i))$, where V_j is the vector containing the real indices of the sparse subpackets in segment j. In particular, the users perform the same calculation in (6.12) for segment 1 as well as for the other two segments, based on the received sets $\tilde{V}_2$ and $\tilde{V}_3$. Similar to case 1, each database generates a query to send each of the chosen subpackets. The query corresponding to the ith permuted sparse subpacket of segment j is given by,

$$Q_n^{[\tilde{V}_j(i)]} = R_n^{[j]}(:, \tilde{V}_j(i)), \quad j \in \{1, \ldots, B\}. \tag{6.44}$$

Following are the two queries generated by database n, $n \in \{1, \ldots, N\}$ to send the two sparse subpackets (with permuted indices $\tilde{V}_1 = \{1, 3\}$) of segment 1 to the users,

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = R_n^{[1]}(:, 1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_1 \tag{6.45}$$

$$Q_n^{[\tilde{V}_1(2)]} = Q_n^{[3]} = R_n^{[1]}(:,3) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_3, \tag{6.46}$$

where $\hat{Z}_1$ and $\hat{Z}_3$ are the first and third columns of $\bar{Z}^{[1]}$ in (6.41). Then, database n, $n \in \{1, \ldots, N\}$ sends the answers corresponding to each sparse subpacket of each segment to the users as,

$$A_n^{[\tilde{V}_j(i)]} = (S_n^{[j]})^T Q_n^{[\tilde{V}_j(i)]}, \quad j \in \{1, \ldots, B\} \tag{6.47}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[V_j(i),j]} + P_{\alpha_n}(2\ell), \tag{6.48}$$

where $P_{\alpha_n}(2\ell)$ is a polynomial in α_n of degree 2ℓ. For example, the answer corresponding to the first sparse subpacket of segment 1 ($\tilde{V}_1(1) = 1$), sent by database n, $n \in \{1, \ldots, N\}$ to the users is given by,

$$A_n^{[\tilde{V}_1(1)]} = (S_n^{[1]})^T Q_n^{[\tilde{V}_1(1)]} \tag{6.49}$$

$$= \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,1]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[1,1]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[5,1]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[5,1]} \end{bmatrix}^T \left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_1 \right) \tag{6.50}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[2,1]} + P_{\alpha_n}(2\ell). \tag{6.51}$$

The users obtain the parameters of the real subpacket 2 of segment 1, (i.e., $V_1(1) = \tilde{P}_1(\tilde{V}_1(1)) = 2$) by solving,

$$\begin{bmatrix} A_1^{\tilde{V}_1(1)} \\ \vdots \\ A_N^{\tilde{V}_1(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_1^\ell} & \dots & \frac{1}{\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{2\ell} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\alpha_N^\ell} & \dots & \frac{1}{\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{2\ell} \end{bmatrix} \begin{bmatrix} W_\ell^{[2,1]} \\ \vdots \\ W_1^{[2,1]} \\ R_0 \\ \vdots \\ R_{2\ell} \end{bmatrix}, \tag{6.52}$$

where R_i are the coefficients of the polynomial in (6.51). Note that (6.52) (and also the general answers in (6.48)) is solvable given that $N = 3\ell + 1$, which determines the subpacketization as $\ell = \frac{N-1}{3}$. The same procedure described above is carried out for all sparse subpackets in each of the B segments. The resulting reading cost is given by,

$$C_R = \frac{Pr' \log_q P + Pr'N}{L} = \frac{Pr'(\log_q P + N)}{P\frac{N-1}{3}} = \frac{3r'(1 + \frac{\log_q P}{N})}{1 - \frac{1}{N}}. \tag{6.53}$$

Writing Phase: In the writing phase, the user generates Pr combined updates corresponding to the Pr subpackets with the most significant updates. The combined update of the ith subpacket of segment j is defined as (assuming this

subpacket is among the Pr selected subpackets),

$$U_n^{[i,j]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} \Delta_k^{[i,j]} + Z^{[i,j]}, \tag{6.54}$$

where $\Delta_k^{[i,j]}$ is the update of the kth bit of the ith subpacket of segment j and $Z^{[i,j]}$ is a random noise symbol. Similar to case 1, the user generates the permuted subpacket indices corresponding to the real subpacket indices i of each segment j, of the selected Pr subpackets, using (6.25). Once the permuted subpacket indices are generated, the user sends the permuted (update, subpacket, segment) tuples of the Pr selected subpackets to all databases similar to case 1. For the same example considered in case 1 where the user wants to update the real subpackets 2 and 4 from segment 1, subpacket 2 from segment 2 and subpacket 5 from segment 3, the user sends the same permuted (update, subpacket, segment) tuples sent in case 1 given by, $(U_n^{[2,1]}, 1, 1), (U_n^{[4,1]}, 3, 1)$, $(U_n^{[2,2]}, 3, 2)$, $(U_n^{[5,3]}, 1, 3)$ to database n, $n \in \{1, \ldots, N\}$ assuming the same three permutations given by $\tilde{P}_1 = \{2, 1, 4, 5, 3\}$, $\tilde{P}_2 = \{3, 5, 2, 4, 1\}$ and $\tilde{P}_3 = \{5, 2, 3, 1, 4\}$, for the three segments. Once the databases receive the Pr (update, subpacket, segment) tuples, they create the permuted update vectors $\hat{Y}_n^{[j]}$ for each segment $j \in \{1, \ldots, B\}$ using (6.30). For the example considered, the three

permuted update vectors created at database n are given by,

$$\hat{Y}_n^{[1]} = \begin{bmatrix} U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \\ 0 \end{bmatrix}, \quad \hat{Y}_n^{[2]} = \begin{bmatrix} 0 \\ 0 \\ U_n^{[2,2]} \\ 0 \\ 0 \end{bmatrix}, \quad \hat{Y}_n^{[3]} = \begin{bmatrix} U_n^{[5,3]} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{6.55}$$

based on the received information $(U_n^{[2,1]}, 1, 1), (U_n^{[4,1]}, 3, 1)$, $(U_n^{[2,2]}, 3, 2)$, $(U_n^{[5,3]}, 1, 3)$. Using the permuted update vectors $\hat{Y}_n^{[j]}$, $j \in \{1, \ldots, B\}$, database n, $n \in \{1, \ldots, N\}$ privately calculates the correctly rearranged incremental update vector of each segment as $\bar{U}_n^{[j]} = R_n^{[j]} \hat{Y}_n^{[j]}$, $j \in \{1, \ldots, B\}$. The resulting incremental update is of the same form as the storage in (6.39), and therefore can be added to the existing storage to obtain the updated storage of each segment. To explain the above process in terms of an example, consider the incremental update of segment 1 in the same example considered so far,

$$\bar{U}_n^{[1]} = R_n^{[1]} \hat{Y}_n^{[1]} = \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]} \right) \begin{bmatrix} U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \\ 0 \end{bmatrix} \tag{6.56}$$

$$= \begin{bmatrix} 0 \\ U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0 \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,1]} \\ 0 \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[4,1]} \\ 0 \end{bmatrix} + P_{\alpha_n}(\ell), \tag{6.57}$$

where $P_{\alpha_n}(\ell)$ here is a vector of size 5×1, consisting of polynomials in α_n of degree ℓ. Since the incremental update of each segment (i.e., (6.57)) is in the same form as the storage in (6.39), the incremental update is directly added to the existing storage to obtain it's updated version, i.e.,

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \quad j \in \{1, \ldots, B\},\ n \in \{1, \ldots, N\}. \tag{6.58}$$

The writing cost for case 2 is given by,

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} = \frac{PrN(1 + \log_q P)}{P^{\frac{N-1}{3}}} = \frac{3r(1 + \log_q P)}{1 - \frac{1}{N}}. \tag{6.59}$$

The total storage complexity (including both data and the permutation reversing matrices) is given by $O(P) + O(\frac{P^2}{B^2} \times B) = O(\frac{P^2}{B}) = O(\frac{L^2}{BN^2})$. The information leakage is derived in Section 6.4.2.

Case 3: In this case, we use uncoded storage with large permutation reversing matrices. Note that in both cases 1 and 2, only the subpacket indices within each segment were permuted, and the real segment indices were uploaded to the

databases by the users. In this case, we permute subpacket indices within segments as well as the segment indices to reduce the information leakage further. However, this increases the storage cost since the permutation of segment indices requires an additional noise added permutation reversing matrix to be stored at the databases. The communication cost is not significantly affected by the additional round of permutation, compared to case 1 (lowest communication cost thus far).

For cases 3 and 4, we present the general scheme along with the example setting with $P = 12$ subpackets (with subpacketization ℓ) which are divided into and $B = 3$ equal segments, as shown in Fig. 6.5.

Initialization: The storage of a single subpacket (subpacket s) in case 3 is given by,

$$S_n^{[s]} = \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[s]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[s]} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[s]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[s]} \end{bmatrix}, \tag{6.60}$$

with the same notation as in case 1. The subpackets are stacked one after the other (subpacket 1 through subpacket $\frac{P}{B}$ in each segment) in the order of segment 1 through segment B. At the initialization stage, the coordinator sends B randomly and independently chosen permutations of the $\frac{P}{B}$ subpackets in each of the B segments , $\tilde{P}_1, \ldots, \tilde{P}_B$, as well as a randomly and independently chosen permutation of the B segments $\hat{P}$ to the users. The coordinator also places the corresponding noise added permutation reversing matrices (corresponding to B within-segments permu-

tations: $R_n^{[1]}$,...,$R_n^{[B]}$ and one inter-segment permutation: $\hat{R}_n$) at each database n, $n \in \{1, \ldots, N\}$. The noise added permutation reversing matrix corresponding to the ith segment, $i \in \{1, \ldots, B\}$ stored in database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n^{[i]} = (\tilde{R}^{[i]} \otimes \Gamma_n) + \tilde{Z}^{[i]}, \tag{6.61}$$

with the same notation as in case 1. The noise added permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$ stored in database n, $n \in \{1, \ldots, N\}$ is given by,

$$\hat{R}_n = (\bar{R} \otimes I_\ell) + (I_B \otimes \Gamma_n^{-1})\hat{Z} = \begin{bmatrix} b_{1,1}^{[n]} & \ldots & b_{1,B}^{[n]} \\ \vdots & \ddots & \vdots \\ b_{B,1}^{[n]} & \ldots & b_{B,B}^{[n]} \end{bmatrix}, \tag{6.62}$$

where $\bar{R}$ is the permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$, I_k is the identity matrix of size $k \times k$, Γ_n^{-1} is the diagonal matrix given by,

$$\Gamma_n^{-1} = \begin{bmatrix} f_1 - \alpha_n & & \\ & \ddots & \\ & & f_\ell - \alpha_n \end{bmatrix} \tag{6.63}$$

and $\hat{Z}$ is a random noise matrix of size $B\ell \times B\ell$. Each matrix $\hat{R}_n$ is represented in blocks of size $\ell \times \ell$, as shown in the last equality in (6.62). According to the

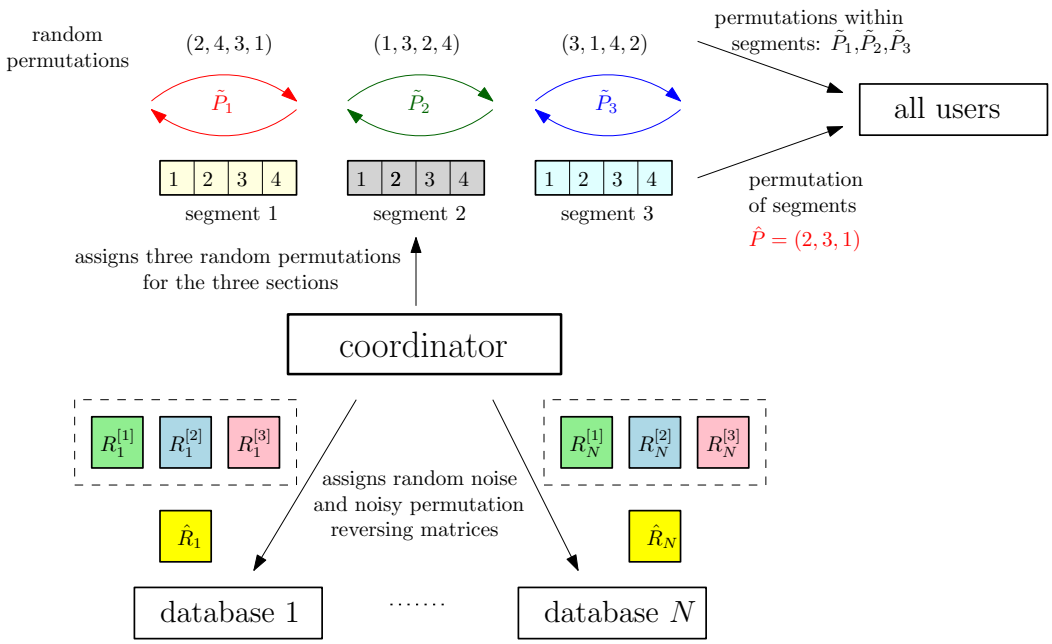


Figure 6.5: Initialization of the scheme for cases 3 and 4.

given permutations in Figure 6.5, the noise added permutation reversing matrix corresponding to the first within-segment permutation ($\tilde{P}_1 = (2, 4, 3, 1)$), i.e., $R_n^{[1]}$ is given by,

$$R_n^{[1]} = \left(\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \otimes \Gamma_n \right) + \tilde{Z}^{[1]} = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]}. \quad (6.64)$$

Similarly, $R_n^{[2]}$ and $R_n^{[3]}$ are given by,

$$R_n^{[2]} = \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} + \tilde{Z}^{[2]}, \quad R_n^{[3]} = \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[3]}, \tag{6.65}$$

For the same example, the inter-segment noise added permutation reversing matrix for database n, $n \in \{1, \ldots, N\}$ is given by,

$$\hat{R}_n = \left(\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \Phi \right) + \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \Gamma_n^{-1} \right) \hat{Z} \tag{6.66}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi \\ \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} \end{bmatrix} \hat{Z} = \begin{bmatrix} b_{1,1}^{[n]} & b_{1,2}^{[n]} & b_{1,3}^{[n]} \\ b_{2,1}^{[n]} & b_{2,2}^{[n]} & b_{2,3}^{[n]} \\ b_{3,1}^{[n]} & b_{3,2}^{[n]} & b_{3,3}^{[n]} \end{bmatrix}, \tag{6.67}$$

where $\Phi = I_\ell$. Each matrix $\hat{R}_n$ is represented in blocks of size $\ell \times \ell$, as shown in the last equality in (6.67), which is useful in the subsequent calculations. The coordinator leaves the system once the storage and permutations are initialized, and the noise added permutation reversing matrices are placed at the databases, before the FL process begins. All communications in the FL process are carried out between the users and databases using permuted subpacket and segment indices.

Reading Phase: The databases determine the set of Pr' subpackets to be sent to the users at time t, based on the permuted information received from the users in the writing phase at time $t-1$. For example, the databases consider the permuted indices of the subpackets updated by all users at time $t-1$, and select the most popular Pr' of them (in terms of permuted indices) to be sent to the users in the reading phase of time t. In case 3, the databases are unaware of the real indices of subpackets within each segment, as well as the corresponding real inidces of the segments, updated by the users. However, the databases can communicate the sparse subpacket and segment indices with the users in their permuted versions (same as what was received in the writing phase at time $t-1$). The users are able to convert the permuted indices into their real versions as all permutations are known by the users. Let the permuted (subpacket, segment) information of each of the Pr' subpackets be indicated by (η_p, ϕ_p). This information is sent to all users at time t by one designated database. The users can convert each permuted (η_p, ϕ_p) into its real versions (η_r, ϕ_r) using,

$$\phi_r = \hat{P}(\phi_p) \tag{6.68}$$

$$\eta_r = \tilde{P}_{\phi_r}(\eta_p). \tag{6.69}$$

For the example in Fig. 6.5, assume that the following permuted (subpacket, segment) pairs are received by a given user from the designated database,

$$(\eta_p, \phi_p) = \{(1,3), (1,1), (1,2)\}. \tag{6.70}$$

Recall that the permutations considered in this example are given by $\tilde{P}_1 = (2, 4, 3, 1)$, $\tilde{P}_2 = (1, 3, 2, 4)$, $\tilde{P}_3 = (3, 1, 4, 2)$ and $\hat{P} = (2, 3, 1)$. In order to see how the real subpacket and segment indices are obtained, consider the first permuted pair $(1, 3)$. Since the permuted segment index is $\phi_p = 3$, the corresponding real segment index is $\phi_r = \hat{P}(3) = 1$. Then, the user can decode the subpacket index within the first segment as, $\eta_r = \tilde{P}_1(1) = 2$. Therefore, the real (subpacket, segment) pair corresponding to the permuted (subpacket, segment) pair $(\eta_p, \phi_p) = (1, 3)$ is given by $(\eta_r, \phi_r) = (2, 1)$. Similarly, the real set of (subpacket, segment) pairs corresponding to the three permuted pairs are given by,

$$(\eta_r, \phi_r) = \{(2, 1), (1, 2), (3, 3)\}. \tag{6.71}$$

Once the real indices of the sparse subpackets and segments are obtained, the user downloads the corresponding subpackets, one by one. To perform the calculations in the reading phase, each database first generates a combined noise added permutation reversing matrix, that combines the within-segment and inter-segment noise added permutation reversing matrices into a single noise added permutation reversing matrix, to facilitate the subsequent calculations. The combined noise added permutation reversing matrix of database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times \begin{bmatrix} I_{\frac{P}{B}} \otimes b_{1,1}^{[n]} & \ldots & I_{\frac{P}{B}} \otimes b_{1,B}^{[n]} \\ \vdots & \vdots & \vdots \\ I_{\frac{P}{B}} \otimes b_{B,1}^{[n]} & \ldots & I_{\frac{P}{B}} \otimes b_{B,B}^{[n]} \end{bmatrix} \tag{6.72}$$

$$
= \begin{bmatrix} R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} & \dots & R_n^{[1]} \begin{bmatrix} b_{1,B}^{[n]} & & \\ & \ddots & \\ & & b_{1,B}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} \\ & \ddots & \\ R_n^{[B]} \begin{bmatrix} b_{B,1}^{[n]} & & \\ & \ddots & \\ & & b_{B,1}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} & \dots & R_n^{[B]} \begin{bmatrix} b_{B,B}^{[n]} & & \\ & \ddots & \\ & & b_{B,B}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} \end{bmatrix} \tag{6.73}
$$

$$
= \dot{R}_n + P_{\alpha_n}(1), \tag{6.74}
$$

where $\dot{R}_n$ is the combined permutation reversing matrix obtained by replacing the 1 in the ith row of $\bar{R}$ in (6.62) by $R^{[i]} \otimes \Gamma_n$ in (6.61), and the zeros by all zeros matrices of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$. $P_{\alpha_n}(1)$ is a matrix of size $L \times L$ consisting of elements that are degree 1 polynomials in α_n.

As an example, consider the generation of the combined noise added permutation reversing matrix of database n, $n \in \{1, \dots, N\}$, for the example setting in Fig. 6.5,

$$
R_n = \begin{bmatrix} R_n^{[1]} & 0 & 0 \\ 0 & R_n^{[2]} & 0 \\ 0 & 0 & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} I_4 \otimes b_{1,1}^{[n]} & \dots & I_4 \otimes b_{1,3}^{[n]} \\ \vdots & \vdots & \vdots \\ I_4 \otimes b_{3,1}^{[n]} & \dots & I_4 \otimes b_{3,3}^{[n]} \end{bmatrix} \tag{6.75}
$$

$$
= \begin{bmatrix}
R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} & R_n^{[1]} \begin{bmatrix} b_{1,2}^{[n]} & & \\ & \ddots & \\ & & b_{1,2}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} & R_n^{[1]} \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} \\
R_n^{[2]} \begin{bmatrix} b_{2,1}^{[n]} & & \\ & \ddots & \\ & & b_{2,1}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} & R_n^{[2]} \begin{bmatrix} b_{2,2}^{[n]} & & \\ & \ddots & \\ & & b_{2,2}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} & R_n^{[2]} \begin{bmatrix} b_{2,3}^{[n]} & & \\ & \ddots & \\ & & b_{2,3}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} \\
R_n^{[3]} \begin{bmatrix} b_{3,1}^{[n]} & & \\ & \ddots & \\ & & b_{3,1}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} & R_n^{[3]} \begin{bmatrix} b_{3,2}^{[n]} & & \\ & \ddots & \\ & & b_{3,2}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} & R_n^{[3]} \begin{bmatrix} b_{3,3}^{[n]} & & \\ & \ddots & \\ & & b_{3,3}^{[n]} \end{bmatrix}_{4\ell\times 4\ell}
\end{bmatrix}. \tag{6.76}
$$

Note that,

$$
R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} = \left(\begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right)
$$

$$
\times \begin{bmatrix} \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} \end{bmatrix} \tag{6.77}
$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \hat{Z}_{1,1} \\ \hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \hat{Z}_{1,1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + P_{\alpha_n}(1) = P_{\alpha_n}(1) \tag{6.78}$$

where $\hat{Z}_{1,1}$ is the submatrix of $\hat{Z}$ in (6.67) consisting of the first ℓ rows and first ℓ columns, $P_{\alpha_n}(1)$ here are matrices of size $4\ell \times 4\ell$, consisting of polynomials in α_n of degree 1 and $0_{\ell\times\ell}$ is the all zeros matrix of size $\ell \times \ell$. Next, consider the calculation of,

$$\begin{aligned} & R_n^{[1]} \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} \\ &= \left(\begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right) \\ &\times \left(\begin{bmatrix} \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1}\hat{Z}_{1,3} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,3} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,3} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,3} \end{bmatrix} \right) \end{aligned} \tag{6.79}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + P_{\alpha_n}(1), \tag{6.80}$$

where $\hat{Z}_{1,3}$ is the submatrix of $\hat{Z}$ in (6.67) consisting of the first ℓ rows and the column indices given by $2\ell+1$ to 3ℓ and $P_{\alpha_n}(1)$ is a matrix of size $4\ell \times 4\ell$ consisting of polynomials of α_n of degree 1. Based on similar calculations, we can write the combined permutation reversing matrix R_n in (6.76) as,

$$R_n = \begin{bmatrix} 0_{4\ell\times4\ell} & 0_{4\ell\times4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} \\ \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} & 0_{4\ell\times4\ell} & 0_{4\ell\times4\ell} \\ 0_{4\ell\times4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} & 0_{4\ell\times4\ell} \end{bmatrix}$$

$$+ P_{\alpha_n}(1), \tag{6.81}$$

where $P_{\alpha_n}(1)$ here is a matrix of size $12\ell \times 12\ell$, whose elements are polynomials in α_n of degree 1. Note that the first part of the combined noise added permutation reversing matrix R_n is simply the 1 in the ith row of $\bar{R}$ (permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$) replaced by $\tilde{R}^{[i]} \otimes \Gamma_n$, where $\tilde{R}^{[i]}$ is the permutation reversing matrix corresponding to the within-segment permutation $\tilde{P}_i$, for each $i \in \{1, 2, 3\}$.

In order to download the subpacket corresponding to the permuted pair (η_p, ϕ_p), each database generates the query given by,

$$Q_n^{[\eta_p,\phi_p]} = (I_P \otimes \Gamma_n^{-1}) \times \sum_{k=1}^{\ell} R_n(:, (\phi_p - 1)\frac{P}{B}\ell + (\eta_p - 1)\ell + k), \tag{6.82}$$

where I_P is the identity matrix of size $P \times P$. Then, database n, $n \in \{1, \ldots, N\}$ sends the answer corrresponding to the permuted subpacket (η_p, ϕ_p) as,

$$A_n^{[\eta_p,\phi_p]} = S_n^T Q_n^{[\eta_p,\phi_p]} = \frac{1}{f_1 - \alpha_n} W_1^{[\eta_r,\phi_r]} + \cdots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[\eta_r,\phi_r]} + P_{\alpha_n}(\ell + 3), \tag{6.83}$$

where $P_{\alpha_n}(\ell + 3)$ is a polynomial in α_n of degree $\ell + 3$. Then, the user can obtain the values of the corresponding real subpacket indicated by (η_r, ϕ_r), i.e.,

$W_1^{[\eta_r,\phi_r]},\ldots,W_\ell^{[\eta_r,\phi_r]}$ using all answers received by the N databases as,

$$\begin{bmatrix} A_1^{[\eta_p,\phi_p]} \\ \vdots \\ A_N^{[\eta_p,\phi_p]} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \ldots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^{\ell+3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \ldots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \ldots & \alpha_N^{\ell+3} \end{bmatrix} \begin{bmatrix} W_1^{[\eta_r,\phi_r]} \\ \vdots \\ W_\ell^{[\eta_r,\phi_r]} \\ R_{0:\ell+3} \end{bmatrix}, \tag{6.84}$$

where each R_i corresponds to the ith coefficient of the polynomial $P_{\alpha_n}(\ell+3)$ in (6.83). The equation in (6.84) is solvable if $N = 2\ell+4$, which determines the subpacketization as $\ell = \frac{N-4}{2}$.

As an example, consider the download of the permuted subpacket indicated by $(\eta_p,\phi_p) = (1,3)$, from the same example setting in Fig. 6.5. Database n, $n \in \{1,\ldots,N\}$ creates the query given by,

$$Q_n^{[1,3]} = (I_{12} \otimes \Gamma_n^{-1}) \times \sum_{k=1}^{\ell} R_n(:, 8\ell+k) \tag{6.85}$$

$$= \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}_{12\ell\times 12\ell} \times \left(\begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_{10\ell} \end{bmatrix} + \dot{P}_{\alpha_n}(1) \right) \tag{6.86}$$

$$= \begin{bmatrix} 0_\ell \\ 1_\ell \\ 0_{10\ell} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1-\alpha_n)P_{\alpha_n}(1) \\ \vdots \\ (f_\ell-\alpha_n)P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_1-\alpha_n)P_{\alpha_n}(1) \\ \vdots \\ (f_\ell-\alpha_n)P_{\alpha_n}(1) \end{bmatrix} \end{bmatrix}_{L\times 1}, \tag{6.87}$$

where $\dot{P}_{\alpha_n}(1)$ is a vector of size $12\ell \times 1$, consisting of polynomials in α_n of degree 1, and $P_{\alpha_n}(1)$ are polynomials of α_n of degree 1. Note that 0_k and 1_k refer to all zeros and all ones vectors of size $k \times 1$, respectively. The answer corresponding to the above query, sent to the users by database n, $n \in \{1, \ldots, N\}$ is given by,

$$A_n^{[1,3]} = S_n^T Q_n^{[1,3]} \tag{6.88}$$

$$= \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[1]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[1]} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[1]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[12]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[12]} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[12]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[12]} \end{bmatrix} \end{bmatrix}^T \times \left(\begin{bmatrix} 0_\ell \\ 1_\ell \\ 0_{10\ell} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1-\alpha_n)P_{\alpha_n}(1) \\ \vdots \\ (f_\ell-\alpha_n)P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_1-\alpha_n)P_{\alpha_n}(1) \\ \vdots \\ (f_\ell-\alpha_n)P_{\alpha_n}(1) \end{bmatrix} \end{bmatrix}_{L\times 1} \right) \tag{6.89}$$

$$= \frac{1}{f_1 - \alpha_n} W_1^{[2]} + \cdots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell + 3), \tag{6.90}$$

from which the (real) second subpacket of segment 1, i.e., $(\eta_r, \phi_r) = (2, 1)$ can be obtained using all answers of the N databases if $N = 2\ell + 4$ is satisfied, which determines the subpacketization as $\ell = \frac{N-4}{2}$. The resulting reading cost is given by,

$$C_R = \frac{Pr'(N + \log_q B + \log_q \frac{P}{B})}{L} = \frac{Pr'(N + \log_q P)}{P\frac{N-4}{2}} = \frac{2r'(1 + \frac{\log_q P}{N})}{1 - \frac{4}{N}}. \tag{6.91}$$

Writing Phase: In the writing phase, the user sends the combined updates, permuted subpacket indices and permuted segment indices of the Pr subpackets with the most significant updates to all databases. Let $(\eta_r^{[i]}, \phi_r^{[i]})$, $i \in \{1, \ldots, Pr\}$ be the real (subpacket, segment) pair corresponding to the ith selected subpacket. For each of the Pr selected subpackets, the user generates a combined update bit given by,

$$U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]} = \sum_{k=1}^{\ell} \prod_{j=1, j\neq k}^{\ell} (f_j - \alpha_n) \tilde{\Delta}_k^{[\eta_r^{[i]}, \phi_r^{[i]}]} + \prod_{j=1}^{\ell} (f_j - \alpha_n) Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \quad i \in \{1, \ldots, Pr\}, \tag{6.92}$$

where $\tilde{\Delta}_k^{[\eta_r^{[i]}, \phi_r^{[i]}]} = \frac{\Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]}}{\prod_{j=1, j\neq k}^{\ell} (f_j - f_k)}$, with $\Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ being the update of the kth parameter of subpacket $\eta_r^{[i]}$ of segment $\phi_r^{[i]}$ and $Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ is a random noise symbol. The user sends the permuted (update, subpacket, segment) tuple given by $(U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$ for the ith selected subpacket where $U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ is the combined update of the subpacket

of the form (6.92), $\eta_p^{[i]}$ is the permuted subpacket index obtained by,

$$\eta_p^{[i]} = \tilde{P}^{-1}_{\phi_r^{[i]}}(\eta_r^{[i]}) \tag{6.93}$$

and $\phi_p^{[i]}$ is the permuted segment index obtained by,

$$\phi_p^{[i]} = \hat{P}^{-1}(\phi_r^{[i]}) \tag{6.94}$$

where $\tilde{P}_{\phi_r^{[i]}}$ and $\hat{P}$ are considered to be the one-to-one mappings from j to $\tilde{P}_{\phi_r^{[i]}}(j)$ for $j = 1, \ldots, \frac{P}{B}$ and i to $\hat{P}(i)$, for $i = 1, \ldots, B$, respectively. For the example in Fig. 6.5, assume that a given user wants to update the Pr sparse subpackets identified by the real (subpacket, segment) pairs given by, $(\eta_r, \phi_r) = \{(2,1),(2,2),(3,3)\}$. Based on the within segment permutations given by $\tilde{P}_1 = (2,4,3,1)$, $\tilde{P}_2 = (1,3,2,4)$, $\tilde{P}_3 = (3,1,4,2)$, and the segment-wise permutation given by $\hat{P} = (2,3,1)$, the user sends the following (permuted) information to database n, $n \in \{1, \ldots, N\}$,

$$\begin{aligned}
&(U_n^{[\eta_r,\phi_r]}, \eta_p, \phi_p) \\
&\quad = \{(U_n^{[2,1]}, \tilde{P}_1^{-1}(2), \hat{P}^{-1}(1)), (U_n^{[2,2]}, \tilde{P}_2^{-1}(2), \hat{P}^{-1}(2)), (U_n^{[3,3]}, \tilde{P}_3^{-1}(3), \hat{P}^{-1}(3))\} \qquad (6.95) \\
&\quad = \{(U_n^{[2,1]}, 1, 3), (U_n^{[2,2]}, 3, 1), (U_n^{[3,3]}, 1, 2)\}, \qquad (6.96)
\end{aligned}$$

where the three $U_n^{[\eta_r,\phi_r]}$ terms are generated as in (6.92). As an illustration, the real (subpacket, segment) pair given by $(2,1)$, is converted to the permuted pair $(1,3)$ as

follows. Note that in the segment-wise permutation $\hat{P} = (2, 3, 1)$, the real segment index 1 lies in the third position. Therefore, $\phi_r = 1$ is converted to $\phi_p = 3$. Next, in the permutation corresponding to segment 1 ($\tilde{P}_1 = (2, 4, 3, 1)$), the subpacket index 2 lies in the first position. Therefore, $\eta_r = 2$ is converted to $\eta_p = 1$.

Once the databases receive all permuted (update, subpacket, segment) tuples, they construct the permuted update vector as,

$$\tilde{U}_n = \sum_{i=1}^{Pr} U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]} e_P((\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]}), \tag{6.97}$$

where $e_p((\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]})$ is the all zeros vector of size P with a 1 at the $(\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]}$th position. This vector is then scaled by an all ones vector of size $\ell \times 1$ (i.e., 1_ℓ) to aid the rest of the calculations. The scaled permuted update vector is given by,

$$\hat{U}_n = \tilde{U}_n \otimes 1_\ell = \begin{bmatrix} \tilde{U}_n(1) \cdot 1_\ell \\ \vdots \\ \tilde{U}_n(P) \cdot 1_\ell \end{bmatrix}. \tag{6.98}$$

Then, database n, $n \in \{1, \ldots, N\}$ calculates the incremental update using the combined noise added permutation reversing matrix in (6.74) as,

$$\bar{U}_n = R_n \times \hat{U}_n, \tag{6.99}$$

which is of the same form as the storage in (6.60). Therefore, the storage at time t,

$S_n^{[t]}$ can be updated as,

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n. \tag{6.100}$$

For the example considered, the scaled permuted update vector based on the information received by the databases in (6.96) is given by,

$$\hat{U}_n = (U_n^{[2,1]} e_{12}(9) + U_n^{[2,2]} e_{12}(3) + U_n^{[3,3]} e_{12}(5)) \otimes 1_\ell = \begin{bmatrix} \begin{bmatrix} 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[2,2]} \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[3,3]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \end{bmatrix} \tag{6.101}$$

Then, database n, $n \in \{1, \ldots, N\}$ computes the incremental update using the combined noise added permutation reversing matrix in (6.81) as,

$$\bar{U}_n = R_n \times \hat{U}_n \tag{6.102}$$

$$
\begin{aligned}
&= \begin{bmatrix} 0_{4\ell\times 4\ell} & 0_{4\ell\times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} \\ \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} & 0_{4\ell\times 4\ell} & 0_{4\ell\times 4\ell} \\ 0_{4\ell\times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} & 0_{4\ell\times 4\ell} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ U_n^{[2,2]}\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[3,3]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[2,1]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \end{bmatrix} \\
&\quad + P_{\alpha_n}(\ell+1) \qquad (6.103) \\
&= \left[0_\ell, \frac{U_n^{[2,1]}}{f_1-\alpha_n}, \ldots, \frac{U_n^{[2,1]}}{f_\ell-\alpha_n}, 0_{3\ell}, \frac{U_n^{[2,2]}}{f_1-\alpha_n}, \ldots, \frac{U_n^{[2,2]}}{f_\ell-\alpha_n}, 0_{4\ell}, \frac{U_n^{[3,3]}}{f_1-\alpha_n}, \ldots, \frac{U_n^{[3,3]}}{f_\ell-\alpha_n}, 0_\ell, \right]^T \\
&\quad + P_{\alpha_n}(\ell+1) \qquad (6.104) \\
&= \left[0_\ell, \frac{\Delta_1^{[2,1]}}{f_1-\alpha_n}, \ldots, \frac{\Delta_\ell^{[2,1]}}{f_\ell-\alpha_n}, 0_{3\ell}, \frac{\Delta_1^{[2,2]}}{f_1-\alpha_n}, \ldots, \frac{\Delta_\ell^{[2,2]}}{f_\ell-\alpha_n}, 0_{4\ell}, \frac{\Delta_1^{[3,3]}}{f_1-\alpha_n}, \ldots, \frac{\Delta_\ell^{[3,3]}}{f_\ell-\alpha_n}, 0_\ell, \right]^T \\
&\quad + P_{\alpha_n}(\ell+1), \qquad (6.105)
\end{aligned}
$$

where $P_{\alpha_n}(\ell+1)$ is a vector of size $L \times 1$ whose elements are polynomials in α_n of degree $\ell+1$, and (6.105) follows from Lemma 3.1. Note from (6.105) that the (real) subpacket 2 of segment 1, subpacket 2 of segment 2 and subpacket 3 of segment 3

are correctly updated, without revealing these real indices to any of the databases.[10] Since $\bar{U}_n$ is in the same form as the storage in (6.60), the storage at time t can be updated by adding $\bar{U}_n$ to the existing storage. The resulting writing cost is given by,

$$C_W = \frac{PrN(1+\log_q B + \log_q \frac{P}{B})}{L} = \frac{PrN(1+\log_q P)}{P^{\frac{N-4}{2}}} = \frac{2r(1+\log_q P)}{1-\frac{4}{N}}. \quad (6.106)$$

The storage complexities of data, within-segment noise added permutation reversing matrices and the inter-segment noise added permutation reversing matrix are given by $O(L)$, $O(\frac{L^2}{B})$ and $O(\ell^2 B^2) = O(N^2 B^2)$, respectively. Therefore the storage complexity is $\max\{O(\frac{L^2}{B}), O(N^2B^2)\}$. The information leakage (on the indices of sparse updates) is derived in Section 6.4.2.

Case 4: In this case, we consider coded storage and smaller permutation reversing matrices to reduce the storage cost. Both within-segment and inter-segment permutations are considered in this case to reduce the information leakage.

Initialization: The storage of a single subpacket s in database n, $n \in \{1, \ldots, N\}$ is given by,

$$S_n^{[s]} = \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[s]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[s]}, \quad (6.107)$$

with the same notation used in case 2. Therefore, the storage of a given segment j,

[10]Note that the vector $P_{\alpha_n}(\ell+1)$ in (6.105) hides these non-zero updates from the databases.

$j \in \{1, \ldots, B\}$ is given by,

$$S_{n,j} = \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,j]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,j]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[\frac{P}{B},j]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[\frac{P}{B},j]} \end{bmatrix}, \tag{6.108}$$

where $W_i^{[k,j]}$ is the ith parameter of subpacket k of segment j and $Z_i^{[k,j]}$ are random noise symbols. The segments are stacked one after the other in the order of segment 1 through segment B. As described in case 3, the coordinator sends the within-segment and inter-segment permutations $\tilde{P}_1, \ldots, \tilde{P}_B$ and $\hat{P}$ to the users and the corresponding noise added permutation reversing matrices given by $R_n^{[1]}, \ldots, R_n^{[B]}$ and $\hat{R}_n$ to database n, $n \in \{1, \ldots, N\}$. The noise added permutation reversing matrices for the within segment permutations $\tilde{P}_i$ are of the form (6.40), and the noise added permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$ is of the form,

$$\hat{R}_n = \hat{R} + \alpha_n^{\ell} Z, \tag{6.109}$$

where $\hat{R}$ is the permutation reversing matrix corresponding to $\hat{P}$ and Z is a random noise matrix, both of size $B \times B$. For the example considered in Figure 6.5, the noise added permutation reversing matrices corresponding to $\tilde{P}_1 = \{2, 4, 3, 1\}$, $\tilde{P}_2 = \{1, 3, 2, 4\}$, $\tilde{P}_3 = \{3, 1, 4, 2\}$ and $\hat{P} = \{2, 3, 1\}$, stored in database n, $n \in \{1, \ldots, N\}$

are given by,

$$R_n^{[1]} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]},\ R_n^{[2]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[2]},\ R_n^{[3]} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[3]} \tag{6.110}$$

$$\text{and} \quad \hat{R}_n = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} Z. \tag{6.111}$$

The initialization stage ends and the coordinator leaves the system once the storage, permutations and the noise added permutation reversing matrices are initialized.

To aid the calculations in the reading and writing phases described next, the databases compute a combined noise added permutation reversing matrix as described in case 3. This matrix for database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times (\hat{R}_n \otimes I_{\frac{P}{B}}) \tag{6.112}$$

$$= \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times \begin{bmatrix} \hat{R}_n(1,1) I_{\frac{P}{B}} & \ldots & \hat{R}_n(1,B) I_{\frac{P}{B}} \\ \vdots & \vdots & \vdots \\ \hat{R}_n(B,1) I_{\frac{P}{B}} & \ldots & \hat{R}_n(B,B) I_{\frac{P}{B}} \end{bmatrix} \tag{6.113}$$

where $I_{\frac{P}{B}}$ is the identity matrix of size $\frac{P}{B} \times \frac{P}{B}$. The combined matrix R_n places $R_n^{[i]}$ at the position of the 1 in the ith row of $\hat{R}$ in (6.109), with added noise. The combined noise added permutation reversing matrix for the example considered in Fig. 6.5 for database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n = \begin{bmatrix} R_n^{[1]} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & R_n^{[2]} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} \hat{R}_n(1,1)I_4 & \hat{R}_n(1,2)I_4 & \hat{R}_n(1,3)I_4 \\ \hat{R}_n(2,1)I_4 & \hat{R}_n(2,2)I_4 & \hat{R}_n(2,3)I_4 \\ \hat{R}_n(3,1)I_4 & \hat{R}_n(3,2)I_4 & \hat{R}_n(3,3)I_4 \end{bmatrix} \tag{6.114}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell}\bar{Z}^{[1]} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \alpha_n^{\ell}\bar{Z}^{[2]} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell}\bar{Z}^{[3]} \end{bmatrix}$$

$$
\times \left(\begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{4\times4} \\ \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{4\times4} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{4\times4} & 0_{4\times4} \end{bmatrix} + \alpha_n^\ell \tilde{Z} \right) \tag{6.115}
$$

$$
= \begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} & 0_{4\times4} \end{bmatrix} + \alpha_n^\ell P_{\alpha_n}(\ell), \tag{6.116}
$$

where $\tilde{Z}$ in (6.115) is a random noise matrix of size 12×12, resulted by the noise component of $\hat{R}_n$ and $P_{\alpha_n}(\ell)$ in (6.116) is a matrix of size 12×12 with entries consisting of polynomials in α_n of degree ℓ. Note that the combined noise added permutation reversing matrix in (6.116) places each $R_n^{[i]}$ at the position of the 1 in the ith row of the permutation reversing matrix in (6.111) for $i = 1, 2, 3$, with added noise.

Reading Phase: As explained in case 3, the databases determine the permuted (subpacket, segment) tuples given by (η_p, ϕ_p) for the Pr' sparse subpackets and send them to the users. The users obtain the real (subpacket, segment) information of the (η_p, ϕ_p) tuples from (6.68) and (6.69). For the example considered in Fig. 6.5, the (η_p, ϕ_p) and (η_r, ϕ_r) pairs in (6.70) and (6.71) considered in case 3 are valid for case 4 as well.

In order to send the subpacket corresponding to the permuted (subpacket, segment) pair (η_p, ϕ_p), database n, $n \in \{1, \ldots, N\}$ creates a query given by,

$$Q_n^{[\eta_p, \phi_p]} = R_n(:, (\phi_p - 1)\frac{P}{B} + \eta_p). \tag{6.117}$$

For $(\eta_p, \phi_p) = (1, 3)$, the corresponding query is given by,

$$Q_n^{[1,3]} = R_n(:, 9) = \begin{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ 0_8 \end{bmatrix} + \alpha_n^{\ell} P_{\alpha_n}(\ell), \tag{6.118}$$

where $P_{\alpha_n}(\ell)$ here is a vector of size 12×1 with entries consisting of polynomials in α_n of degree ℓ. The databases send the answer to each query corresponding to (η_p, ϕ_p) as,

$$A_n^{[\eta_p,\phi_p]} = S_n^T Q_n^{[\eta_p,\phi_p]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} W_k^{[\eta_r,\phi_r]} + P_{\alpha_n}(4\ell), \tag{6.119}$$

where $P_{\alpha_n}(4\ell)$ is a polynomial in α_n of degree 4ℓ. The users can obtain the parameters of the corresponding real (subpacket, segment) pair (η_r, ϕ_r) using,

$$\begin{bmatrix} A_1^{[\eta_p,\phi_p]} \\ \vdots \\ A_N^{[\eta_p,\phi_p]} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_1^{\ell}} & \dots & \frac{1}{\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{4\ell} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\alpha_N^{\ell}} & \dots & \frac{1}{\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{4\ell} \end{bmatrix} \begin{bmatrix} W_{\ell}^{[\eta_r,\phi_r]} \\ \vdots \\ W_1^{[\eta_r,\phi_r]} \\ R_{0:4\ell} \end{bmatrix}, \tag{6.120}$$

where R_i are the coefficients of the polynomial in (6.119), if $N = 5\ell + 1$ is satisfied. This determines the subpacketization as $\ell = \frac{N-1}{5}$. For the example considered, the

answer for $(\eta_p, \phi_p) = (1, 3)$ from database n, $n \in \{1, \ldots, N\}$ is given by,

$$A_n^{[1,3]} = S_n^T Q_n^{[1,3]} = \begin{bmatrix} \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,1]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,1]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,1]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,1]} \end{bmatrix} \\ \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,2]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,2]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,2]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,2]} \end{bmatrix} \\ \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,3]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,3]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,3]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,3]} \end{bmatrix} \end{bmatrix}^T \left(\begin{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ 0_8 \end{bmatrix} + \alpha_n^{\ell} P_{\alpha_n}(\ell) \right) \tag{6.121}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[2,1]} + P_{\alpha_n}(4\ell). \tag{6.122}$$

The users can obtain the parameters of the (real) second subpacket of segment 1 (since the real indices corresponding to permuted $(\eta_p, \phi_p) = (1, 3)$ are $(\eta_r, \phi_r) = (2, 1)$ from (6.70) and (6.71).) using the N answers received if $N = 5\ell + 1$ is satisfied. This defines the subpacketization for case 4 as $\ell = \frac{N-1}{5}$. The resulting reading cost is given by,

$$C_R = \frac{Pr'(N + \log_q B + \log_q \frac{P}{B})}{L} = \frac{Pr'(N + \log_q P)}{P^{\frac{N-1}{5}}} = \frac{5r'(1 + \frac{\log_q P}{N})}{1 - \frac{1}{N}}. \tag{6.123}$$

Writing phase: Similar to case 3, the user selects the Pr subpackets with the

most significant updates and let $(\eta_r^{[i]}, \phi_r^{[i]})$, $i \in \{1, \ldots, Pr\}$ be the real (subpacket, segment) information of the ith selected subpacket. For each such subpacket, the user generates a combined update (single symbol) given by,

$$U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} \Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]} + Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \tag{6.124}$$

where $\Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ is the update of the kth parameter of subpacket $\eta_r^{[i]}$ of segment $\phi_r^{[i]}$, and $Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ is a random noise symbol. The user sends the permuted (update, subpacket, segment) tuple given by $(U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$ for the ith sparse subpacket for $i \in \{1, \ldots, Pr\}$ where $U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ is the combined update of the subpacket of the form (6.124), $\eta_p^{[i]}$ is the permuted subpacket index obtained by $\eta_p^{[i]} = \tilde{P}_{\phi_r^{[i]}}^{-1}(\eta_r^{[i]})$ and $\phi_p^{[i]}$ is the permuted segment index obtained by $\phi_p^{[i]} = \hat{P}^{-1}(\phi_r^{[i]})$, with the same notation used in the description of case 3. For the example considered, assume that a user wants to update real (subpacket, segment) pairs given by $(\eta_r, \phi_r) = \{(2,1), (2,2), (3,3)\}$. Based on the within-segment permutations given by $\tilde{P}_1 = (2,4,3,1)$, $\tilde{P}_2 = (1,3,2,4)$, $\tilde{P}_3 = (3,1,4,2)$, and the inter-segment permutation given by $\hat{P} = (2,3,1)$, the user sends the following (permuted) information to database n, $n \in \{1, \ldots, N\}$, as described in case 3,

$$(U_n, \eta_p, \phi_p) = \{(U_n^{[2,1]}, 1, 3), (U_n^{[2,2]}, 3, 1), (U_n^{[3,3]}, 1, 2)\}. \tag{6.125}$$

Based on the permuted sparse update tuples $(U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$, $i \in \{1, \ldots, Pr\}$ received, database n, $n \in \{1, \ldots, N\}$ constructs the permuted update vector given

in (6.97). Then, database n, $n \in \{1, \ldots, N\}$ calculates the permutation-reversed incremental update as $\bar{U}_n = R_n \tilde{U}_n$ which is of the same form as the storage in (6.108). Therefore, the storage at time t can be updated as $S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n$. For the example considered, the permuted update vector from (6.97) is given by,

$$\tilde{U}_n = [0, 0, U_n^{[2,2]}, 0, U_n^{[3,3]}, 0, 0, 0, U_n^{[2,1]}, 0, 0, 0]^T. \tag{6.126}$$

Then, each database calculates the permutation-reversed incremental update as,

$$\bar{U}_n = R_n \tilde{U}_n \tag{6.127}$$

$$= \left(\begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} & 0_{4\times4} \end{bmatrix} + \alpha_n^{\ell} P_{\alpha_n}(\ell) \right) \times \begin{bmatrix} 0 \\ 0 \\ U_n^{[2,2]} \\ 0 \\ U_n^{[3,3]} \\ 0 \\ 0 \\ 0 \\ U_n^{[2,1]} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.128}$$

$$= [0, U_n^{[2,1]}, 0, 0, 0, U_n^{[2,2]}, 0, 0, 0, 0, U_n^{[3,3]}, 0]^T + P_{\alpha_n}(2\ell) \tag{6.129}$$

$$= \left[0, \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,1]}, 0, 0, 0, \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,2]}, 0, 0, 0, 0, \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[3,3]}, 0\right]^T + P_{\alpha_n}(2\ell) \tag{6.130}$$

where $P_{\alpha_n}(2\ell)$ is a vector of size 12×1, consisting of polynomials in α_n of degree 2ℓ. Note that the (real) subpacket 2 of segment 1, subpacket 2 of segment 2 and subpacket 3 of segment 3 ($(\eta_r, \phi_r) = \{(2,1), (2,2), (3,3)\}$) are correctly placed in (6.130), without revealing the real indices to the databases.[11] Since the incremental update in (6.130) is of the same form as (6.108), it is directly added to the existing storage to obtain the updated storage. The writing cost is given by,

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} = \frac{PrN(1 + \log_q P)}{P^{\frac{N-1}{5}}} = \frac{5r(1 + \log_q P)}{1 - \frac{1}{N}}. \tag{6.131}$$

The storage complexities of data, noise added within and inter-segment permutation reversing matrix are given by $O(P) = O(\frac{L}{N})$, $O(\frac{P^2}{B}) = O(\frac{L^2}{N^2 B})$ and $O(B^2)$, respectively. Therefore, the storage complexity is $\max\{O(\frac{L^2}{N^2 B}), O(B^2)\}$. The information leakage on the indices of the sparse updates is derived in Section 6.4.2.

6.4.2 Information Leakage

In this section, we quantify the information leakage of the four cases for a given FL setting with N databases, P subpackets, B segments and uplink and downlink

[11] The sparse updates in the first part of (6.130) are hidden from the databases by the noise vector $P_{\alpha_n}(2\ell)$.

sparsficiation rates give by r, r'. In the proposed scheme, the users send no information to the databases in the reading phase, and the databases determine the sparse set of subpackets and send them to the users. Therefore, there is no information leakage in the reading phase. In the writing phase, the users send Pr permuted tuples of the form (update, subpacket, segment) to databases, from which a given amount of information about the sparse subpacket indices updated by a given user is allowed to leak.

There are two types of information within a given user's uploads that leak information about the user's local data, namely, 1) values of sparse updates, 2) positions of sparse updates. Note that in the proposed scheme, a random noise symbol is added to all combined sparse updates sent to the databases in all four cases. Therefore, from Shannon's one time pad theorem, the combined update values sent by the user to all databases are random noise symbols which are independent of the values of the sparse updates included in it. Therefore, the amount of information leaked by the values of the sparse updates in this work is zero.

From the set of permuted (update, subpacket, segment) tuples sent by a given user at time t, only the (subpacket, segment) pairs may possibly contain information about the real positions of the sparse updates, since the *update* component is simply random noise that is independent of the real positions of the sparse updates. Let $Y^{[t]}$ be the set of Pr subpacket indices corresponding to the set of permuted (subpacket, segment) pairs sent by a given user at time t. Let $X^{[t]}$ be the set of real indices of the Pr sparse subpackets of the model, chosen to be updated by the user at time t. Therefore, the amount of information leaked to the databases about the real

positions of the sparse updates at time t is quantified by the mututal information between $X^{[t]}$ and $Y^{[t]}$, i.e., $I(X^{[t]};Y^{[t]})$. In this section, we quantify this mutual information for all four cases.

Cases 1 and 2: In cases 1 and 2, permutations exist only within each segment and not among segments. To simplify the notation, we drop the time index in the following calculation, and assume that each X and Y correspond to real and permuted quantities at time t. In order to quantify $I(X;Y)$, we first derive the following conditional probability.

$$P(X=x|Y=y) = \frac{\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} P(X=x, Y=y, \tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)}{P(Y=y)} \tag{6.132}$$

$$= \frac{\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} P(Y=y|X=x, \tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)P(X=x)\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i)}{P(Y=y)} \tag{6.133}$$

$$= \frac{P(X=x)\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \mathbf{1}_{\{Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}}\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i)}{P(Y=y)} \tag{6.134}$$

$$= \begin{cases} \frac{P(X=x)\prod_{i=1}^{B}\hat{y}_i!\prod_{i=1}^{B}(\frac{P}{B}-\hat{y}_i)!}{\left(\frac{P}{B}\right)!^B P(Y=y)}, & \text{for } x,y: \hat{x}_i=\hat{y}_i,\ \forall\ i \\ 0, & \text{otherwise} \end{cases} \tag{6.135}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\prod_{i=1}^{B}\frac{\hat{y}_i!(\frac{P}{B}-\hat{y}_i)!}{\left(\frac{P}{B}\right)!}, & \text{for } x,y: \hat{x}_i=\hat{y}_i,\ \forall\ i \\ 0, & \text{otherwise} \end{cases} \tag{6.136}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}, & \text{for } x,y: \hat{x}_i=\hat{y}_i,\ \forall\ i \\ 0, & \text{otherwise} \end{cases} \tag{6.137}$$

where $\hat{x}_i$ and $\hat{y}_i$ are the numbers of real and permuted sparse subpackets in segment i,

respectively, and (6.133) is obtained by the mutual independence of the permutations of the B segments and the real positions of the sparse updates. (6.135) is derived by counting the number of all possible permutations that result in the given $Y = y$ from the given $X = x$. Next, we compute the probability,

$$P(Y = y) = \sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_x P(Y = y, X = x, \tilde{P}_1 = \tilde{p}_1, \ldots, \tilde{P}_B = \tilde{p}_B) \tag{6.138}$$

$$= \sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_x P(Y = y|X = x, \tilde{P}_1 = \tilde{p}_1, \ldots, \tilde{P}_B = \tilde{p}_B) P(X = x) \prod_{i=1}^{B} P(\tilde{P}_i = \tilde{p}_i) \tag{6.139}$$

$$= \sum_x P(X = x) \sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \mathbf{1}_{\{Y=y, X=x, \tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}} \prod_{i=1}^{B} P(\tilde{P}_i = \tilde{p}_i) \tag{6.140}$$

$$= \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X = x) \frac{\prod_{i=1}^{B} \hat{y}_i! \prod_{i=1}^{B} (\frac{P}{B} - \hat{y}_i)!}{\frac{1}{(\frac{P}{B})!^B}} \tag{6.141}$$

$$= \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X = x) \tag{6.142}$$

for each y such that $\sum_{i=1}^{B} \hat{y}_i = Pr$. Therefore, from (6.137),

$$P(X = x|Y = y) = \begin{cases} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)}, & \text{for } x, y : \hat{x}_i = \hat{y}_i,\ \forall i \\ 0, & \text{otherwise} \end{cases} \tag{6.143}$$

. Then,

$$H(X|Y = y) = -\sum_x P(X = x|Y = y) \log P(X = x|Y = y) \tag{6.144}$$

$$= -\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X = x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X = x)} \log \frac{P(X = x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X = x)}, \tag{6.145}$$

from which we obtain,

$$H(X|Y) = \sum_{y} P(Y=y)H(X|Y=y) \tag{6.146}$$

$$= -\sum_{y}\left(\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)\right)$$

$$\times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i}\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)}\log\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)} \tag{6.147}$$

$$= -\sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr}\prod_{i=1}^{B}\binom{P/B}{\hat{y}_i}\left(\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)\right)$$

$$\times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i}\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)}\log\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)} \tag{6.148}$$

$$= -\sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr}\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)}$$

$$\times \log\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)} \tag{6.149}$$

$$= -\sum_{\hat{x}:\sum_{i=1}^{B}\hat{x}_i=Pr}\sum_{x\in\hat{x}}P(X=x)\sum_{x\in\hat{x}}\frac{P(X=x)}{\sum_{x\in\hat{x}}P(X=x)}\log\frac{P(X=x)}{\sum_{x\in\hat{x}}P(X=x)}, \tag{6.150}$$

where $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_B)$ and $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_B)$ are specific realizations of the numbers of permuted and real sparse subpackets in each of the B segments, respectively, and $x \in \hat{X}$ corresponds to each realization of X that result in $\hat{x}_1, \ldots, \hat{x}_B$ numbers of sparse subpackets in the B segments. In general, the random variable $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_B)$ represents the numbers of (real) sparse subpackets updated by the user in each of the B segments, such that they sum up to Pr. Note that we do not assume any specific distribution of X in this calculation. Observing that

$\sum_{x\in\hat{x}} P(X=x) = P(\hat{X}=\hat{x})$, the conditional entropy in (6.150) simplifies to,

$$H(X|Y) = \sum_{\hat{x}:\sum_{i=1}^{B}\hat{x}_i=Pr} P(\hat{X}=\hat{x})\log P(\hat{X}=\hat{x})$$

$$- \sum_{\hat{x}:\sum_{i=1}^{B}\hat{x}_i=Pr} \sum_{x\in\hat{x}} P(X=x)\log P(X=x) \tag{6.151}$$

$$= -H(\hat{X}) + H(X), \tag{6.152}$$

since all realizations of X satisfy $\sum_{i=1}^{B}\hat{x}_i = Pr$, based on the given uplink sparsification rate. Therefore,

$$I(X;Y) = H(X) - H(X|Y) = H(\hat{X}) = H(\hat{X}_1,\ldots,\hat{X}_B). \tag{6.153}$$

Cases 3 and 4: In cases 3 and 4, we consider permutations within segments as well as among segments to reduce the information leakage further. Recall that the permutations within the B segments are denoted by $\{\tilde{P}_i\}_{i=1}^{B}$ and the permutation among segments is denoted by $\hat{P}$. Similar to the above calculation, in order to calculate the information leakage $I(X;Y)$, we first compute the conditional distribution given by,

$$P(X=x|Y=y)$$

$$= \Bigg(\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_{\hat{p}} P(Y=y|X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)$$

$$\times P(X=x)P(\hat{P}=\hat{p})\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i)\Bigg) / P(Y=y) \tag{6.154}$$

$$= \frac{P(X=x)}{P(Y=y)} \frac{1}{B!} \frac{1}{(P/B)!^B} \sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_{\hat{p}} \mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}} \quad (6.155)$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)} \frac{1}{B!} \frac{1}{(P/B)!^B} \prod_{i=1}^{B} \hat{y}_i!(P/B-\hat{y}_i)! \prod_{i=1}^{B} K_{\hat{y}_i}, & \text{for } x,y : \{\hat{x}\} = \{\hat{y}\} \\ 0, & \text{otherwise} \end{cases} \quad (6.156)$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)} \frac{1}{B!} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}, & \text{for } x,y : \{\hat{x}\} = \{\hat{y}\} \\ 0, & \text{otherwise} \end{cases} \quad (6.157)$$

where $K_{\hat{y}_i} = \sum_{j=i}^{B} \mathbf{1}_{\hat{y}_j=\hat{y}_i}$, (i.e., number of segments after (and including) segment i with equal number of sparse subpackets as that of segment i)[12] and the notation $\{\hat{x}\} = \{\hat{y}\}$ implies that the two sets $\hat{x}$ and $\hat{y}$ are the same, irrespective of their order, i.e., if $\{\hat{x}\} = \{\hat{y}\}$, for each $\hat{x}_i \in \hat{x}$, there exist some $\hat{y}_j \in \hat{y}$, such that $\hat{x}_i = \hat{y}_j$, and vice versa. Next we calculate $P(Y=y)$ for any y such that $\sum_{i=1}^{Pr} \hat{y}_i = Pr$ as,

$$P(Y=y) = \sum_{x} \sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_{\hat{p}} P(Y=y|X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)$$
$$\times P(X=x)P(\hat{P}=\hat{p}) \prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i) \quad (6.158)$$

$$= \sum_{x} P(X=x) \frac{1}{B!} \frac{1}{(P/B)!^B} \sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_{\hat{p}} \mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}} \quad (6.159)$$

$$= \frac{1}{B!} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \quad (6.160)$$

[12]This definition of $K_{\hat{y}_i}$ and the term $\prod_{i=1}^{B} K_{\hat{y}_i}$ in (6.156) makes sure that all distinct permutations within each set of segments with equal number of sparse subpackets is counted in the term $\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \sum_{\hat{p}} \mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}}$ of (6.155).

Therefore, from (6.157),

$$P(X=x|Y=y) = \begin{cases} \frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)}, & \text{for } x, y : \{\hat{x}\} = \{\hat{y}\} \\ 0, & \text{otherwise} \end{cases}. \tag{6.161}$$

Then,

$$H(X|Y=y)$$

$$= -\sum_{x} P(X=x|Y=y) \log P(X=x|Y=y) \tag{6.162}$$

$$= -\sum_{x:\{\hat{x}\}=\{\hat{y}\}} \frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \tag{6.163}$$

$$= \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right) - \frac{1}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x), \tag{6.164}$$

from which we obtain,

$$H(X|Y)$$

$$= \sum_{y} P(Y=y) H(X|Y=y) \tag{6.165}$$

$$= \sum_{y} \frac{1}{B!} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$
$$- \sum_{y} \frac{1}{B!} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \frac{1}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)}$$
$$\times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{6.166}$$

$$= \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \prod_{i=1}^{B} \binom{P/B}{\hat{y}_i} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$
$$- \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \prod_{i=1}^{B} \binom{P/B}{\hat{y}_i} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{6.167}$$

$$= \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \prod_{i=1}^{B} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$
$$- \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \prod_{i=1}^{B} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{6.168}$$

$$= \frac{1}{B!} \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \frac{B!}{\prod_{i=1}^{B} K_{\hat{y}_i}} \prod_{i=1}^{B} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$
$$- \frac{1}{B!} \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \frac{B!}{\prod_{i=1}^{B} K_{\hat{y}_i}} \prod_{i=1}^{B} K_{\hat{y}_i} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{6.169}$$

$$= \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$
$$- \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x), \tag{6.170}$$

where $\tilde{y}$ introduced in (6.169) and $\tilde{x}$ are the realizations of corresponding random variables $\tilde{Y}$ and $\tilde{X}$ representing all distinct sets of $\hat{y}$ and $\hat{x}$, respectively. For example, if $B = 2$, $(1, 2)$ and $(2, 1)$ are considered to be two different realizations of $\hat{y}$ (or $\hat{x}$), while it is the same realization of $\tilde{y}$ (or $\tilde{x}$). Moreover,

$$P(\tilde{X} = \tilde{x}) = \sum_{x \in \tilde{x}} P(X = x) \tag{6.171}$$
$$= \sum_{\text{all permutations of } x \in \hat{x}} P(X = x) \tag{6.172}$$

With this notation, the above entropy is further simplified to,

$$H(X|Y) = \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right)$$

$$- \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x) \tag{6.173}$$

$$= \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} P(\tilde{X}=\tilde{y})\log P(\tilde{X}=\tilde{y}) + H(X) \tag{6.174}$$

$$= -H(\tilde{X}) + H(X). \tag{6.175}$$

Therefore, the information leakage is given by,

$$I(X;Y) = H(X) - H(X|Y) = H(\tilde{X}) = H(\tilde{X}_1, \ldots, \tilde{X}_B). \tag{6.176}$$

6.5 Conclusions

In this chapter, we considered the problem of private FL with top r sparsification. In FL with top r sparsification, the values and the positions of the sparse updates/parameters leak information about the user's private data. We proposed four schemes with different properties to perform FL with top r sparsification without revealing the values or the positions of the sparse updates/parameters to the databases. The schemes follow a permutation technique which requires a large storage cost. To this end, we generalized the schemes to incur a reduced storage cost at the expense of a certain amount of information leakage, using a model segmentation mechanism. The four proposed schemes differ from each other based on their

storage structures, i.e., MDS coded storage/uncoded storage, and the number of permutation stages. It was shown that having two stages of permutations reduces the information leakage significantly compared to what was achieved with one. The intuition behind the expressions derived for the information leakage in the four cases is as follows. When no segmentation is present, the user sends a random set of indices (in the writing phase) corresponding to the real sparse update indices based on some random permutation known only to the user, which leaks zero information. However, when segmentation is present, the user sends the permuted indices of each segment separately, which reveals information the databases about the numbers of sparse updates in each segment, even though the indices are permuted. This is what is reflected by the entropy expressions in the derivations for the information leakage. With single-stage permutations, the user only caries out within-segment permutations, which increases the user's information leakage with increasing number of segments. This is because it reveals more information on how the sparse updates are distributed across the model, to the databases. With two-stage permutations, the user carries out both within and inter-segment permutations, which reduces the information leakage after a certain number of segments as the permuted segment indices also increase the uncertainty at the databases on the mapping between the real sparse indices and what is received.

Based on the requirements of a given FL system, one could select one of the four schemes with the parameters tuned to minimize the total communication cost while satisfying the given information leakage budget and the storage limitations of databases. In other words, this work presents the trade-off between the communi-

cation cost, storage complexity and information leakage in private FL with top r sparsification.

CHAPTER 7

Private Read-Update-Write (PRUW) with Storage Constrained Databases

7.1 Introduction

In this chapter, we study the problem of PRUW with storage constrained databases in the context of FSL. The problem considers the practical scenario where the multiple non-colluding databases in the PRUW setting are allowed to have arbitrary storage constraints. The goal of this work is to develop read-write schemes and storage mechanisms for FSL that efficiently utilize the available storage in each database to store the submodel parameters in such a way that the total communication cost is minimized while guaranteeing information-theoretic privacy of the updating submodel index and the values of the updates. As the main result, we consider both heterogeneous and homogeneous storage constrained databases, and propose private read-write and storage schemes for the two cases.

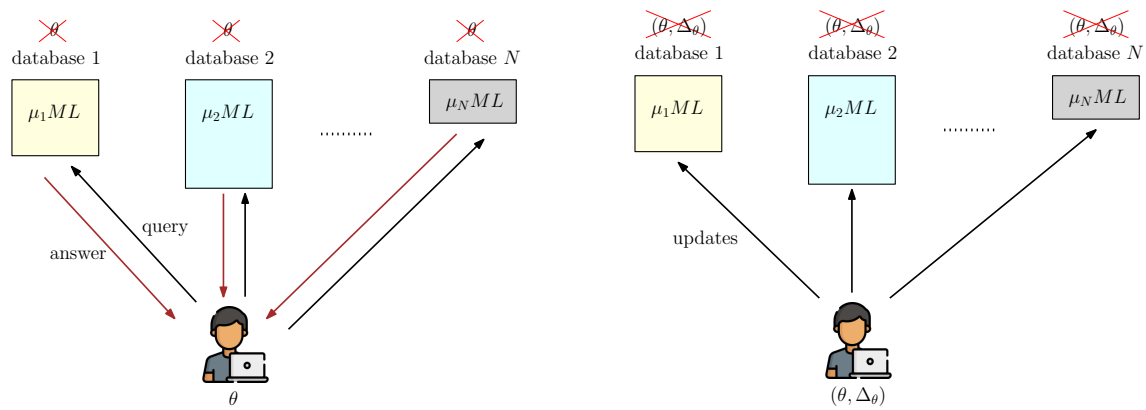
7.2 Problem Formulation

We consider an FSL setting where the model to be trained consists of M independent submodels, each containing L parameters, taking values from a finite field $\mathbb{F}_q$. The submodels are stored in a system of N, $N \geq 4$, non-colluding databases. Database n has a storage capacity of $\mu(n)ML$ symbols, where $\mu(n) \in (0, 1]$ for $n \in \{1, \ldots, N\}$. In other words, each database must satisfy

$$H(S_n) \leq \mu(n)ML, \quad n \in \{1, \ldots, N\}, \tag{7.1}$$

where S_n is the content of database n.

The storage constraints $\mu(n)$ can be divided into two main categories, namely, heterogeneous constraints where there exists at least one pair of distinct storage constraints in the system, i.e., $\mu(\tilde{n}) \neq \mu(n)$ for some $\tilde{n} \neq n$, $\tilde{n}, n \in \{1, \ldots, N\}$, and homogeneous constraints that satisfy $\mu(\tilde{n}) = \mu(n)$ for all $\tilde{n}, n \in \{1, \ldots, N\}$. At any given time instance, a user downloads a required submodel by sending queries to all databases, without revealing the required submodel index to any of the databases. This is the reading phase of the PRUW process. The user then updates the submodel using the local data, and uploads the updates back to the same submodel in all databases without revealing the values of the updates or the updating submodel index to any of the databases. This is known as the writing phase. The two phases of the PRUW process are illustrated in Fig. 7.1. The following formal privacy and security constraints must be met in the PRUW process.



(a) Reading phase. (b) Writing phase.

Figure 7.1: A user reads a submodel, updates it, and writes it back to the databases.

Privacy of the submodel index: At any given time t, no information on the indices of submodels updated up to time t, $\theta^{[1:t]}$, is allowed to leak to any of the databases,[1] i.e., for each n, $n \in \{1, \ldots, N\}$,

$$I(\theta^{[1:t]}; Q_n^{[1:t]}, U_n^{[1:t]}, S_n^{[0:t]}) = 0, \tag{7.2}$$

where Q_n and U_n are the queries and information on updates sent by the user to database n at time instances denoted in square brackets in the reading and writing phases, respectively.

Privacy of the values of the updates: At any given time t, no information on the values of the updates generated by the user up to time t, $\Delta_\theta^{[1:t]}$, is allowed to leak to any of the databases, i.e., for each n, $n \in \{1, \ldots, N\}$,

$$I(\Delta_\theta^{[1:t]}; U_n^{[1:t]}, Q_n^{[1:t]}, S_n^{[0:t]}) = 0. \tag{7.3}$$

[1]The notation $[1:t]$ indicates all integers from 1 to t.

Security of the submodels: At any given time t, no information on the values of the parameters up to time t in submodels is allowed to leak to any of the databases, i.e., for each n, $n \in \{1, \ldots, N\}$,

$$I(W_{1:M}^{[0:t]}; U_n^{[1:t]}, Q_n^{[1:t]}, S_n^{[0:t]}) = 0, \tag{7.4}$$

where W_k represents the values of the parameters in the kth submodel at the time instance stated within brackets. Apart from the privacy and security guarantees, the process requires the following correctness conditions to be met in the reading and writing phases to ensure the reliability of the FSL process.

Correctness in the reading phase: The user at time t should be able to correctly download the required submodel from the answers received in the reading phase, i.e.,

$$H(W_\theta^{[t-1]} | Q_{1:N}^{[t]}, A_{1:N}^{[t]}) = 0, \tag{7.5}$$

where $A_n^{[t]}$ is the answer received from database n at time t.

Correctness in the writing phase: The submodels in all databases must be correctly updated as,

$$W_m^{[t]} = \begin{cases} W_m^{[t-1]} + \Delta_m^{[t]}, & \text{if } m = \theta^{[t]} \\ W_m^{[t-1]}, & \text{if } m \neq \theta^{[t]}. \end{cases} \tag{7.6}$$

The reading and writing costs are defined as $C_R = \frac{\mathcal{D}}{L}$ and $C_W = \frac{\mathcal{U}}{L}$, respec-

tively, where $\mathcal{D}$ is the total number of symbols downloaded in the reading phase and $\mathcal{U}$ is the total number of symbols uploaded in the writing phase. The total cost C_T is the sum of the reading and writing costs, i.e., $C_T = C_R + C_W$.

7.3 Main Result

Theorems 7.1 and 7.2 in this section present the achievable total communication costs of the proposed schemes for heterogeneous and homogeneous storage constraints, respectively.

Theorem 7.1 *In a PRUW setting with N non-colluding databases having arbitrary heterogeneous storage constraints $\mu(n)$, $n \in \{1, \ldots, N\}$, let $k = \frac{1}{\max_n \mu(n)}$, $p = \sum_{n=1}^{N} \mu(n)$, $r = kp$ and $s = \lfloor k \rfloor p$. Then, there exist a storage mechanism and a PRUW scheme that completely fills the N databases and achieves a total communication cost of $\mathcal{C} = \min\{C_1, C_2\}$, where*

$$C_1 = (\lceil s \rceil - s) C_T(\lfloor k \rfloor, \lfloor s \rfloor) + (s - \lfloor s \rfloor) C_T(\lfloor k \rfloor, \lceil s \rceil) \tag{7.7}$$

$$\begin{aligned} C_2 = {} & \alpha\beta C_T(\lfloor k \rfloor, \lfloor r \rfloor) + \alpha(1-\beta) C_T(\lfloor k \rfloor, \lceil r \rceil) + (1-\alpha)\delta C_T(\lceil k \rceil, \lfloor r \rfloor) \\ & + (1-\alpha)(1-\delta) C_T(\lceil k \rceil, \lceil r \rceil) \end{aligned} \tag{7.8}$$

where a, β and δ are given as,

$$\alpha = \begin{cases} \frac{\lfloor k \rfloor (p\lceil k \rceil - \lceil r \rceil)}{\lceil k \rceil \lfloor r \rfloor - \lfloor k \rfloor \lceil r \rceil}, & \text{if } r - \lfloor r \rfloor > k - \lfloor k \rfloor,\ s \leq \lfloor r \rfloor \\ \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k), & \text{else} \end{cases} \tag{7.9}$$

$$\beta = \begin{cases} \frac{\lceil r \rceil - r}{\lceil k \rceil - k}, & \textit{if } r - \lfloor r \rfloor > k - \lfloor k \rfloor,\ s > \lfloor r \rfloor \\ 1, & \textit{else} \end{cases} \tag{7.10}$$

$$\delta = \begin{cases} 1 - \frac{r - \lfloor r \rfloor}{k - \lfloor k \rfloor}, & \textit{if } r - \lfloor r \rfloor \leq k - \lfloor k \rfloor \\ 0, & \textit{else} \end{cases}, \tag{7.11}$$

if $\lfloor r \rfloor - \lfloor k \rfloor$ *is odd, and*

$$\alpha = \begin{cases} \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k), & \textit{if } r - \lfloor r \rfloor < \lceil k \rceil - k \\ \frac{\lfloor k \rfloor (p \lceil k \rceil - \lfloor r \rfloor)}{\lceil k \rceil \lceil r \rceil - \lfloor k \rfloor \lfloor r \rfloor}, & \textit{else} \end{cases} \tag{7.12}$$

$$\beta = \begin{cases} 1 - \frac{r - \lfloor r \rfloor}{\lceil k \rceil - k}, & \textit{if } r - \lfloor r \rfloor < \lceil k \rceil - k \\ 0, & \textit{else} \end{cases} \tag{7.13}$$

$$\delta = 1 \tag{7.14}$$

if $\lfloor r \rfloor - \lfloor k \rfloor$ *is even, with the function* $C_T(a, b)$ *defined as,*

$$C_T(a, b) = \begin{cases} \frac{4b}{b - a - 1}, & \textit{if } b - a \textit{ is odd} \\ \frac{4b - 2}{b - a - 2}, & \textit{if } b - a \textit{ is even.} \end{cases} \tag{7.15}$$

Remark 7.1 *The intuition behind the value* p *is the maximum number of times a given uncoded parameter can be replicated within the system of databases. The values* r *and* s *represent the number of times the parameters can be replicated if they are* k *and* $\lfloor k \rfloor$ *coded, respectively.*

Remark 7.2 *As an illustration of the main result, consider the following example. Let* $\max_n \mu(n) = 0.37$ *and* $p = \sum_{n=1}^{N} \mu(n) = 4.3$. *Then,* $k = 2.7$, $r = 11.61$ *and* $s = 8.6$, *which results in* $C_1 = 6.6$ *as shown by the blue star in Fig. 7.2. Since* $\lfloor r \rfloor - \lfloor k \rfloor = 9$ *is odd,* $s = 8.6 < 11 = \lfloor r \rfloor$ *and* $r - \lfloor r \rfloor = 0.61 < 0.7 = k - \lfloor k \rfloor$, *for the calculation of* C_2, α, β *and* δ *are given by* $\alpha = \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) = \frac{2}{9}$, $\beta = 1$ *and* $\delta = 1 - \frac{r - \lfloor r \rfloor}{k - \lfloor k \rfloor} = \frac{9}{70}$, *respectively. Hence,* $C_2 = 5.99$ *(blue dot in Fig. 7.2). Note that* C_1 *is obtained by storing* $\lceil s \rceil - s$ *and* $s - \lfloor s \rfloor$ *fractions of parameters of all submodels using* $(\lfloor k \rfloor, \lfloor s \rfloor)$ *and* $(\lfloor k \rfloor, \lceil s \rceil)$ *MDS codes, respectively. Similarly,* C_2 *is obtained by storing* α, $(1-\alpha)\delta$ *and* $(1-\alpha)(1-\delta_1)$ *fractions of all submodels using* $(\lfloor k \rfloor, \lfloor r \rfloor)$, $(\lceil k \rceil, \lfloor r \rfloor)$ *and* $(\lceil k \rceil, \lceil r \rceil)$ *MDS codes, respectively. This is illustrated in Fig. 7.2. The minimum achievable cost for this example is* $\min\{C_1, C_2\} = C_2$.

Remark 7.3 *The values of* a *and* b *in the total cost expression in* (7.15) *represent the coding parameter, i.e., the number of parameters linearly combined to a single symbol, and the number of databases each parameter is replicated at, respectively. The total communication cost decreases with the number of replications* b, *and increases with the coding parameter* a, *as shown in* (7.15). *However, when the number of replications is chosen as a linear function of the coding parameter, i.e.,* $r = kp$, $s = \lfloor k \rfloor p$, *the total cost decreases with the coding parameter (see Section 7.4.2.2). Therefore,* C_2 *in the main result uses the maximum* k *by considering both* $\lfloor k \rfloor$ *and* $\lceil k \rceil$ *when* $k \notin \mathbb{Z}^+$. *However, based on the fluctuating structure of the total cost in* (7.15) *(dotted lines in Fig. 7.2) for special cases of* k *and* p, *a lower total cost can be achieved by only considering* $\lfloor k \rfloor$. *These cases are handled by* C_1.

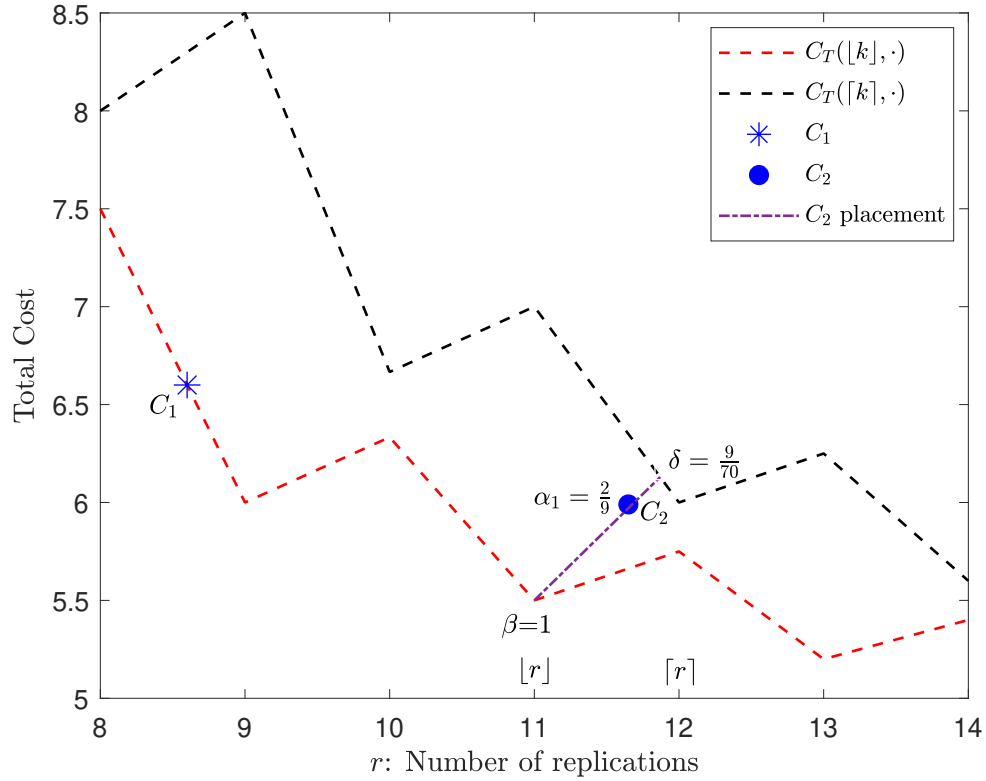


Figure 7.2: Example setting with $k = 2.7$ and $p = 4.3$.

Remark 7.4 *For a given PRUW setting with arbitrary storage constraints* $\{\mu(n)\}_{n=1}^{N}$, *let the total available storage be* $p = \sum_{n=1}^{N} \mu(n)$. *Then, there exists an uncoded PRUW scheme (the proposed scheme in Section 7.4.2 with* $k = 1$ *and* $r = p$*) that replicates* $\lceil p \rceil - p$ *and* $p - \lfloor p \rfloor$ *fractions of uncoded submodel parameters in* $\lfloor p \rfloor$ *and* $\lceil p \rceil$ *databases, respectively, achieving a total cost of* $(\lceil p \rceil - p)C_T(1, \lfloor p \rfloor) + (p - \lfloor p \rfloor)C_T(1, \lceil p \rceil)$. *Note that the total cost only depends on the sum of all arbitrary storage constraints (not on individual constraints), which is consistent with the corresponding result of private information retrieval with uncoded heterogeneous storage constraints. However, in PRUW, the total cost is minimized when* $k = \frac{1}{\max_n \mu(n)}$ *is maximized, i.e., when all storage constraints are equal.*

Theorem 7.2 *Consider a PRUW setting consisting of N non-colluding databases with a homogeneous storage constraint given by $\mu(n) = \mu \in \left[\frac{1}{N-3}, 1\right]$ for all $n \in \{1, \ldots, N\}$. Then, there exists a scheme that satisfies all privacy and security constraints for any given $\mu \in \left[\frac{1}{N-3}, 1\right]$, and the resulting total communication cost is characterized by the lower convex hull of the following $(\bar{\mu}, C_T(\bar{\mu}))$ pairs, where $\bar{\mu}$ and $C_T(\bar{\mu})$ are given by,*

$$\left(\tilde{\mu} = \frac{R}{NK_R}, \quad C_T(\tilde{\mu}) = \frac{4R}{R - K_R - 1}\right), \quad R = 4, \ldots, N, \quad K_R = 1, \ldots, R-3, \tag{7.16}$$

satisfying $(R - K_R - 1) \bmod 2 = 0$.

Remark 7.5 *The scheme proposed for heterogeneous storage constraints is based on the idea of finding the optimum coding parameters in (K, R) MDS coded storage that minimizes the total communication cost. However, as PRUW results in lower and higher communication costs for (K, R) MDS codes with odd and even $R - K$, respectively, as shown in Fig. 7.2, the costs presented in Theorem 7.1 are generally greater than what is presented in Theorem 7.2, as the scheme proposed for homogeneous constraints are able to avoid the (K, R) MDS codes with even $R-K$. However, a direct comparison cannot be made, as in general, the scheme proposed for heterogeneous constraints is not applicable for homogeneous constraints, as shown in the proofs of Lemmas 7.1 and 7.2 in Appendices 7.7.1 and 7.7.2.*

7.4 Proposed Scheme: Heterogeneous Storage Constraints

In this section, we present the proof of Theorem 7.1. The proposed scheme for heterogeneous storage constrained PRUW consist of two components, namely, the storage mechanism and the PRUW scheme. The storage mechanism deals with finding the optimum coding parameters and optimum submodel partitions to be stored in each of the N databases that result in the minimum reading and writing costs. The PRUW scheme is what states the steps of the read-write process based on the given storage structure. The PRUW scheme we use in this work is an optimized version of the general PRUW scheme in [94]. In this section, we first present the optimized PRUW scheme, and then move on to the proposed storage mechanisms for any given set of storage constraints.

7.4.1 General PRUW scheme

In this section, we use the general PRUW scheme proposed in [94] for general (K, R) MDS coded storage, and modify it by including the optimum subpacketization[2] and the optimum number of noise terms in storage to guarantee privacy, while minimizing the reading and writing costs.

Storage: The contents of a single subpacket in database n, $n \in \{1, \ldots, R\}$,

[2]Subpacketization is the number of parameters considered in each subpacket. A subpacket is a collection of parameters of all submodels, on which the scheme is defined. The scheme is then applied repeatedly on all subpackets identically.

with a subpacketization of y and $x+1$ noise terms is given by,[3]

$$S_n = \begin{bmatrix} \sum_{i=1}^{K} \frac{1}{f_{1,i}-\alpha_n} \begin{bmatrix} W_{1,1}^{[i]} \\ \vdots \\ W_{M,1}^{[i]} \end{bmatrix} + \sum_{j=0}^{x} \alpha_n^j Z_{1,j} \\ \vdots \\ \sum_{i=1}^{K} \frac{1}{f_{y,i}-\alpha_n} \begin{bmatrix} W_{1,y}^{[i]} \\ \vdots \\ W_{M,y}^{[i]} \end{bmatrix} + \sum_{j=0}^{x} \alpha_n^j Z_{y,j} \end{bmatrix}, \tag{7.17}$$

where $W_{m,j}^{[i]}$ is the ith parameter of the jth coded symbol of submodel m (in the subpacket considered), $Z_{i,j}$ are random noise vectors of size $M \times 1$ and $f_{i,j}$s and α_ns are globally known distinct constants from $\mathbb{F}_q$. Note that the jth coded data symbol (without the noise) is stored in terms of a (K, R) MDS code, with the generator matrix given by,

$$G = \begin{bmatrix} \frac{1}{f_{j,1}-\alpha_1} & \frac{1}{f_{j,2}-\alpha_1} & \cdots & \frac{1}{f_{j,K}-\alpha_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_{j,1}-\alpha_R} & \frac{1}{f_{j,2}-\alpha_R} & \cdots & \frac{1}{f_{j,K}-\alpha_R} \end{bmatrix}, \quad j \in \{1, \ldots, y\}. \tag{7.18}$$

The variables K, R, x and y are optimized to achieve minimum reading and writing costs at the end of this section.

Reading phase: In the reading phase, the user sends queries to all databases

[3]Here, we consider storing each subpacket in only R databases out of N, since all subpackets cannot be stored in all databases due to limited storage capacities in databases. R is a variable which will be optimized later.

to download the required submodel W_θ. The queries sent to database n, $n \in \{1, \ldots, R\}$, to download the $K \times y$ parameters of each subpacket of the required submodel is given by,

$$Q_{n,\ell} = \begin{bmatrix} \frac{\prod_{i=1, i\neq\ell}^{K}(f_{1,i}-\alpha_n)}{\prod_{i=1, i\neq\ell}^{K}(f_{1,i}-f_{1,\ell})} e_M(\theta) + \prod_{i=1}^{K}(f_{1,i}-\alpha_n)\tilde{Z}_{1,\ell} \\ \vdots \\ \frac{\prod_{i=1, i\neq\ell}^{K}(f_{y,i}-\alpha_n)}{\prod_{i=1, i\neq\ell}^{K}(f_{y,i}-f_{y,\ell})} e_M(\theta) + \prod_{i=1}^{K}(f_{y,i}-\alpha_n)\tilde{Z}_{y,\ell} \end{bmatrix}, \quad \ell \in \{1, \ldots, K\}, \tag{7.19}$$

where $e_M(\theta)$ is the all zeros vector of size $M \times 1$ with a 1 at the θth position and $\tilde{Z}_{j,\ell}$ are random noise vectors of size $M \times 1$. Then, database n, $n \in \{1, \ldots, R\}$, sends the corresponding answers given by,

$$A_{n,\ell} = S_n^T Q_{n,\ell}, \quad \ell \in \{1, \ldots, K\} \tag{7.20}$$

$$= \begin{bmatrix} \sum_{i=1}^{K} \frac{1}{f_{1,i}-\alpha_n} \begin{bmatrix} W_{1,1}^{[i]} \\ \vdots \\ W_{M,1}^{[i]} \end{bmatrix} + \sum_{j=0}^{x} \alpha_n^j Z_{1,j} \\ \vdots \\ \sum_{i=1}^{K} \frac{1}{f_{y,i}-\alpha_n} \begin{bmatrix} W_{1,y}^{[i]} \\ \vdots \\ W_{M,y}^{[i]} \end{bmatrix} + \sum_{j=0}^{x} \alpha_n^j Z_{y,j} \end{bmatrix}^T \begin{bmatrix} \frac{\prod_{i=1, i\neq\ell}^{K}(f_{1,i}-\alpha_n)}{\prod_{i=1, i\neq\ell}^{K}(f_{1,i}-f_{1,\ell})} e_M(\theta) + \prod_{i=1}^{K}(f_{1,i}-\alpha_n)\tilde{Z}_{1,\ell} \\ \vdots \\ \frac{\prod_{i=1, i\neq\ell}^{K}(f_{y,i}-\alpha_n)}{\prod_{i=1, i\neq\ell}^{K}(f_{y,i}-f_{y,\ell})} e_M(\theta) + \prod_{i=1}^{K}(f_{y,i}-\alpha_n)\tilde{Z}_{y,\ell} \end{bmatrix} \tag{7.21}$$

$$= \sum_{j=1}^{y} \frac{1}{f_{j,\ell}-\alpha_n} W_{\theta,j}^{[\ell]} + P_{\alpha_n}(K+x), \quad \ell \in \{1, \ldots, K\}, \tag{7.22}$$

where $P_{\alpha_n}(h)$ is a polynomial in α_n of degree h, and (7.22) is a result of

$$\frac{1}{f_{j,i}-\alpha_n} \times \frac{\prod_{i=1,i\neq\ell}^{K}(f_{j,i}-\alpha_n)}{\prod_{i=1,i\neq\ell}^{K}(f_{j,i}-f_{j,\ell})} = \begin{cases} \frac{1}{f_{j,\ell}-\alpha_n} + P_{\alpha_n}(K-2), & \text{if } i=\ell \\ P_{\alpha_n}(K-2), & \text{if } i\neq\ell, \end{cases} \tag{7.23}$$

which is obtained by applying Lemma 3.1. Note that the terms corresponding to $W_{\theta,j}^{[i\neq\ell]}$ for $j=1,\ldots,y$ and $i=1,\ldots,K$ in the calculation of the answers in (7.22) are included in the combined noise polynomial $P_{\alpha_n}(K+x)$. Using the K answers from each database, the user obtains the $K\times y$ parameters of each subpacket of the required submodel W_θ by solving,

$$\begin{bmatrix} A_{1,\ell} \\ \vdots \\ A_{R,\ell} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_{1,\ell}-\alpha_1} & \ldots & \frac{1}{f_{y,\ell}-\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^{K+x} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_{1,\ell}-\alpha_R} & \ldots & \frac{1}{f_{y,\ell}-\alpha_R} & 1 & \alpha_R & \ldots & \alpha_R^{K+x} \end{bmatrix} \begin{bmatrix} W_{\theta,1}^{[\ell]} \\ \vdots \\ W_{\theta,y}^{[\ell]} \\ v_0^{[\ell]} \\ \vdots \\ v_{K+x}^{[\ell]} \end{bmatrix}, \quad \ell\in\{1,\ldots,K\}, \tag{7.24}$$

where $v_i^{[\ell]}$ is the coefficient of α_n^i in $P_{\alpha_n}(K+x)$ of (7.22). Note that the $K\times y$ parameters in each subpacket of W_θ can be obtained by (7.24) if $R=y+K+x+1$ is satsisfied. This determines the subpacketization (number of coded symbols in a subpacket of each submodel) as $y=R-K-x-1$, which results in the reading

cost given by,

$$C_R = \frac{R \times K}{y \times K} = \frac{R}{R - K - x - 1}. \tag{7.25}$$

Writing phase: In the writing phase, the user sends K symbols to each of the R databases. The ℓth symbol (out of the K symbols uploaded) is the combined update that consists of the updates of W_θ corresponding to the ℓth parameter of each of the y coded parameters in each subpacket. The K combined updates sent to database n, $n \in \{1, \ldots, R\}$, are given by,

$$U_{n,\ell} = \sum_{j=1}^{y} \prod_{i=1, i\neq j}^{y} (f_{i,\ell} - \alpha_n)\tilde{\Delta}_{\theta,j}^{[\ell]} + \prod_{i=1}^{y}(f_{i,\ell} - \alpha_n)\hat{z}_\ell, \quad \ell \in \{1, \ldots, K\}, \tag{7.26}$$

where $\tilde{\Delta}_{\theta,j}^{[\ell]} = \frac{\prod_{i=1,i\neq\ell}^{K}(f_{j,i}-f_{j,\ell})}{\prod_{i=1,i\neq j}^{y}(f_{i,\ell}-f_{j,\ell})}\Delta_{\theta,j}^{[\ell]}$ for $j \in \{1, \ldots, y\}$, with $\Delta_{\theta,j}^{[\ell]}$ being the update of the ℓth parameter of the jth coded bit of the considered subpacket in submodel θ and $\hat{z}_\ell$ is a random noise symbol. Once database n receives the update bits, it calculates the incremental update with the aid of the two matrices given by,

$$\Omega_{n,\ell} = \text{diag}\left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_{1,\ell})}1_M, \ldots, \frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_{y,\ell})}1_M\right), \tag{7.27}$$

$$\tilde{D}_{n,\ell} = \text{diag}\left(\frac{1}{\prod_{i=1}^{K}(f_{1,i} - \alpha_n)}1_M, \ldots, \frac{1}{\prod_{i=1}^{K}(f_{y,i} - \alpha_n)}1_M\right), \tag{7.28}$$

where $\Omega_{n,\ell}$ is the null shaper in [94] with $\mathcal{F}$ being any subset of randomly chosen databases satisfying $|\mathcal{F}| = x - y$. The null shaper is used to place some of the zeros ($x - y$ zeros) of the incremental update polynomial at specific α_ns to reduce the

writing cost by not having to send any updates to the databases corresponding to those α_ns. Here, $\mathcal{F}$ is any subset of $x-y$ databases. $\tilde{D}_{n,\ell}$ is a scaling matrix and 1_M is the vector of all ones of size $1 \times M$. The incremental update is calculated as,

$$\bar{U}_{n,\ell} = \Omega_{n,\ell} \times U_{n,\ell} \times \tilde{D}_{n,\ell} \times Q_{n,\ell}, \quad \ell \in \{1,\dots,K\} \tag{7.29}$$

$$= \Omega_{n,\ell} \times U_{n,\ell} \times \begin{bmatrix} \frac{1}{\prod_{i=1}^{K}(f_{1,i}-\alpha_n)} \left(\frac{\prod_{i=1,i\neq\ell}^{K}(f_{1,i}-\alpha_n)}{\prod_{i=1,i\neq\ell}^{K}(f_{1,i}-f_{1,\ell})} e_M(\theta) + \prod_{i=1}^{K}(f_{1,i}-\alpha_n)\tilde{Z}_{1,\ell} \right) \\ \vdots \\ \frac{1}{\prod_{i=1}^{K}(f_{y,i}-\alpha_n)} \left(\frac{\prod_{i=1,i\neq\ell}^{K}(f_{y,i}-\alpha_n)}{\prod_{i=1,i\neq\ell}^{K}(f_{y,i}-f_{y,\ell})} e_M(\theta) + \prod_{i=1}^{K}(f_{y,i}-\alpha_n)\tilde{Z}_{y,\ell} \right) \end{bmatrix} \tag{7.30}$$

$$= \Omega_{n,\ell} \times U_{n,\ell} \times \begin{bmatrix} \frac{1}{\prod_{i=1,i\neq\ell}^{K}(f_{1,i}-f_{1,\ell})} \frac{1}{(f_{1,\ell}-\alpha_n)} e_M(\theta) + \tilde{Z}_{1,\ell} \\ \vdots \\ \frac{1}{\prod_{i=1,i\neq\ell}^{K}(f_{y,i}-f_{y,\ell})} \frac{1}{(f_{y,\ell}-\alpha_n)} e_M(\theta) + \tilde{Z}_{y,\ell} \end{bmatrix} \tag{7.31}$$

$$= \Omega_{n,\ell} \times \begin{bmatrix} \frac{1}{(f_{1,\ell}-\alpha_n)} \Delta_{\theta,1}^{[\ell]} e_M(\theta) + P_{\alpha_n}^{[1]}(y) \\ \vdots \\ \frac{1}{(f_{y,\ell}-\alpha_n)} \Delta_{\theta,y}^{[\ell]} e_M(\theta) + P_{\alpha_n}^{[y]}(y) \end{bmatrix} \tag{7.32}$$

$$= \begin{bmatrix} \frac{1}{f_{1,\ell}-\alpha_n} \Delta_{\theta,1}^{[\ell]} e_M(\theta) + P_{\alpha_n}^{[1]}(x) \\ \vdots \\ \frac{1}{f_{y,\ell}-\alpha_n} \Delta_{\theta,y}^{[\ell]} e_M(\theta) + P_{\alpha_n}^{[y]}(x) \end{bmatrix}, \quad \ell \in \{1,\dots,K\}, \tag{7.33}$$

where $P_{\alpha_n}^{[j]}(h)$ here is a vector of size $M \times 1$ consisting of polynomials in α_n of degree h. Note that (7.32) and (7.33) are obtained by applying Lemma 3.1 and Lemma 3.2, respectively. Since the incremental updates are of the same form as the storage in

(7.17), the submodels are updated by,

$$S_n(t) = S_n(t-1) + \sum_{\ell=1}^{K} \bar{U}_{n,\ell}. \tag{7.34}$$

The resulting writing cost is,

$$C_W = \frac{K \times (R - (x-y))}{K \times y} = \frac{2R - 2x - K - 1}{R - x - K - 1}, \tag{7.35}$$

which together with the reading cost in (7.25) gives the total cost,

$$C_T = C_R + C_W = \frac{3R - 2x - K - 1}{R - x - K - 1}. \tag{7.36}$$

The total cost is an increasing function of x since $\frac{dC_T}{dx} = \frac{R+K+1}{(R-x-K-1)^2} > 0$. Note that $x \geq y$ must be satisfied by x in order to write to y parameters using a single symbol. This is because the decomposition of the combined update in (7.32) results in a noise polynomial of degree y, which requires the existing storage to have at least a degree y noise polynomial, as the incremental update is added to the existing storage in the updating process. Therefore, the optimum value of x that minimizes the total cost is,

$$x = \begin{cases} y = \frac{R-K-1}{2}, & \text{if } R-K \text{ is odd,} \\ y+1 = \frac{R-K}{2}, & \text{if } R-K \text{ is even.} \end{cases} \tag{7.37}$$

The resulting minimum total cost of the PRUW process with (K, R) MDS coded

storage is,

$$C_T = \begin{cases} \frac{4R}{R-K-1}, & \text{if } R-K \text{ is odd,} \\ \\ \frac{4R-2}{R-K-2}, & \text{if } R-K \text{ is even.} \end{cases} \tag{7.38}$$

Note that since the subpacketization $y \geq 1$, R and K must satisfy,

$$1 \leq K \leq \begin{cases} R-3, & \text{if } R-K \text{ is odd,} \\ \\ R-4, & \text{if } R-K \text{ is even.} \end{cases} \tag{7.39}$$

7.4.2 Storage Mechanism

The proposed storage mechanism consists of submodel partitioning and submodel encoding, where the parameters are encoded with a specific (K, R) MDS code and stored at different subsets of databases in parts. In this section, we find the optimum coding parameters and fractions of submodels stored in each database to minimize the total cost.

7.4.2.1 Submodel Partitioning

This determines the fractions of all (K, R) MDS coded submodels to be stored in the N databases with arbitrary storage constraints. Based on the (K, R) MDS coded structure, each coded parameter must be stored in exactly R databases. Let η_i be the fraction of all submodels that are stored in the same subset of R databases (the subset of databases indexed by i). Let $\mathcal{B}$ be the basis containing all $N \times 1$ vectors

with elements in $\{0,1\}$ with exactly R ones, denoted by $\mathcal{B} = \{b_1, b_2, \ldots, b_{|\mathcal{B}|}\}$. Note that each b_i corresponds to a specific subset of R databases, indexed by i. Then, it is required to find the η_is that satisfy,

$$\frac{1}{K}\sum_{i=1}^{|\mathcal{B}|}\eta_i b_i = \mu, \quad \text{where} \quad \mu = [\mu(1), \ldots, \mu(N)]^T \tag{7.40}$$

$$\sum_{i=1}^{|\mathcal{B}|}\eta_i = 1, \quad 0 \leq \eta_1, \ldots, \eta_{|\mathcal{B}|} \leq 1, \tag{7.41}$$

to replicate each coded parameter at exactly R databases, while ensuring that all databases are completely filled. The solution to this problem with uncoded parameters ($K = 1$) is provided in [18] and [24] along with a necessary and sufficient condition for a solution to exist, given by,

$$\mu(n) \leq \frac{\sum_{n=1}^{N}\mu(n)}{R}, \quad n \in \{1, \ldots, N\}. \tag{7.42}$$

The intuition behind this condition is as follows. Consider database n which can store up to $\mu(n)ML$ bits. If each bit is expected to be replicated at R databases, the total available storage in all databases $\sum_{i=1}^{N}\mu(n)ML$ must be greater than or equal to $\mu(n)MLR$, to successfully replicate the $\mu(n)ML$ bits of database n in R databases. This must be satisfied by all N databases, which results in (7.42). Note that this condition is valid for any (K, R) MDS coded setting, as the available storage in each database does not change with the submodel encoding structure.

In this work, we find the optimum coding parameters K and R that result in the minimum total cost of the PRUW process while satisfying (7.42), and solve

(7.40)-(7.41) to find the partitions η_i of coded submodels to be stored in each database.

7.4.2.2 Submodel Encoding

For a given set of storage constraints $\{\mu(n)\}_{n=1}^{N}$, the total available storage is $p = \sum_{n=1}^{N} \mu(n)$ where the intuition behind the value of p is the maximum number of times each parameter in the model can be replicated in the system of databases if the parameters are uncoded. Let the parameters of the model be stored using a (K, R) MDS code. Therefore, the total number of coded parameters in the model is $\frac{ML}{K}$, which allows each coded parameter to be replicated at a maximum of,

$$R \leq \frac{\sum_{n=1}^{N} \mu(n) ML}{\frac{ML}{K}} = Kp \tag{7.43}$$

databases, given that the value of K satisfies $K \leq \frac{N}{p}$. At the same time, to completely utilize the available space in each database, the number of parameters stored in each database must satisfy,

$$\mu(n) ML \leq \frac{ML}{K}, \quad n \in \{1, \ldots, N\} \implies K \leq \frac{1}{\bar{\mu}}, \tag{7.44}$$

where $\bar{\mu} = \max_n \mu(n)$. Therefore, the coding parameter K must satisfy,

$$K \leq \min\left\{\frac{1}{\bar{\mu}}, \frac{N}{p}\right\} = \frac{1}{\bar{\mu}}, \tag{7.45}$$

as $p = \sum_{n=1}^{N} \mu(n) \leq \bar{\mu}N$ implies $\frac{1}{\bar{\mu}} \leq \frac{N}{p}$. Since the total cost of the PRUW scheme decreases with the number of replications R (see (7.38)), for given $K, p \in \mathbb{Z}^+$ satisfying (7.45), the optimum R for the (K, R) MDS code is given by Kp (from (7.43)), and the resulting total cost is,

$$C_T(K, R) = \begin{cases} \frac{4Kp}{Kp-K-1}, & \text{odd } K, \text{ even } p, \\ \\ \frac{4Kp-2}{Kp-K-2}, & \text{otherwise,} \end{cases} \tag{7.46}$$

which decreases with K. Therefore, for a given set of storage constraints $\{\mu(n)\}_{n=1}^{N}$, the minimum total cost is achieved by the (K, R) MDS code in (7.17) with K and R given by,

$$K = \frac{1}{\bar{\mu}} \stackrel{\text{def}}{=} k, \qquad R = kp \stackrel{\text{def}}{=} r \tag{7.47}$$

which automatically satisfies (7.42) since $\frac{\sum_{n=1}^{N} \mu(n)}{r} = \bar{\mu} \geq \mu(n)$ for all n. However, since k and r are not necessarily integers, we need additional calculations to obtain the optimum (k, r) MDS codes with $k, r \in \mathbb{Z}^+$ that collectively result in the lowest total cost.

Consider the general case where $k, r \notin \mathbb{Z}^+$. The general idea here is to divide each submodel into four sections and encode them using the four MDS codes given in Table 7.1. The sizes (fractions) of the four sections are defined by the parameters α, β and δ, where $0 \leq \alpha, \beta, \delta \leq 1$ (see Table 7.1). The total spaces allocated for the

four MDS codes in the entire system of databases are given by,

$$\sum_{n=1}^{N} \hat{\mu}_1(n) = \frac{\alpha\beta}{\lfloor k \rfloor} \lfloor r \rfloor \tag{7.48}$$

$$\sum_{n=1}^{N} \hat{\mu}_2(n) = \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \lceil r \rceil \tag{7.49}$$

$$\sum_{n=1}^{N} \bar{\mu}_1(n) = \frac{(1-\alpha)\delta}{\lceil k \rceil} \lfloor r \rfloor \tag{7.50}$$

$$\sum_{n=1}^{N} \bar{\mu}_2(n) = \frac{(1-\alpha)(1-\delta)}{\lceil k \rceil} \lceil r \rceil \tag{7.51}$$

case	MDS code	fraction of submodel	space allocated in database n
1	$(\lfloor k \rfloor, \lfloor r \rfloor)$	$\alpha\beta$	$\hat{\mu}_1(n)$
2	$(\lfloor k \rfloor, \lceil r \rceil)$	$\alpha(1-\beta)$	$\hat{\mu}_2(n)$
3	$(\lceil k \rceil, \lfloor r \rfloor)$	$(1-\alpha)\delta$	$\bar{\mu}_1(n)$
4	$(\lceil k \rceil, \lceil r \rceil)$	$(1-\alpha)(1-\delta)$	$\bar{\mu}_2(n)$

Table 7.1: Fractions of submodels and corresponding MDS codes.

The space allocated for each individual MDS code in each database must satisfy (7.42) separately, to ensure that all databases are completely filled while also replicating each coded parameter at the respective number of databases, i.e.,

$$\hat{\mu}_1(n) \leq \frac{\alpha\beta}{\lfloor k \rfloor} \tag{7.52}$$

$$\hat{\mu}_2(n) \leq \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \tag{7.53}$$

$$\bar{\mu}_1(n) \leq \frac{(1-\alpha)\delta}{\lceil k \rceil} \tag{7.54}$$

$$\bar{\mu}_2(n) \leq \frac{(1-\alpha)(1-\delta)}{\lceil k \rceil}. \tag{7.55}$$

All four storage allocations in each database must satisfy,

$$\hat{\mu}_1(n) + \hat{\mu}_2(n) + \bar{\mu}_1(n) + \bar{\mu}_2(n) = \mu(n), \quad \forall n. \tag{7.56}$$

It remains to find the values of $\hat{\mu}_1(n)$, $\hat{\mu}_2(n)$, $\bar{\mu}_1(n)$, $\bar{\mu}_2(n)$, α, β and δ that minimize the total cost given by,

$$\begin{aligned} C = {} & \alpha\beta C_T(\lfloor k \rfloor, \lfloor r \rfloor) + \alpha(1-\beta) C_T(\lfloor k \rfloor, \lceil r \rceil) + (1-\alpha)\delta C_T(\lceil k \rceil, \lfloor r \rfloor) \\ & + (1-\alpha)(1-\delta) C_T(\lceil k \rceil, \lceil r \rceil) \end{aligned} \tag{7.57}$$

while satisfying (7.48)-(7.56), where C_T is defined in (7.15). We consider two cases, 1) $\alpha = 1$, 2) $\alpha < 1$. When $\alpha = 1$, only cases 1, 2 in Table 7.1 are used in storage, and (7.42) becomes $R \leq \lfloor k \rfloor p$, which results in $\lfloor k \rfloor p = s$ maximum replications. This replaces all rs in Table 7.1 and (7.48)-(7.55) by s when $\alpha = 1$. The optimum values of the parameters in Table 7.1 that minimize the total cost in (7.57) when $\alpha = 1$ and $\alpha < 1$ are stated in Lemma 7.1 and Lemmas 7.2-7.3, respectively. For a given set of storage constraints, we choose the set of optimum values corresponding to either $\alpha = 1$ or $\alpha < 1$, based on the case that results in the minimum total cost. In other words, C_1 and C_2 stated in (7.7) and (7.8) in Section 7.3 indicate the total costs corresponding to the two cases $\alpha = 1$ and $\alpha < 1$, respectively, from which the minimum is chosen.

Lemma 7.1 *When $\alpha = 1$, β is fixed at $\beta = \lceil s \rceil - s$ to satisfy* (7.48)-(7.49)*, and*

$\hat{\mu}_1(n), \hat{\mu}_2(n)$ *satisfying* (7.52)-(7.53) *are given by,*

$$\hat{\mu}_1(n) = \tilde{m}(n) + (\mu(n) - \tilde{m}(n) - \tilde{h}(n))\tilde{\gamma} \tag{7.58}$$

$$\hat{\mu}_2(n) = \tilde{h}(n) + (\mu(n) - \tilde{m}(n) - \tilde{h}(n))(1 - \tilde{\gamma}) \tag{7.59}$$

for all $n \in \{1, \ldots, N\}$ *where,*

$$\tilde{m}(n) = \left[\mu(n) - \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor}\right]^+, \quad \tilde{h}(n) = \left[\mu(n) - \frac{\lceil s \rceil - s}{\lfloor k \rfloor}\right]^+ \tag{7.60}$$

$$\tilde{\gamma} = \frac{\frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s) - \sum_{n=1}^{N} \tilde{m}(n)}{p - \sum_{n=1}^{N} \tilde{m}(n) - \sum_{n=1}^{N} \tilde{h}(n)}, \tag{7.61}$$

with $[x]^+ = \max\{x, 0\}$. *The resulting total cost is given in* (7.7).

Lemma 7.2 *The following values of* $\hat{\mu}_1(n)$, $\hat{\mu}_2(n)$, $\bar{\mu}_1(n)$ *and* $\bar{\mu}_2(n)$ *for* $n \in \{1, \ldots, N\}$ *satisfy* (7.48)-(7.56) *with any* $\alpha < 1$, β *and* δ *that satisfy* $\alpha \geq \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k)$, $\beta \geq \left[1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor)\right]^+$ *and* $\delta \geq \left[1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor)\right]^+$,

$$\hat{\mu}_1(n) = \begin{cases} \hat{\mu}(n)\beta, & \text{if } \beta \in \{0, 1\} \\ \hat{m}(n) + (\hat{\mu}(n) - \hat{m}(n) - \hat{h}(n))\hat{\gamma}, & \text{if } \beta \in (0, 1) \end{cases} \tag{7.62}$$

$$\hat{\mu}_2(n) = \begin{cases} \hat{\mu}(n)(1 - \beta), & \text{if } \beta \in \{0, 1\} \\ \hat{h}(n) + (\hat{\mu}(n) - \hat{m}(n) - \hat{h}(n))(1 - \hat{\gamma}), & \text{if } \beta \in (0, 1) \end{cases} \tag{7.63}$$

$$\bar{\mu}_1(n) = \begin{cases} \bar{\mu}(n)\delta, & \text{if } \delta \in \{0, 1\} \\ \bar{m}(n) + (\bar{\mu}(n) - \bar{m}(n) - \bar{h}(n))\bar{\gamma}, & \text{if } \delta \in (0, 1) \end{cases} \tag{7.64}$$

$$\bar{\mu}_2(n) = \begin{cases} \bar{\mu}(n)(1-\delta), & \text{if } \delta \in \{0,1\} \\ \\ \bar{h}(n) + (\bar{\mu}(n) - \bar{m}(n) - \bar{h}(n))(1-\bar{\gamma}), & \text{if } \delta \in (0,1) \end{cases} \tag{7.65}$$

for all $n \in \{1, \ldots, N\}$ *where,*

$$\hat{\mu}(n) = m(n) + (\mu(n) - m(n) - h(n))\gamma \tag{7.66}$$

$$\bar{\mu}(n) = h(n) + (\mu(n) - m(n) - h(n))(1-\gamma) \tag{7.67}$$

$$m(n) = \left[\mu(n) - \frac{1-\alpha}{\lceil k \rceil}\right]^+, \quad h(n) = \left[\mu(n) - \frac{\alpha}{\lfloor k \rfloor}\right]^+ \tag{7.68}$$

$$\gamma = \frac{\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} m(n)}{p - \sum_{n=1}^{N} m(n) - \sum_{n=1}^{N} h(n)} \tag{7.69}$$

$$\hat{m}(n) = \left[\hat{\mu}(n) - \frac{\alpha(1-\beta)}{\lfloor k \rfloor}\right]^+, \quad \hat{h}(n) = \left[\hat{\mu}(n) - \frac{\alpha\beta}{\lfloor k \rfloor}\right]^+ \tag{7.70}$$

$$\hat{\gamma} = \frac{\frac{\alpha\beta}{\lfloor k \rfloor}\lfloor r \rfloor - \sum_{n=1}^{N} \hat{m}(n)}{\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} \hat{m}(n) - \sum_{n=1}^{N} \hat{h}(n)} \tag{7.71}$$

$$\bar{m}(n) = \left[\bar{\mu}(n) - \frac{(1-\alpha)(1-\delta)}{\lceil k \rceil}\right]^+, \quad \bar{h}(n) = \left[\bar{\mu}(n) - \frac{(1-\alpha)\delta}{\lceil k \rceil}\right]^+ \tag{7.72}$$

$$\bar{\gamma} = \frac{\frac{(1-\alpha)\delta}{\lceil k \rceil}\lfloor r \rfloor - \sum_{n=1}^{N} \bar{m}(n)}{\frac{1-\alpha}{\lceil k \rceil}(\lceil r \rceil - \delta) - \sum_{n=1}^{N} \bar{m}(n) - \sum_{n=1}^{N} \bar{h}(n)}. \tag{7.73}$$

Lemma 7.3 *For the case where* $\alpha < 1$*, the values of* α*,* β *and* δ *that minimize the total cost in* (7.57) *while satisfying the constraints in Lemma 7.2 are specified in Theorem 7.1, and the corresponding total cost is given in* (7.8)*.*

The three lemmas stated above provide the proof of Theorem 7.1. The proofs of Lemmas 7.1, 7.2 and 7.3 are given in the Appendix. Once all parameters in Table 7.1 are determined from Lemmas 7.1, 7.2, 7.3, equations (7.40)-(7.41) are solved for η_is for the four sets of storage allocations, $\{\hat{\mu}_1\}_{n=1}^N$, $\{\hat{\mu}_2\}_{n=1}^N$, $\{\bar{\mu}_1\}_{n=1}^N$,

$\{\bar{\mu}_2\}_{n=1}^{N}$ separately. These four sets of solutions determine where each individual coded symbol is replicated in the system of databases. Then, the storage structure and the scheme provided in Section 7.4.1 is used to perform private FSL.

7.4.3 Example

As an example, consider a PRUW setting with $N = 12$ databases, where $\mu(1) = \ldots \mu(5) = 0.37$ and $\mu(6) = \ldots \mu(12) = 0.35$. Note that the values of k, r and s here are the same as what is considered in Remark 7.2, and therefore result in the same α, β, δ and total cost stated in Remark 7.2, i.e., $k = 2.7$, $r = 11.61$, $s = 8.6$, $\alpha = \frac{2}{9}$, $\beta = 1$ and $\delta = \frac{9}{70}$. Using (7.62)-(7.73) we find the storage allocations as,

$$\hat{\mu}_1(n) = \hat{\mu}(n) = \begin{cases} 0.1107, & \text{for } n = 1, \ldots 5 \\ 0.0951, & \text{for } n = 6, \ldots 12 \end{cases} \tag{7.74}$$

$$\bar{\mu}_1(n) = \begin{cases} 0.033, & \text{for } n = 1, \ldots 5 \\ 0.029, & \text{for } n = 6, \ldots 12 \end{cases} \tag{7.75}$$

$$\hat{\mu}_2(n) = 0, \quad \bar{\mu}_2(n) = 0.226, \quad \forall n. \tag{7.76}$$

Note that the $(\lfloor k \rfloor, \lceil r \rceil)$ MDS code is not used as $\hat{\mu}_2(n) = 0$, $\forall n$, and the allocated space for the $(\lceil k \rceil, \lceil r \rceil)$ MDS code in all databases is the same since $\bar{\mu}_2(n) = 0.226$, $\forall n$. Therefore, the optimum storage mechanism for PRUW with homogeneous storage constraints presented in Section 7.5 is used to place the $(\lceil k \rceil, \lceil r \rceil)$ coded parameters. The storage allocations for $(\lfloor k \rfloor, \lfloor r \rfloor)$ and $(\lceil k \rceil, \lfloor r \rfloor)$ codes have different storage

allocations for different databases (two different values in (7.74), (7.75)). Therefore, for these two cases, we solve (7.40)-(7.41) separately, to find the fractions of coded parameters to be stored in each database. For example, for the $(\lfloor k \rfloor, \lfloor r \rfloor)$ MDS code, we find the solution (values of η_i) to (7.40)-(7.41) with $N = 12$, $K = \lfloor k \rfloor = 2$, $R = \lfloor r \rfloor = 11$ and $\mu = [\hat{\mu}_1(1) : \hat{\mu}_1(12)]^T$ as follows. For this case, (7.40)-(7.41) is given by,

$$\sum_{i=1}^{12} \frac{\alpha\beta ML}{\lfloor k \rfloor} \eta_i b_i = \mu ML \tag{7.77}$$

$$\sum_{i=1} \eta_i = 1, \tag{7.78}$$

where b_i is the all ones vector of size 12×1 with a zero at the ith position. As a solution, we get $\tilde{\eta}_1 = \ldots = \tilde{\eta}_5 = 0$ and $\tilde{\eta}_6 = \ldots = \tilde{\eta}_{12} = 0.0315$, where $\tilde{\eta}_i = \alpha\beta\eta_i$. To explain how the submodel parameters are stored, consider $\tilde{\eta}_6 = 0.0315$. From each submodel, a fraction of $\tilde{\eta}_6 = 0.0315$ is chosen to be replicated at all databases except for database 6. The chosen set of parameters from all submodels are $(\lfloor k \rfloor, \lfloor r \rfloor)$ MDS coded based on the structure shown in Section 7.4.1, and the PRUW process is carried out accordingly.

7.4.4 Improved Scheme for Special Cases

The key idea behind the scheme proposed in Section 7.4.2 is to find the optimum linear combination of (K, R) MDS codes to store the submodel parameters, as shown in the example in Fig. 7.2. However, note in the same figure that the achievable

total communication costs have a fluctuating structure based on whether the values of K and R result in odd or even $R-K$ in (7.38). For any (K,R) MDS code with even $R-K$, the total cost results in a *local peak*. Therefore, the total communication cost can be decreased further if for a given set of storage constraints $\{\mu(n)\}_{n=1}^{N}$, the model parameters can be stored as a linear combination of (K,R) MDS codes with only odd $R-K$ instead of the four codes considered in Table 7.1. This eliminates the involvement of the *local peaks*. We begin the analysis of such cases with the following lemma.

Lemma 7.4 *Let $(\boldsymbol{\mu_1}, C_T(\boldsymbol{\mu_1}))$ and $(\boldsymbol{\mu_2}, C_T(\boldsymbol{\mu_2}))$ be two pairs of storage constraints and the corresponding achievable total costs. The storage constraints are given by $\boldsymbol{\mu_1} = \{\mu_1(n)\}_{n=1}^{N}$ and $\boldsymbol{\mu_2} = \{\mu_2(n)\}_{n=1}^{N}$. Then, the pair $(\boldsymbol{\mu}, C_T(\boldsymbol{\mu}))$ is also achievable for any $\gamma \in [0,1]$ where $\boldsymbol{\mu} = \{\mu(n)\}_{n=1}^{N}$ with,*

$$\mu(n) = \gamma\mu_1(n) + (1-\gamma)\mu_2(n), \quad n \in \{1,\ldots,N\} \tag{7.79}$$

$$C_T(\boldsymbol{\mu}) = \gamma C_T(\boldsymbol{\mu_1}) + (1-\gamma)C_T(\boldsymbol{\mu_2}) \tag{7.80}$$

Proof: Since $(\boldsymbol{\mu_1}, C_T(\boldsymbol{\mu_1}))$ and $(\boldsymbol{\mu_2}, C_T(\boldsymbol{\mu_2}))$ are achievable, let S_1 and S_2 be the schemes that produce the achievable pairs $(\boldsymbol{\mu_1}, C_T(\boldsymbol{\mu_1}))$ and $(\boldsymbol{\mu_2}, C_T(\boldsymbol{\mu_2}))$, respectively. A new scheme can be generated by applying S_1 on a γ fraction of bits of all submodels and S_2 on the rest of the bits. The storage capacity of database n in this combined scheme is given by $\gamma ML\mu_1(n) + (1-\gamma)ML\mu_2(n) = \mu(n)ML$ bits. The

corresponding total cost is,

$$C_T = \frac{\gamma L C_T(\boldsymbol{\mu_1}) + (1-\gamma) L C_T(\boldsymbol{\mu_2})}{L} = \gamma C_T(\boldsymbol{\mu_1}) + (1-\gamma) C_T(\boldsymbol{\mu_2}). \tag{7.81}$$

completing the proof. ■

Corollary 7.1 *Consider a PRUW setting with an arbitrary set of storage constraints* $\boldsymbol{\mu} = \{\mu(n)\}_{n=1}^{N}$. *Based on Lemma 7.4, a* γ *fraction of all submodel parameters can be stored using any* (K_1, R_1) *MDS code, and the rest of the* $1-\gamma$ *fraction can be stored using any* (K_2, R_2) *MDS code to achieve a total cost of* $\gamma C_T(K_1, R_1) + (1-\gamma) C_T(K_2, R_2)$ *if there exist two sets of storage constraints* $\boldsymbol{\mu_1} = \{\mu_1(n)\}_{n=1}^{N}$ *and* $\boldsymbol{\mu_2} = \{\mu_2(n)\}_{n=1}^{N}$ *that only use* (K_1, R_1) *and* (K_2, R_2) *MDS codes, respectively, to store the submodel parameters. For this,* $\boldsymbol{\mu_1}$ *and* $\boldsymbol{\mu_2}$ *must satisfy the following conditions.*

$$\gamma \mu_1(n) + (1-\gamma)\mu_2(n) = \mu(n), \quad n \in \{1, \ldots, N\} \tag{7.82}$$

$$\sum_{n=1}^{N} \mu_1(n) = \frac{R_1}{K_1} \tag{7.83}$$

$$\max_n \mu_1(n) \leq \frac{1}{K_1} \tag{7.84}$$

$$\sum_{n=1}^{N} \mu_2(n) = \frac{R_2}{K_2} \tag{7.85}$$

$$\max_n \mu_2(n) \leq \frac{1}{K_2} \tag{7.86}$$

where (7.82) *is straightforward from Lemma 7.4,* (7.83), (7.85) *are based on the fact that a* (K, R) *MDS code combines* K *bits together and stores them at* R *databases,*

resulting in a total of $\frac{ML}{K} \times R$ *bits stored across all databases, and* (7.84), (7.86) *are required by* (7.42).

The method proposed in Section 7.4.2 cannot be used to find $\boldsymbol{\mu_1} = \{\mu_1(n)\}_{n=1}^N$ and $\boldsymbol{\mu_2} = \{\mu_2(n)\}_{n=1}^N$ in (7.82)-(7.86) for any given set of arbitrary storage constraints because of (7.82) and limitations on R_1, R_2, K_1, K_2. In this section, we discuss a specific type of storage constraints $\boldsymbol{\mu} = \{\mu(n)\}_{n=1}^N$ for which a direct solution to (7.82)-(7.86) is available. For this type of storage constraints, the *local peaks* in the achievable costs curves can be eliminated, which results in reduced total communication costs compared to what can be achieved from the scheme in Section 7.4.2 for the same storage constraints.

Consider the case of homogeneous storage constraints, i.e., $\mu(n) = \mu$ for $n \in \{1, \ldots, N\}$. This is the specific type of storage constraints we consider in this section with a direct solution to (7.82)-(7.86). Note that for this case, (7.82), (7.83) and (7.85) imply,

$$\mu = \gamma \frac{R_1}{NK_1} + (1-\gamma)\frac{R_2}{NK_2}. \tag{7.87}$$

Therefore, for any μ such that $\frac{R_1}{NK_1} \leq \mu \leq \frac{R_2}{NK_2}$,[4] one solution to (7.82)-(7.86) is given by,

$$\mu_1(n) = \frac{R_1}{NK_1}, \quad n \in \{1, \ldots, N\} \tag{7.88}$$

$$\mu_2(n) = \frac{R_2}{NK_2}, \quad n \in \{1, \ldots, N\}, \tag{7.89}$$

[4]Without loss of generality, we assume that $\frac{R_1}{K_1} \leq \frac{R_2}{K_2}$.

as $N \geq R_1, R_2$ must be satisfied by any (K_i, R_i) MDS coded storage in a system of N databases. The value of γ for the given μ is determined by (7.87).

The above solution basically implies that when a given homogeneous storage constraint μ is in the range $\frac{R_1}{NK_1} \leq \mu \leq \frac{R_2}{NK_2}$ for any R_1, R_2, K_1, K_2, a total cost of $\gamma C_T(K_1, R_1) + (1-\gamma)C_T(K_2, R_2)$ is achievable, where $\gamma = \frac{N\mu - \frac{R_2}{K_2}}{\frac{R_1}{K_1} - \frac{R_2}{K_2}}$. This further implies that all points on the line connecting $(\frac{R_1}{NK_1}, C_T(\frac{R_1}{NK_1}))$ and $(\frac{R_2}{NK_2}, C_T(\frac{R_2}{NK_2}))$ for any R_1, R_2, K_1, K_2 with $\frac{R_1}{K_1} \leq \frac{R_2}{K_2}$ are achievable. This allows for any point on the line connecting the adjacent local minima in the total costs curves to be achievable, which eliminates the *local peaks*.

Next, we present the general storage scheme for homogeneous storage constraints based on the above arguments.

7.5 Proposed Scheme: Homogeneous Storage Constraints

In this section, we present the general scheme for arbitrary homogeneous storage constraints denoted by $\mu(n) = \mu$, $n \in \{1, \ldots, N\}$. The basic idea of this scheme is to find all achievable pairs of the form $(\mu, C_T(\mu))$,[5], find its lower convex hull, and apply Lemma 7.4 for a given μ with the closest points on the convex hull to obtain the minimum achievable cost with the PRUW scheme presented in Section 7.4.1. The detailed storage scheme is given next.

For a given N we first find the achievable pairs of $(\mu, C_T(\mu))$ as follows. Let $\mu = \frac{R}{NK_R}$ for $R = 4, \ldots, N$ and $K_R = 1, \ldots, R-3$. For a given μ with a given R and K_R, the following steps need to be followed in order to perform PRUW while

[5] $\boldsymbol{\mu}$ in Lemma 7.4 is replaced by μ, as all storage constraints are equal.

meeting the storage constraint:

1. Divide the L bits of each submodel into N sections and label them as $\{1, \ldots, N\}$.

2. Allocate sections $n : (n - 1 + R) \bmod N$ to database n for $n \in \{1, \ldots, N\}$.[6]

3. Use the storage specified in (7.17) with $K = K_R$ and x, y given in (7.37) to encode each of the allocated sections of all submodels. Note that a given coded bit of a given section of each submodel stored across different databases contains the same noise polynomial that only differs in α_n.

4. Use the PRUW scheme described in Section 7.4.1 on each of the subsets of $n : (n-1+R) \bmod N$ databases to read/write to section $(n-1+R) \mod N$ of the required submodel for $n \in \{1, \ldots, N\}$.

For each $\mu = \frac{R}{NK_R}$, $R = 4, \ldots, N$, $K_R = 1, \ldots, R-3$, the above process encodes the submodel parameters using a (K_R, R) MDS code, and gives an achievable $(\mu, C_T(\mu))$ pair, where $C_T(\mu)$ is given by

$$C_T(\mu) = \begin{cases} \frac{4R}{R-K_R-1}, & \text{if } R - K_R \text{ is odd} \\ \\ \frac{4R-2}{R-K_R-2}, & \text{if } R - K_R \text{ is even.} \end{cases} \tag{7.90}$$

Note that the above two cases, which correspond to the value of $(R - K_R) \bmod 2$, are a result of two different schemes. The case with odd values of $R - K_R$ has a subpacketization that is equal to the degree of noise polynomial in storage, which

[6]The indices here follow a cyclic pattern, i.e., if $(n - 1 + R) \bmod N < n$, then $n : (n - 1 + R) \bmod N$ implies $\{n, \ldots, N, 1, \ldots, (n - 1 + R) \bmod N\}$.

does not require the null shaper, while the case with even values of $R-K_R$ contains two more noise terms than the subpacketization, which requires the null shaper; see (7.37). The scheme corresponding to even values of $R-K_R$ is inefficient compared to the even case due to the additional noise term present in storage. This observation combined with Lemma 7.4 results in the following lemma, which formally states how the *local peaks* in achievable costs can be eliminated.

Lemma 7.5 *For a given $\mu = \frac{R}{NK_R}$, if R and K_R are such that $R-K_R$ is even, it is more efficient to perform a linear combination of two PRUW schemes with nearest two odd $R^{[i]} - K_R^{[i]}$, $i = 1, 2$, instead of performing direct PRUW with the given R and K_R, while satisfying the same storage constraint μ, i.e., with $\mu_1 = \frac{R^{[1]}}{NK_R^{[1]}}$ and $\mu_2 = \frac{R^{[2]}}{NK_R^{[2]}}$.*

Proof: For a given $\mu = \frac{R}{NK_R}$ with even $R-K_R$, the nearest $\mu_1 = \frac{R^{[1]}}{NK_R^{[1]}}$ is $\frac{R-1}{NK_R}$, since (7.39) with K replaced by K_R needs to be satisfied for the PRUW scheme to work. Similarly, $\mu_2 = \frac{R+1}{NK_R}$. Let $C_T(\mu)$, $C_T(\mu_1)$ and $C_T(\mu_2)$ be the total costs incurred by the scheme with μ, μ_1 and μ_2, respectively. From (7.90), we have

$$C_T(\mu) = \frac{4R-2}{R-K_R-2} > C_T(\mu_1) = \frac{4R-4}{R-K_R-2} > C_T(\mu_2) = \frac{4(R+1)}{R-K_R}. \tag{7.91}$$

Note that $\mu_1 < \mu < \mu_2$. From Lemma 7.4, there exists some $\gamma \in [0, 1]$ that allocates the storage for the two PRUW schemes corresponding to μ_1 and μ_2 that achieves the same storage constraint as μ, and results in a total cost of $\gamma C_T(\mu_1) + (1-\gamma)C_T(\mu_2)$,

that satisfies

$$C_T(\mu_2) < \gamma C_T(\mu_1) + (1-\gamma)C_T(\mu_2) < C_T(\mu_1) < C_T(\mu), \tag{7.92}$$

completing the proof. ■

Once the basic $(\mu, C_T(\mu))$ pairs corresponding to $\mu = \frac{R}{NK_R}$ for $R = 4, \ldots, N$, $K_R = 1, \ldots, R-3$ with odd $R - K_R$ are obtained, the minimum achievable total cost of the improved scheme for any μ is characterized by the lower convex hull of the above basic $(\mu, C_T(\mu))$ pairs, denoted by T_{ach}. This is straightforward from Lemmas 7.4 and 7.5. Therefore, for a given N and μ, if $\mu_1 = \frac{R_1}{NK_1}$ and $\mu_2 = \frac{R_2}{NK_2}$ are the nearest storage constraints to μ such that $\mu_1 \leq \mu \leq \mu_2$, with $(\mu_1, C_T(\mu_1))$ and $(\mu_1, C_T(\mu_1))$ being elements of the set of basic pairs that determine T_{ach}, a total cost of $\gamma C_T(K_1, R_1) + (1-\gamma)C_T(K_2, R_2)$ with $\gamma = \frac{\mu_2 - \mu}{\mu_2 - \mu_1}$ is achievable by storing γ fraction of all submodels using a (K_1, R_1) MDS code, and the rest of the $1-\gamma$ fractions of all submodels using a (K_2, R_2) MDS code. Once the storage is determined, the PRUW scheme presented in Section 7.4.1 is used to perform the private FSL.

7.5.1 Comparison with Other Schemes

The two straightforward methods to handle homogeneous storage constraints is to consider divided storage and coded storage. Divided storage, i.e., $K_R = 1$, $R < N$ is where the submodel parameters are uncoded, but divided and stored at subsets of databases to meet the storage constraints. Coded storage, i.e., $K_R > 1$, $R = N$ is where the submodel parameters are encoded to combine multiple symbols into a

$K_R \backslash R$	4	6	8	10
1	0.4	0.6	0.8	1
3		0.2	0.27	0.33
5			0.16	0.2
7				0.14

$K_R \backslash R$	5	7	9
2	0.25	0.35	0.45
4		0.175	0.225
6			0.15

uncoded-divided storage

coded-undivided storage

points on lower convex hull

Figure 7.3: All possible pairs of (R, K_R) and corresponding values of μ for $N = 10$.

single symbol and stored at all N databases. Note that both divided and coded storage mechanisms are subsets of the proposed storage mechanism which considers all cases that correspond to $K_R \geq 1$ and $R \leq N$. In other words, the proposed scheme can be viewed as a hybrid mechanism of both divided and coded schemes. The hybrid scheme achieves lower total costs compared to divided and coded schemes, as it considers a larger set of basic $(\mu, C_T(\mu))$ pairs to find the lower convex hull, which includes all points considered in divided and coded schemes individually. As an illustration, consider the example with $N = 10$ databases. The proposed hybrid storage mechanism first determines the basic achievable $\left(\mu = \frac{R}{NK_R}, C_T(\mu)\right)$ pairs for $R = 4, \ldots, N$, $K_R = 1, \ldots, R-3$ with odd $R - K_R$. The pairs of (R, K_R) and the corresponding values of μ are shown in Fig. 7.3.

The set of basic achievable $\left(\mu = \frac{r}{NK_r}, C_T(\mu)\right)$ pairs on the lower convex hull corresponds to storage constraints μ with (R, K_R) pairs corresponding to $R = N = 10$, $R = N - 1 = 9$ and $R = N - 2 = 8$ with K_R values that satisfy $(R - K_R - 1)$

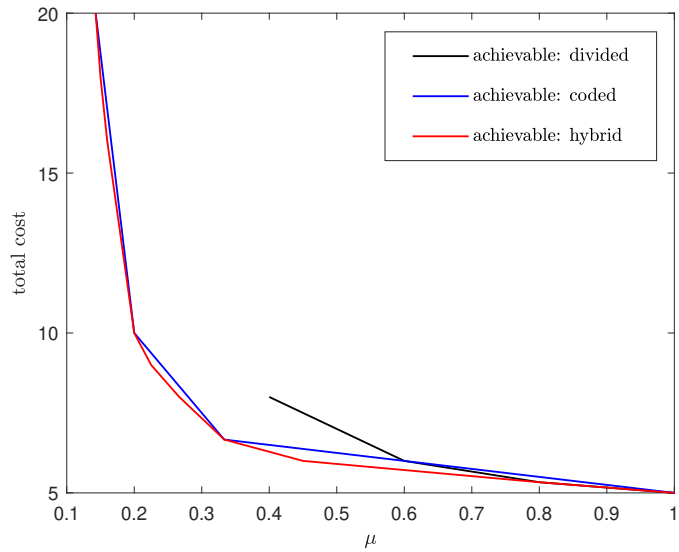


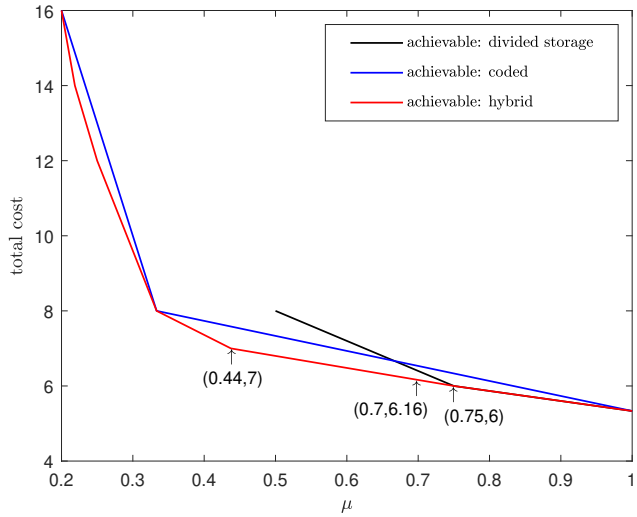
Figure 7.4: Lowest achievable costs of coded, divided and hybrid schemes for $N = 10$. mod $2 = 0$, as marked in green in Fig. 7.3. Note that the minimum achievable costs of divided and coded schemes are determined by the lower convex hull of the points marked in blue and red in Fig. 7.3, respectively, which are subsets of all points considered in the lower convex hull search of the hybrid scheme, which clearly results in lower achievable costs as shown in Fig. 7.4.

7.5.2 Example

In this section, we describe how the PRUW process is carried out in an arbitrary setting with given N and μ. Consider an example with $N = 8$ databases and $\mu = 0.7$. The first step is to find the basic achievable $\left(\mu = \frac{R}{NK_R}, C_T(\mu)\right)$ pairs of $N = 8$ that lie on the lower convex hull boundary. Fig. 7.5(a) shows the (R, K_R) pairs and the corresponding μs of such pairs. The required storage constraint $\mu = 0.7$ is in between 0.44 and 0.75, which correspond to (R, K_R) pairs $(7, 2)$ and $(6, 1)$, respectively. Therefore, the PRUW scheme for $N = 8$, $\mu = 0.7$ is obtained by the

K_R \ R	6	7	8
1	0.75		1
2		0.44	
3	0.25		0.33
4		0.22	
5			0.2

(a) $\mu = \frac{r}{NK_r}$ on the boundary.



(b) Achievable total cost at $\mu = 0.7$.

Figure 7.5: Example with $N = 8$.

following steps:

1. γL bits of all submodels are stored according to the proposed storage mechanism corresponding to $(R, K_R) = (7, 2)$, and the rest of the $(1-\gamma)L$ bits of all submodels are stored according to $(R, K_R) = (6, 1)$. Therefore, γL bits of the required submodel are updated using the scheme corresponding to $(R, K_R) = (7, 2)$, and the rest of the bits are updated by the scheme corresponding to $(6, 1)$. In order to find the value of γ, we equate the total storage of each database to the given constraint, i.e.,

$$\gamma ML \times \frac{7}{8} \times \frac{1}{2} + (1-\gamma)ML \times \frac{6}{8} = 0.7ML \tag{7.93}$$

   which gives $\gamma = 0.16$.

2. Let $L_1 = 0.16L$ and $L_2 = 0.84L$. L_1 bits of each submodel is divided into

8 sections and labeled $1, \ldots, 8$. Sections $n : (n+6) \bmod 8$ are allocated to database n for $n \in \{1, \ldots, N\}$. Each database uses the storage in (7.17) with $K = 2$ and $y = x = \frac{R-K_R-1}{2} = 2$ to store each subpacket of all sections allocated to it. Then, the PRUW scheme described in Section 7.4.1 is applied to read/write to the L_1 bits of the required submodel.

3. The same process is carried out on the rest of the L_2 bits with the scheme corresponding to $(6, 1)$.

The total costs incurred by the two schemes are $C_{T_1} = \frac{4R}{R-K_R-1} = \frac{4\times 7}{7-2-1} = 7$ and $C_{T_2} = \frac{4R}{R-K_R-1} = \frac{4\times 6}{6-1-1} = 6$, respectively. Therefore, the total cost of $N = 8$ and $\mu = 0.7$ is $C_T = \frac{\gamma L C_{T_1} + (1-\gamma) L C_{T_2}}{L} = 6.16$, which is shown in Fig. 7.5(b).

7.6 Conclusions

In this work, we considered the problem of information-theoretically private FSL with storage constrained databases. We considered both heterogeneous and homogeneous storage constraints, and proposed schemes to perform private FSL in both settings, with the goal of minimizing the total communication cost while guaranteeing the privacy of the updating submodel index and the values of the updates. As the main result, we proposed a PRUW scheme and a storage mechanism that is applicable to any given set of heterogeneous storage constraints, and a different storage mechanism built upon the same PRUW scheme for homogeneous storage constraints. Although the proposed scheme for heterogeneous constraints is not applicable to homogeneous constraints in general, the communication costs achieved

by the latter are lower than what is achieved by the former. This is because the scheme proposed for homogeneous constraints is able to eliminate an inefficient MDS coding structure in its storage, which is not possible in the heterogeneous case if the scheme must serve any given set of heterogeneous storage constraints.

7.7 Appendix

7.7.1 Proof of Lemma 7.1

Proof: When $\alpha = 1$, the only coding parameter used in the entire storage is $\lfloor k \rfloor$, which modifies the maximum number of replications in (7.43) as,

$$R \leq \frac{\sum_{n=1}^{N} \mu(n) ML}{\frac{ML}{\lfloor k \rfloor}} = \lfloor k \rfloor p = s. \tag{7.94}$$

If $s \in \mathbb{Z}^+$, all parameters are encoded with the $(\lfloor k \rfloor, s)$ MDS code. However, if $s \notin \mathbb{Z}^+$ we need to find the optimum fractions β and storage allocations[7] $\hat{\mu}_1(n)$, $\hat{\mu}_2(n)$ of each submodel to be encoded with $(\lfloor k \rfloor, \lfloor s \rfloor)$ and $(\lfloor k \rfloor, \lceil s \rceil)$ MDS codes that minimize the modified total cost in (7.57) given by,

$$C = \beta C_T(\lfloor k \rfloor, \lfloor s \rfloor) + (1 - \beta) C_T(\lfloor k \rfloor, \lceil s \rceil). \tag{7.95}$$

[7] $\hat{\mu}_1(n)$ and $\hat{\mu}_2(n)$ are the storage allocations for $(\lfloor k \rfloor, \lfloor s \rfloor)$ and $(\lfloor k \rfloor, \lceil s \rceil)$ MDS codes in database n.

The total storage allocations in the entire system of databases for the two MDS codes must satisfy (modified (7.48) and (7.49)),

$$\sum_{n=1}^{N} \hat{\mu}_1(n) = \frac{\beta}{\lfloor k \rfloor} \lfloor s \rfloor \tag{7.96}$$

$$\sum_{n=1}^{N} \hat{\mu}_2(n) = \frac{1-\beta}{\lfloor k \rfloor} \lceil s \rceil. \tag{7.97}$$

Moreover, since

$$\sum_{n=1}^{N} \hat{\mu}_1(n) + \sum_{n=1}^{N} \hat{\mu}_2(n) = \sum_{n=1}^{N} \mu(n) = p = \frac{\lceil s \rceil - \beta}{\lfloor k \rfloor}, \tag{7.98}$$

β is fixed at $\beta = \lceil s \rceil - s$, and the resulting total cost is given by (7.7). Note that the total cost in (7.95) does not depend on each individual $\hat{\mu}_1(n)$ and $\hat{\mu}_2(n)$. Therefore, any set of $\hat{\mu}_1(n)$ and $\hat{\mu}_2(n)$ satisfying (7.96) and (7.97) with $\beta = \lceil s \rceil - s$ along with the following constraints (modified (7.52) and (7.53)) results in the same total cost. Modified (7.52) and (7.53) are given by,

$$\hat{\mu}_1(n) \leq \frac{\lceil s \rceil - s}{\lfloor k \rfloor} \tag{7.99}$$

$$\hat{\mu}_2(n) \leq \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor}. \tag{7.100}$$

Therefore, it remains to prove that $\hat{\mu}_1(n)$ and $\hat{\mu}_1(n)$ for each n provided in Lemma 7.1 satisfy (7.96)-(7.100) when $\beta = \lceil s \rceil - s$. Note that (7.58) results in,

$$\sum_{n=1}^{N} \hat{\mu}_1(n) = \sum_{n=1}^{N} \tilde{m}(n) + \tilde{\gamma}(p - \sum_{n=1}^{N} \tilde{m}(n) - \sum_{n=1}^{N} \tilde{h}(n)) = \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s), \tag{7.101}$$

which proves (7.96). Similarly, summing up (7.59) results in (7.97). Assuming that $0 \leq \tilde{\gamma} \leq 1$, (7.58) gives,

$$\hat{\mu}_1(n) \leq \mu(n) - \tilde{h}(n) \tag{7.102}$$

$$\leq \begin{cases} \frac{\lceil s \rceil - s}{\lfloor k \rfloor}, & \text{if } \mu(n) \geq \frac{\lceil s \rceil - s}{\lfloor k \rfloor} \\ \mu(n), & \text{if } \mu(n) < \frac{\lceil s \rceil - s}{\lfloor k \rfloor}, \end{cases} \tag{7.103}$$

which proves (7.99). A similar proof is valid for (7.100) as well, given that $0 \leq \tilde{\gamma} \leq 1$, which is proved next. Let $\mathcal{N}$ and $\mathcal{D}$ denote the numerator and the denominator of $\tilde{\gamma}$ in (7.61). Note that $\tilde{m}(n) + \tilde{h}(n)$ is given by,

$$\tilde{m}(n) + \tilde{h}(n) = \begin{cases} \mu(n) - \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor}, & \text{if } \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor} \leq \mu(n) \leq \frac{\lceil s \rceil - s}{\lfloor k \rfloor} \\ \mu(n) - \frac{\lceil s \rceil - s}{\lfloor k \rfloor}, & \text{if } \frac{\lceil s \rceil - s}{\lfloor k \rfloor} \leq \mu(n) \leq \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor} \\ 2\mu(n) - \frac{1}{\lfloor k \rfloor}, & \text{if } \frac{\lceil s \rceil - s}{\lfloor k \rfloor}, \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor} \leq \mu(n) \\ 0, & \text{if } \frac{\lceil s \rceil - s}{\lfloor k \rfloor}, \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor} \geq \mu(n) \end{cases} \leq \mu(n), \tag{7.104}$$

since $\max_n \mu(n) \leq \frac{1}{k} \leq \frac{1}{\lfloor k \rfloor}$, and the equality holds only if $\mu(n) = \frac{1}{\lfloor k \rfloor}$. Therefore,

$$\sum_{n=1}^{N} \tilde{m}(n) + \sum_{n=1}^{N} \tilde{h}(n) \leq p, \tag{7.105}$$

and the equality holds only if $\mu(n) = \frac{1}{\lfloor k \rfloor}$, $\forall n$, which specifies a set of homogeneous storage constraints. Since we only consider heterogeneous storage constraints, we have, $\sum_{n=1}^{N} \tilde{m}(n) + \sum_{n=1}^{N} \tilde{h}(n) < p$, which proves $\mathcal{D} > 0$. Therefore, it remains to

prove that $\mathcal{N} \leq \mathcal{D}$ and $\mathcal{N} \geq 0$, to prove $0 \leq \tilde{\gamma} \leq 1$. Note that $\mathcal{N} \leq \mathcal{D}$ is equivalent to $\frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s) \leq p - \sum_{n=1}^{N} \tilde{h}(n)$. Let V be defined as,

$$V = \sum_{n=1}^{N} \mathbf{1}_{\{\mu(n) \geq \frac{\lceil s \rceil - s}{\lfloor k \rfloor}\}}. \tag{7.106}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function. From the definition of $\tilde{h}(n)$ in (7.60) and since $\mu(n) \leq \frac{1}{\lfloor k \rfloor}$,

$$\sum_{n=1}^{N} \tilde{h}(n) \leq \frac{V}{\lfloor k \rfloor} - \frac{V(\lceil s \rceil - s)}{\lfloor k \rfloor}. \tag{7.107}$$

Moreover, since the total available space in the V databases must not exceed p,

$$\sum_{n=1}^{N} \tilde{h}(n) + \frac{V(\lceil s \rceil - s)}{\lfloor k \rfloor} \leq p \quad \Longrightarrow \quad \sum_{n=1}^{N} \tilde{h}(n) \leq p - \frac{V(\lceil s \rceil - s)}{\lfloor k \rfloor} \tag{7.108}$$

must be satisfied. Note that the upper bound on $\sum_{n=1}^{N} \tilde{h}(n)$ in (7.107) is tighter than that of (7.108) when $V \leq \lfloor k \rfloor p = s$ and vice versa. Therefore, the highest upper bound on $\sum_{n=1}^{N} \tilde{h}(n)$ is given by,

$$\sum_{n=1}^{N} \tilde{h}(n) \leq \begin{cases} \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(s - \lfloor s \rfloor), & \text{if } V < s \\ p - \frac{\lceil s \rceil}{\lfloor k \rfloor}(\lceil s \rceil - s), & \text{if } V > s \end{cases}, \tag{7.109}$$

since $V \in \mathbb{Z}^+$ and $s \notin \mathbb{Z}^+$,[8] which proves,

$$p - \sum_{n=1}^{N} \tilde{h}(n) \geq \begin{cases} p - \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(s - \lfloor s \rfloor), & \text{if } V < s \\ \frac{\lceil s \rceil}{\lfloor k \rfloor}(\lceil s \rceil - s), & \text{if } V > s \end{cases} \tag{7.110}$$

$$\geq \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s) \quad \Longrightarrow \quad \mathcal{N} \leq \mathcal{D}. \tag{7.111}$$

To prove $\mathcal{N} \geq 0$, we need to show that $\sum_{n=1}^{N} \tilde{m}(n) \leq \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s)$. Let Y be defined as,

$$Y = \sum_{n=1}^{N} \mathbf{1}_{\{\mu(n) \geq \frac{s - \lfloor s \rfloor}{\lfloor k \rfloor}\}}. \tag{7.112}$$

Then, similar to (7.107) and (7.108), we have,

$$\sum_{n=1}^{N} \tilde{m}(n) \leq \frac{Y}{\lfloor k \rfloor} - \frac{Y(s - \lfloor s \rfloor)}{\lfloor k \rfloor} \tag{7.113}$$

$$\sum_{n=1}^{N} \tilde{m}(n) \leq p - \frac{Y(s - \lfloor s \rfloor)}{\lfloor k \rfloor}, \tag{7.114}$$

and (7.113) provides a tighter upper bound on $\sum_{n=1}^{N} \tilde{m}(n)$ compared to (7.114) when $Y \leq \lfloor k \rfloor p = s$, and vice versa. Therefore, the highest upper bound on $\sum_{n=1}^{N} \tilde{m}(n)$ is given by,

$$\sum_{n=1}^{N} \tilde{m}(n) \leq \begin{cases} \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s), & \text{if } Y < s \\ p - \frac{\lceil s \rceil}{\lfloor k \rfloor}(s - \lfloor s \rfloor), & \text{if } Y > s \end{cases} \tag{7.115}$$

[8] If $s \in \mathbb{Z}^+$ all parameters in all submodels will be $(\lfloor k \rfloor, s)$ MDS coded, which does not require any fractions/storage allocations to be calculated.

$$= \frac{\lfloor s \rfloor}{\lfloor k \rfloor}(\lceil s \rceil - s) \quad \Longrightarrow \quad \mathcal{N} \geq 0 \tag{7.116}$$

since $Y \in \mathbb{Z}^+$ and $s \notin \mathbb{Z}^+$, which completes the proof of $0 \leq \tilde{\gamma} \leq 1$.

■

7.7.2 Proof of Lemma 7.2

Here, we prove that the storage allocations provided in Lemma 7.2 satisfy (7.48)-(7.56) for the case where $\alpha < 1$. The storage allocations $\hat{\mu}_1(n)$, $\hat{\mu}_2(n)$, $\bar{\mu}_1(n)$ and $\bar{\mu}_2(n)$ correspond to MDS codes $(\lfloor k \rfloor, \lfloor r \rfloor)$, $(\lfloor k \rfloor, \lceil r \rceil)$, $(\lceil k \rceil, \lfloor r \rfloor)$ and $(\lceil k \rceil, \lceil r \rceil)$, respectively. We first determine the storage allocations corresponding to the two coding parameters $\lfloor k \rfloor$ and $\lceil k \rceil$, i.e., the total storage allocations for the two pairs of MDS codes $\{(\lfloor k \rfloor, \lfloor r \rfloor), (\lfloor k \rfloor, \lceil r \rceil)\}$ and $\{(\lceil k \rceil, \lfloor r \rfloor), (\lceil k \rceil, \lceil r \rceil)\}$ given by,

$$\hat{\mu}(n) = \hat{\mu}_1(n) + \hat{\mu}_2(n), \quad n \in \{1, \ldots, N\} \tag{7.117}$$

$$\bar{\mu}(n) = \bar{\mu}_1(n) + \bar{\mu}_2(n), \quad n \in \{1, \ldots, N\}. \tag{7.118}$$

Then, the constraints (7.48)-(7.56) impose the following constraints on $\hat{\mu}(n)$ and $\bar{\mu}(n)$,

$$\sum_{n=1}^{N} \hat{\mu}(n) = \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) \tag{7.119}$$

$$\sum_{n=1}^{N} \bar{\mu}(n) = \frac{1-\alpha}{\lceil k \rceil}(\lceil r \rceil - \delta) \tag{7.120}$$

$$\hat{\mu}(n) \leq \frac{\alpha}{\lfloor k \rfloor}, \quad n \in \{1, \ldots, N\} \tag{7.121}$$

$$\bar{\mu}(n) \leq \frac{1-\alpha}{\lceil k \rceil}, \quad n \in \{1, \ldots, N\} \tag{7.122}$$

$$\hat{\mu}(n) + \bar{\mu}(n) = \mu(n), \quad \forall n. \tag{7.123}$$

The combination of (7.119)-(7.123) gives,

$$\mu(n) - \frac{1-\alpha}{\lceil k \rceil} \leq \hat{\mu}(n) \leq \frac{\alpha}{\lfloor k \rfloor}, \quad n \in \{1, \ldots, N\} \tag{7.124}$$

$$\mu(n) - \frac{\alpha}{\lfloor k \rfloor} \leq \bar{\mu}(n) \leq \frac{1-\alpha}{\lceil k \rceil}, \quad n \in \{1, \ldots, N\} \tag{7.125}$$

$$\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) + \frac{1-\alpha}{\lceil k \rceil}(\lceil r \rceil - \delta) = p. \tag{7.126}$$

Based on (7.124) and (7.125), for each $n \in \{1, \ldots, N\}$ define,

$$m(n) = \left[\mu(n) - \frac{1-\alpha}{\lceil k \rceil}\right]^+, \quad h(n) = \left[\mu(n) - \frac{\alpha}{\lfloor k \rfloor}\right]^+. \tag{7.127}$$

Then, the total storage allocations corresponding to the two coding parameters $\lfloor k \rfloor$ and $\lceil k \rceil$ in database n, $n \in \{1, \ldots, N\}$, are chosen as,

$$\hat{\mu}(n) = m(n) + (\mu(n) - m(n) - h(n))\gamma \tag{7.128}$$

$$\bar{\mu}(n) = h(n) + (\mu(n) - m(n) - h(n))(1-\gamma), \tag{7.129}$$

where

$$\gamma = \frac{\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} m(n)}{p - \sum_{n=1}^{N} m(n) - \sum_{n=1}^{N} h(n)}. \tag{7.130}$$

Claim 1: For each $n \in \{1, \ldots, N\}$, $\hat{\mu}(n)$ and $\bar{\mu}(n)$, in (7.128) and (7.129) (same as (7.66) and (7.67)) satisfy (7.119)-(7.123) for those α, β, δ stated in Lemma 7.2.

Proof: Summing each $\hat{\mu}(n)$ term in (7.128) yields,

$$\sum_{n=1}^{N} \hat{\mu}(n) = \sum_{n=1}^{N} m(n) + \gamma(p - \sum_{n=1}^{N} m(n) - \sum_{n=1}^{N} h(n)) = \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta), \tag{7.131}$$

from (7.130), which proves (7.119). A similar proof results in (7.120). Assuming that $0 \leq \gamma \leq 1$, (7.128) can be upper bounded by,

$$\hat{\mu}(n) \leq \mu(n) - h(n) \tag{7.132}$$

$$= \begin{cases} \frac{\alpha}{\lfloor k \rfloor}, & \text{if } \mu(n) \geq \frac{\alpha}{\lfloor k \rfloor} \\ \mu(n), & \text{if } \mu(n) < \frac{\alpha}{\lfloor k \rfloor} \end{cases} \tag{7.133}$$

$$\leq \frac{\alpha}{\lfloor k \rfloor}, \tag{7.134}$$

which proves (7.121). Similarly, (7.122) is proven by considering $\bar{\mu}(n)$. (7.123) is obvious from (7.128) and (7.129). Hence, it remains to prove that $0 \leq \gamma \leq 1$. Similar to the proof of $0 \leq \tilde{\gamma} \leq 1$ in Lemma 7.1, let $\mathcal{N}$ and $\mathcal{D}$ denote the numerator

and the denominator of γ in (7.130). Then, $m(n) + h(n)$ is given by,

$$m(n) + h(n) = \begin{cases} \mu(n) - \frac{\alpha}{\lfloor k \rfloor}, & \text{if } \frac{\alpha}{\lfloor k \rfloor} \leq \mu(n) \leq \frac{1-\alpha}{\lceil k \rceil} \\ \mu(n) - \frac{1-\alpha}{\lceil k \rceil}, & \text{if } \frac{1-\alpha}{\lceil k \rceil} \leq \mu(n) \leq \frac{\alpha}{\lfloor k \rfloor} \\ 2\mu(n) - \frac{\alpha}{\lfloor k \rfloor} - \frac{1-\alpha}{\lceil k \rceil}, & \text{if } \frac{\alpha}{\lfloor k \rfloor}, \frac{1-\alpha}{\lceil k \rceil} \leq \mu(n) \\ 0, & \text{if } \frac{\alpha}{\lfloor k \rfloor}, \frac{1-\alpha}{\lceil k \rceil} \geq \mu(n) \end{cases} \tag{7.135}$$

$$\leq \mu(n), \tag{7.136}$$

if $\mu(n) \leq \frac{\alpha}{\lfloor k \rfloor} + \frac{1-\alpha}{\lceil k \rceil}$, $\forall n$, i.e., $\alpha \geq \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k)$, which is the constraint on α stated in Lemma 7.2. Therefore,

$$\sum_{n=1}^{N} m(n) + \sum_{n=1}^{N} h(n) \leq p, \tag{7.137}$$

and the equality holds only if $\mu(n) = \frac{1}{k}$, $\forall n$ and $\alpha = \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k)$, which specifies a set of homogeneous storage constraints. Since we only consider heterogeneous storage constraints, (7.137) is satisfied with strict inequality when α is chosen such that $\frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) \leq \alpha < 1$, which proves $\mathcal{D} > 0$. Therefore, it remains to prove that $\mathcal{N} \leq \mathcal{D}$ and $\mathcal{N} \geq 0$, to prove $0 \leq \gamma \leq 1$. Note that $\mathcal{N} \leq \mathcal{D}$ is equivalent to $\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) \leq p - \sum_{n=1}^{N} h(n)$. Let V be defined as,

$$V = \sum_{n=1}^{N} \mathbf{1}_{\{\mu(n) \geq \frac{\alpha}{\lfloor k \rfloor}\}}. \tag{7.138}$$

Therefore from the definition of $h(n)$ in (7.127) and since $\mu(n) \leq \frac{1}{k}$, $\forall n$,

$$\sum_{n=1}^{N} h(n) \leq \frac{V}{k} - \frac{V\alpha}{\lfloor k \rfloor}. \tag{7.139}$$

Moreover, since the total available space in the V databases must not exceed p,

$$\sum_{n=1}^{N} h(n) + \frac{V\alpha}{\lfloor k \rfloor} \leq p \quad \Longrightarrow \quad \sum_{n=1}^{N} h(n) \leq p - \frac{V\alpha}{\lfloor k \rfloor} \tag{7.140}$$

must be satisfied. The upper bound on $\sum_{n=1}^{N} h(n)$ in (7.139) is tighter than that of (7.140) when $V \leq kp = r$ and vice versa. Thus, the highest upper bound on $\sum_{n=1}^{N} h(n)$ is given by,

$$\sum_{n=1}^{N} h(n) \leq \begin{cases} \frac{\lfloor r \rfloor}{k} - \frac{\lfloor r \rfloor \alpha}{\lfloor k \rfloor}, & \text{if } V < r \\ p - \frac{\lceil r \rceil \alpha}{\lfloor k \rfloor}, & \text{if } V > r \end{cases}, \tag{7.141}$$

since $V \in \mathbb{Z}^{+}$, which gives the tightest upper bound on $p - \sum_{n=1}^{N} h(n)$ for a general set of $\{\mu(n)\}_{n=1}^{N}$ as,

$$p - \sum_{n=1}^{N} h(n) \geq \begin{cases} p - \frac{\lfloor r \rfloor}{k} + \frac{\lfloor r \rfloor \alpha}{\lfloor k \rfloor}, & \text{if } V < r \\ \frac{\lceil r \rceil \alpha}{\lfloor k \rfloor}, & \text{if } V > r \end{cases}. \tag{7.142}$$

Therefore, in order to satisfy $\mathcal{N} \leq \mathcal{D}$ for any given set of $\{\mu(n)\}_{n=1}^{N}$, i.e.,

$$p - \sum_{n=1}^{N} h(n) \geq \begin{cases} p - \frac{\lfloor r \rfloor}{k} + \frac{\lfloor r \rfloor \alpha}{\lfloor k \rfloor}, & \text{if } V < r \\ \frac{\lceil r \rceil \alpha}{\lfloor k \rfloor}, & \text{if } V > r \end{cases} \tag{7.143}$$

$$\geq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) \tag{7.144}$$

it requires β to satisfy $\beta \geq 1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor)$, which is the constraint given on β in Lemma 7.2. To prove $\mathcal{N} \geq 0$, we need to show that $\sum_{n=1}^{N} m(n) \leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta)$. Let Y be defined as,

$$Y = \sum_{n=1}^{N} \mathbf{1}_{\{\mu(n) \geq \frac{1-\alpha}{\lceil k \rceil}\}}. \tag{7.145}$$

Then, similar to (7.139) and (7.140), we have,

$$\sum_{n=1}^{N} m(n) \leq \frac{Y}{k} - \frac{Y(1-\alpha)}{\lceil k \rceil} \tag{7.146}$$

$$\sum_{n=1}^{N} m(n) \leq p - \frac{Y(1-\alpha)}{\lceil k \rceil}, \tag{7.147}$$

and (7.146) provides a tighter upper bound on $\sum_{n=1}^{N} m(n)$ compared to (7.147) when $Y \leq kp = r$, and vice versa. Therefore, the highest upper bound on $\sum_{n=1}^{N} m(n)$ considering any arbitrary set of $\{\mu(n)\}_{n=1}^{N}$ is given by,

$$\sum_{n=1}^{N} m(n) \leq \begin{cases} \frac{\lfloor r \rfloor}{k} - \frac{\lfloor r \rfloor (1-\alpha)}{\lceil k \rceil}, & \text{if } Y < r \\ p - \frac{\lceil r \rceil (1-\alpha)}{\lceil k \rceil}, & \text{if } Y > r \end{cases} \tag{7.148}$$

since $Y \in \mathbb{Z}^+$. Therefore, $\mathcal{N} \geq 0$ is satisfied for any set of $\{\mu(n)\}_{n=1}^N$ if

$$\sum_{n=1}^{N} m(n) \leq \begin{cases} \frac{\lfloor r \rfloor}{k} - \frac{\lfloor r \rfloor (1-\alpha)}{\lceil k \rceil}, & \text{if } Y < r \\ p - \frac{\lceil r \rceil (1-\alpha)}{\lceil k \rceil}, & \text{if } Y > r \end{cases} \tag{7.149}$$

$$\leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) \tag{7.150}$$

is satisfied, which requires $\delta \geq 1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor)$. This is the constraint stated in Lemma 7.2, which is derived using (7.126). Therefore, $0 \leq \gamma \leq 1$ is satisfied by any given set of arbitrary heterogeneous storage constraints $\{\mu(n)\}_{n=1}^N$ when

$$1 > \alpha \geq \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) \tag{7.151}$$

$$1 \geq \beta \geq \left[1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor)\right]^+ \tag{7.152}$$

$$1 \geq \delta \geq \left[1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor)\right]^+ . \tag{7.153}$$

which completes the proof of Claim 1. ■

The above proof finalizes the storage allocations corresponding to coding parameters $\lfloor k \rfloor$ and $\lceil k \rceil$. Next, we find the storage allocations corresponding to each of the two MDS codes relevant to each coding parameter, i.e., $(\lfloor k \rfloor, \lfloor r \rfloor)$ and $(\lfloor k \rfloor, \lceil r \rceil)$ corresponding to $\lfloor k \rfloor$ and $(\lceil k \rceil, \lfloor r \rfloor)$ and $(\lceil k \rceil, \lceil r \rceil)$ corresponding to $\lceil k \rceil$. In other words, we further divide $\hat{\mu}(n)$ into $\hat{\mu}_1(n), \hat{\mu}_2(n)$ and $\bar{\mu}(n)$ into $\bar{\mu}_1(n), \bar{\mu}_2(n)$ such that $\hat{\mu}_1(n), \hat{\mu}_2(n), \bar{\mu}_1(n), \bar{\mu}_2(n)$ satisfy (7.48)-(7.56). Note that the constraints (7.52)-

(7.55) result in,

$$\hat{\mu}(n) - \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \leq \hat{\mu}_1(n) \leq \frac{\alpha\beta}{\lfloor k \rfloor} \tag{7.154}$$

$$\hat{\mu}(n) - \frac{\alpha\beta}{\lfloor k \rfloor} \leq \hat{\mu}_2(n) \leq \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \tag{7.155}$$

$$\bar{\mu}(n) - \frac{(1-\alpha)(1-\delta)}{\lceil k \rceil} \leq \bar{\mu}_1(n) \leq \frac{(1-\alpha)\delta}{\lceil k \rceil} \tag{7.156}$$

$$\bar{\mu}(n) - \frac{(1-\alpha)\delta}{\lceil k \rceil} \leq \bar{\mu}_2(n) \leq \frac{(1-\alpha)(1-\delta)}{\lceil k \rceil}, \tag{7.157}$$

for each $n \in \{1, \ldots, N\}$. Based on (7.154)-(7.157), for each $n \in \{1, \ldots, N\}$ define

$$\hat{m}(n) = \left[\hat{\mu}(n) - \frac{\alpha(1-\beta)}{\lfloor k \rfloor}\right]^+, \qquad \hat{h}(n) = \left[\hat{\mu}(n) - \frac{\alpha\beta}{\lfloor k \rfloor}\right]^+ \tag{7.158}$$

$$\bar{m}(n) = \left[\bar{\mu}(n) - \frac{(1-\alpha)(1-\delta)}{\lceil k \rceil}\right]^+, \quad \bar{h}(n) = \left[\bar{\mu}(n) - \frac{(1-\alpha)\delta}{\lceil k \rceil}\right]^+. \tag{7.159}$$

Then, define the storage allocations in database n, $n \in \{1, \ldots, N\}$ corresponding to MDS codes $(\lfloor k \rfloor, \lfloor r \rfloor)$, $(\lfloor k \rfloor, \lceil r \rceil)$, $(\lceil k \rceil, \lfloor r \rfloor)$ and $(\lceil k \rceil, \lceil r \rceil)$ as

$$\hat{\mu}_1(n) = \begin{cases} \hat{\mu}(n)\beta, & \text{if } \beta \in \{0,1\} \\ \hat{m}(n) + (\hat{\mu}(n) - \hat{m}(n) - \hat{h}(n))\hat{\gamma}, & \text{if } \beta \in (0,1) \end{cases} \tag{7.160}$$

$$\hat{\mu}_2(n) = \begin{cases} \hat{\mu}(n)(1-\beta), & \text{if } \beta \in \{0,1\} \\ \hat{h}(n) + (\hat{\mu}(n) - \hat{m}(n) - \hat{h}(n))(1-\hat{\gamma}), & \text{if } \beta \in (0,1) \end{cases} \tag{7.161}$$

$$\bar{\mu}_1(n) = \begin{cases} \bar{\mu}(n)\delta, & \text{if } \delta \in \{0,1\} \\ \bar{m}(n) + (\bar{\mu}(n) - \bar{m}(n) - \bar{h}(n))\bar{\gamma}, & \text{if } \delta \in (0,1) \end{cases} \tag{7.162}$$

$$\bar{\mu}_2(n) = \begin{cases} \bar{\mu}(n)(1-\delta), & \text{if } \delta \in \{0,1\} \\ \\ \bar{h}(n) + (\bar{\mu}(n) - \bar{m}(n) - \bar{h}(n))(1-\bar{\gamma}), & \text{if } \delta \in (0,1) \end{cases}, \tag{7.163}$$

respectively, where,

$$\hat{\gamma} = \frac{\frac{\alpha\beta}{\lfloor k \rfloor}\lfloor r \rfloor - \sum_{n=1}^{N} \hat{m}(n)}{\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} \hat{m}(n) - \sum_{n=1}^{N} \hat{h}(n)} \tag{7.164}$$

$$\bar{\gamma} = \frac{\frac{(1-\alpha)\delta}{\lceil k \rceil}\lfloor r \rfloor - \sum_{n=1}^{N} \bar{m}(n)}{\frac{1-\alpha}{\lceil k \rceil}(\lceil r \rceil - \delta) - \sum_{n=1}^{N} \bar{m}(n) - \sum_{n=1}^{N} \bar{h}(n)}. \tag{7.165}$$

Claim 2: For each $n \in \{1, \ldots, N\}$, $\hat{\mu}_1(n)$ and $\hat{\mu}_2(n)$, in (7.160) and (7.161) (same as (7.62) and (7.63)) satisfy (7.48)-(7.49) and (7.52)-(7.53) for those α, β, δ stated in Lemma 7.2.

Proof: *Case 1:* $\beta = 1$: When $\beta = 1$, all parameters that are $\lfloor k \rfloor$ coded are replicated in only $\lfloor r \rfloor$ databases since the fraction of submodels that are $(\lfloor k \rfloor, \lceil r \rceil)$ MDS coded is $\alpha(1-\beta)$ (see Table 7.1). Therefore, $\hat{\mu}_1(n) = \hat{\mu}(n)$ and $\hat{\mu}_2(n) = 0$ for each $n \in \{1, \ldots, N\}$. Then,

$$\sum_{n=1}^{N} \hat{\mu}_1(n) = \sum_{n=1}^{N} \hat{\mu}(n) = \frac{\alpha}{\lfloor k \rfloor}\lfloor r \rfloor, \tag{7.166}$$

from (7.119), which proves (7.48). Moreover, $\hat{\mu}_2(n) = 0$, $\forall n$ proves (7.49). For each $n \in \{1, \ldots, N\}$, (7.121) results in $\hat{\mu}_1(n) = \hat{\mu}(n) \leq \frac{\alpha}{\lfloor k \rfloor}$, which proves (7.52), and $\hat{\mu}_2(n) = 0$, $\forall n$ proves (7.53).

Case 2: $\beta = 0$: The proof contains identical steps to the proof of Case 1.

Case 3: $\beta \in (0,1)$: Summing each $\hat{\mu}_1(n)$ term in (7.160) yields,

$$\sum_{n=1}^{N} \hat{\mu}_1(n) = \sum_{n=1}^{N} \hat{m}(n) + \hat{\gamma}(\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} \hat{m}(n) - \sum_{n=1}^{N} \hat{h}(n)) = \frac{\alpha\beta}{\lceil k \rceil}\lfloor r \rfloor, \quad (7.167)$$

from (7.164), which proves (7.48). A similar proof results in (7.49). Assuming that $0 \leq \gamma \leq 1$, (7.160) can be upper bounded by,

$$\hat{\mu}_1(n) \leq \hat{\mu}(n) - \hat{h}(n) \quad (7.168)$$

$$= \begin{cases} \frac{\alpha\beta}{\lfloor k \rfloor}, & \text{if } \hat{\mu}(n) \geq \frac{\alpha\beta}{\lfloor k \rfloor} \\ \hat{\mu}(n), & \text{if } \mu(n) < \frac{\alpha\beta}{\lfloor k \rfloor} \end{cases} \quad (7.169)$$

$$\leq \frac{\alpha\beta}{\lfloor k \rfloor}, \quad (7.170)$$

which proves (7.52). Similarly, (7.53) is proven by considering $\hat{\mu}_2(n)$. Hence, it remains to prove that $0 \leq \hat{\gamma} \leq 1$. Let $\mathcal{N}$ and $\mathcal{D}$ denote the numerator and the denominator of $\hat{\gamma}$ in (7.164). Then, $\hat{m}(n) + \hat{h}(n)$ is given by,

$$\hat{m}(n) + \hat{h}(n) = \begin{cases} \hat{\mu}(n) - \frac{\alpha(1-\beta)}{\lfloor k \rfloor}, & \text{if } \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \leq \hat{\mu}(n) \leq \frac{\alpha\beta}{\lfloor k \rfloor} \\ \hat{\mu}(n) - \frac{\alpha\beta}{\lfloor k \rfloor}, & \text{if } \frac{\alpha\beta}{\lfloor k \rfloor} \leq \hat{\mu}(n) \leq \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \\ 2\hat{\mu}(n) - \frac{\alpha}{\lfloor k \rfloor}, & \text{if } \frac{\alpha\beta}{\lfloor k \rfloor}, \frac{\alpha(1-\beta)}{\lceil k \rceil} \leq \hat{\mu}(n) \\ 0, & \text{if } \frac{\alpha\beta}{\lfloor k \rfloor}, \frac{\alpha(1-\beta)}{\lfloor k \rfloor} \geq \hat{\mu}(n) \end{cases} \quad (7.171)$$

$$\leq \hat{\mu}(n), \quad (7.172)$$

since $\hat{\mu}(n) \leq \frac{\alpha}{\lfloor k \rfloor}$, $\forall n$, from (7.121). Therefore, from (7.119),

$$\sum_{n=1}^{N} \hat{m}(n) + \sum_{n=1}^{N} \hat{h}(n) \leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta), \tag{7.173}$$

and the equality holds only if $\hat{\mu}(n) = \frac{\alpha}{\lfloor k \rfloor}$, $\forall n$, since the cases where $\beta \in \{0, 1\}$ are already considered. $\hat{\mu}(n) = \frac{\alpha}{\lfloor k \rfloor}$, $\forall n$ specifies a set of homogeneous storage constraints which can be directly stored using the optimum storage mechanism provided in Section 7.5 for homogeneous storage constraints.[9] Note that this case is not in the scope of Claim 2. For all other cases, (7.173) is satisfied with strict inequality, which proves $\mathcal{D} > 0$. Therefore, it remains to prove that $\mathcal{N} \leq \mathcal{D}$ and $\mathcal{N} \geq 0$, to prove $0 \leq \hat{\gamma} \leq 1$. Note that $\mathcal{N} \leq \mathcal{D}$ is equivalent to $\frac{\alpha\beta}{\lfloor k \rfloor}\lfloor r \rfloor \leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} \hat{h}(n)$. Let V be defined as,

$$V = \sum_{n=1}^{N} \mathbf{1}_{\{\hat{\mu}(n) \geq \frac{\alpha\beta}{\lfloor k \rfloor}\}}. \tag{7.174}$$

Therefore, from the definition of $\hat{h}(n)$ in (7.158) and since $\hat{\mu}(n) \leq \frac{\alpha}{\lfloor k \rfloor}$, $\forall n$ from

[9]The purpose of this step in the scheme is to divide the α fraction of all submodels that are $\lfloor k \rfloor$ coded into two MDS codes given by $(\lfloor k \rfloor, \lfloor r \rfloor)$ and $(\lfloor k \rfloor, \lceil r \rceil)$ such that all databases are filled and the coded parameters are replicated in the respective number of databases. This is satisfied by the scheme in Section 7.5 for homogeneous storage constraints as explained next. $\hat{\mu}(n)$ corresponds to the storage allocated in database n for all $\lfloor k \rfloor$ coded parameters. Based on the discussion in Section 7.5, any linear combination of $C_T(\lfloor k \rfloor, \lfloor r \rfloor)$ and $C_T(\lfloor k \rfloor, \lceil r \rceil)$ is achievable by storing β and $1 - \beta$ fractions of all $\lfloor k \rfloor$ coded parameters using $(\lfloor k \rfloor, \lfloor r \rfloor)$ and $(\lfloor k \rfloor, \lceil r \rceil)$ MDS codes, respectively, for any $\beta \in [0, 1]$, given that $\lfloor k \rfloor \leq \lfloor r \rfloor - 4$. Therefore, for the case where $\hat{\mu}(n) = \frac{\alpha}{\lfloor k \rfloor}$, $\forall n$, the α fraction of all submodels that are $\lfloor k \rfloor$ coded in Table 7.1 can be arbitrarily divided (arbitrary β) and encoded with the two MDS codes $(\lfloor k \rfloor, \lfloor r \rfloor)$ and $(\lfloor k \rfloor, \lceil r \rceil)$ to achieve a total cost of $\alpha\beta C_T(\lfloor k \rfloor, \lfloor r \rfloor) + \alpha(1 - \beta)C_T(\lfloor k \rfloor, \lceil r \rceil)$, while filling all databases.

(7.121),

$$\sum_{n=1}^{N} \hat{h}(n) \leq \frac{V\alpha}{\lfloor k \rfloor} - \frac{V\alpha\beta}{\lfloor k \rfloor} = \frac{V\alpha(1-\beta)}{\lfloor k \rfloor}. \tag{7.175}$$

Moreover, since the total available space in the V databases must not exceed $\sum_{n=1}^{N} \hat{\mu}(n) = \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta)$,

$$\sum_{n=1}^{N} \hat{h}(n) + \frac{V\alpha\beta}{\lfloor k \rfloor} \leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) \quad \Longrightarrow \quad \sum_{n=1}^{N} \hat{h}(n) \leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \frac{V\alpha\beta}{\lfloor k \rfloor} \tag{7.176}$$

must be satisfied. The upper bound on $\sum_{n=1}^{N} \hat{h}(n)$ in (7.175) is tighter than that of (7.176) when $V \leq \lceil r \rceil - \beta$ and vice versa. Therefore, the highest upper bound on $\sum_{n=1}^{N} \hat{h}(n)$ is given by,

$$\sum_{n=1}^{N} \hat{h}(n) \leq \begin{cases} \frac{\alpha(1-\beta)}{\lfloor k \rfloor}\lfloor r \rfloor, & \text{if } V \leq \lceil r \rceil - \beta \\ \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \frac{\alpha\beta}{\lfloor k \rfloor}\lceil r \rceil, & \text{if } V > \lceil r \rceil - \beta \end{cases}, \tag{7.177}$$

since $V \in \mathbb{Z}^{+}$, which gives the tightest upper bound on $\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} \hat{h}(n)$ for a general set of $\{\mu(n)\}_{n=1}^{N}$ as,

$$\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \sum_{n=1}^{N} \hat{h}(n) \geq \begin{cases} \frac{\alpha}{\lfloor k \rfloor}(1 - \beta + \beta\lfloor r \rfloor), & \text{if } V \leq \lceil r \rceil - \beta \\ \frac{\alpha\beta}{\lfloor k \rfloor}\lceil r \rceil, & \text{if } V < \lceil r \rceil - \beta \end{cases} \tag{7.178}$$

$$\geq \frac{\alpha\beta}{\lfloor k \rfloor}\lfloor r \rfloor, \tag{7.179}$$

which proves $\mathcal{N} \leq \mathcal{D}$. To prove $\mathcal{N} \geq 0$, we need to show that $\sum_{n=1}^{N} \hat{m}(n) \leq \frac{\alpha\beta}{\lfloor k \rfloor}\lfloor r \rfloor$. Let Y be defined as,

$$Y = \sum_{n=1}^{N} \mathbf{1}_{\{\hat{\mu}(n) \geq \frac{\alpha(1-\beta)}{\lfloor k \rfloor}\}}. \tag{7.180}$$

Then, similar to (7.175) and (7.176), we have

$$\sum_{n=1}^{N} \hat{m}(n) \leq \frac{Y\alpha}{\lfloor k \rfloor} - \frac{Y\alpha(1-\beta)}{\lfloor k \rfloor} \tag{7.181}$$

$$\sum_{n=1}^{N} \hat{m}(n) \leq \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \frac{Y\alpha(1-\beta)}{\lfloor k \rfloor}, \tag{7.182}$$

and (7.181) provides a tighter upper bound on $\sum_{n=1}^{N} \hat{m}(n)$ compared to (7.182) when $Y \leq \lceil r \rceil - \beta$, and vice versa. Therefore, the highest upper bound on $\sum_{n=1}^{N} \hat{m}(n)$ considering any arbitrary set of $\{\mu(n)\}_{n=1}^{N}$ is given by,

$$\sum_{n=1}^{N} \hat{m}(n) \leq \begin{cases} \frac{\lfloor r \rfloor \alpha\beta}{\lfloor k \rfloor}, & \text{if } Y \leq \lceil r \rceil - \beta \\ \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) - \frac{\lceil r \rceil \alpha(1-\beta)}{\lfloor k \rfloor}, & \text{if } Y > \lceil r \rceil - \beta \end{cases} \tag{7.183}$$

$$= \frac{\alpha\beta}{\lfloor k \rfloor}\lfloor r \rfloor \tag{7.184}$$

since $Y \in \mathbb{Z}^{+}$, proving $\mathcal{N} \geq 0$, completing the proof of $0 \leq \hat{\gamma} \leq 1$. This proves Claim 2. ■

Claim 3: For each $n \in \{1, \ldots, N\}$, $\bar{\mu}_1(n)$ and $\bar{\mu}_2(n)$, in (7.162) and (7.163) (same as (7.64) and (7.65)) satisfy (7.50)-(7.51) and (7.54)-(7.55) for those α, β, δ stated in Lemma 7.2.

Proof: The proof of Claim 3 consists of the exact same steps as in the proof of Claim 2. ■

7.7.3 Proof of Lemma 7.3

Proof: Here, we derive the optimum values of α, β, δ that minimize (7.57) while satisfying (7.48)-(7.56) for the case where $\alpha < 1$. Note that the constraints in (7.151)-(7.153) and (7.126) must be satisfied by α, β, δ to guarantee (7.48)-(7.56). The following optimization problem is solved to obtain the optimum fractions:

$$\begin{aligned}
\min \quad & C = \alpha\beta C_T(\lfloor k \rfloor, \lfloor r \rfloor) + \alpha(1-\beta)C_T(\lfloor k \rfloor, \lceil r \rceil) + (1-\alpha)\delta C_T(\lceil k \rceil, \lfloor r \rfloor) \\
& \qquad + (1-\alpha)(1-\delta)C_T(\lceil k \rceil, \lceil r \rceil) && (7.185) \\
\text{s.t.} \quad & \frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) \leq \alpha < 1 && (7.186) \\
& \left[1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor)\right]^+ \leq \beta \leq 1 && (7.187) \\
& \left[1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor)\right]^+ \leq \delta \leq 1 && (7.188) \\
& \frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) + \frac{1-\alpha}{\lceil k \rceil}(\lceil r \rceil - \delta) = p. && (7.189)
\end{aligned}$$

where the total cost is explicitly given by,

$$C = \begin{cases} \frac{4\alpha\beta\lfloor r \rfloor}{\lfloor r \rfloor - \lfloor k \rfloor - 1} + \frac{\alpha(1-\beta)(4\lceil r \rceil - 2)}{\lceil r \rceil - \lfloor k \rfloor - 2} + \frac{(1-\alpha)\delta(4\lfloor r \rfloor - 2)}{\lfloor r \rfloor - \lceil k \rceil - 2} + \frac{4(1-\alpha)(1-\delta)\lceil r \rceil}{\lceil r \rceil - \lceil k \rceil - 1}, & \text{if } \lfloor r \rfloor - \lfloor k \rfloor \text{ is odd} \\ \frac{\alpha\beta(4\lfloor r \rfloor - 2)}{\lfloor r \rfloor - \lfloor k \rfloor - 2} + \frac{4\alpha(1-\beta)\lceil r \rceil}{\lceil r \rceil - \lfloor k \rfloor - 1} + \frac{4(1-\alpha)\delta\lfloor r \rfloor}{\lfloor r \rfloor - \lceil k \rceil - 1} + \frac{(1-\alpha)(1-\delta)(4\lceil r \rceil - 2)}{\lceil r \rceil - \lceil k \rceil - 2}, & \text{if } \lfloor r \rfloor - \lfloor k \rfloor \text{ is even} \end{cases}. \tag{7.190}$$

We consider two cases for the two different total costs, and obtain the KKT conditions:

Case 1: Odd $\lfloor r \rfloor - \lfloor k \rfloor$: The Lagrangian function for this case is given by,[10]

$$
\begin{aligned}
J = C &+ \lambda_1(\alpha - 1) + \lambda_2 \left(\frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) - \alpha \right) + \lambda_3(\beta - 1) + \lambda_4 \left(1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor) - \beta \right) \\
&- \lambda_5 \beta + \lambda_6(\delta - 1) + \lambda_7 \left(1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor) - \delta \right) - \lambda_8 \delta \\
&+ \lambda_9 \left(\frac{\alpha}{\lfloor k \rfloor}(\lceil r \rceil - \beta) + \frac{1-\alpha}{\lceil k \rceil}(\lceil r \rceil - \delta) - p \right). \qquad (7.191)
\end{aligned}
$$

The KKT conditions for this case are,

$$
\begin{aligned}
\frac{\partial J}{\partial \alpha} = &\frac{4\beta \lfloor r \rfloor}{\lfloor r \rfloor - \lfloor k \rfloor - 1} + \frac{(1-\beta)(4\lceil r \rceil - 2)}{\lceil r \rceil - \lfloor k \rfloor - 2} - \frac{\delta(4\lfloor r \rfloor - 2)}{\lfloor r \rfloor - \lceil k \rceil - 2} - \frac{4(1-\delta)\lceil r \rceil}{\lceil r \rceil - \lceil k \rceil - 1} + \lambda_1 - \lambda_2 \\
&+ \lambda_4 \frac{\lfloor k \rfloor}{k\alpha^2}(r - \lfloor r \rfloor) - \lambda_7 \frac{\lceil k \rceil}{k(1-\alpha)^2}(r - \lfloor r \rfloor) + \lambda_9 \left(\frac{\lceil r \rceil - \beta}{\lfloor k \rfloor} - \frac{\lceil r \rceil - \delta}{\lceil k \rceil} \right) = 0
\end{aligned} \qquad (7.192)
$$

$$
\frac{\partial J}{\partial \beta} = \frac{4\alpha \lfloor r \rfloor}{\lfloor r \rfloor - \lfloor k \rfloor - 1} - \frac{\alpha(4\lceil r \rceil - 2)}{\lceil r \rceil - \lfloor k \rfloor - 2} + \lambda_3 - \lambda_4 - \lambda_5 - \lambda_9 \frac{\alpha}{\lfloor k \rfloor} = 0 \qquad (7.193)
$$

$$
\frac{\partial J}{\partial \delta} = \frac{(1-\alpha)(4\lfloor r \rfloor - 2)}{\lfloor r \rfloor - \lceil k \rceil - 2} - \frac{4(1-\alpha)\lceil r \rceil}{\lceil r \rceil - \lceil k \rceil - 1} + \lambda_6 - \lambda_7 - \lambda_8 - \lambda_9 \frac{(1-\alpha)}{\lceil k \rceil} = 0 \qquad (7.194)
$$

$$
\lambda_1(\alpha - 1) = 0, \quad \lambda_2 \left(\frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) - \alpha \right) = 0, \quad \lambda_3(\beta - 1) = 0, \qquad (7.195)
$$

$$
\lambda_4 \left(1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor) - \beta \right) = 0, \quad \lambda_5 \beta = 0, \quad \lambda_6(\delta - 1) = 0, \qquad (7.196)
$$

$$
\lambda_7 \left(1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor) - \delta \right) = 0, \quad \lambda_8 \delta = 0 \qquad (7.197)
$$

$$
\lambda_i \geq 0, \quad i \in \{1, \ldots, 8\}, \qquad (7.198)
$$

[10] We treat the constraint $\alpha < 1$ as $\alpha \leq 1$ in the Lagrangian, and avoid the case $\alpha = 1$ in the analysis.

and (7.186)-(7.189). Solving the above KKT conditions results in the optimum α, β, δ stated in (7.9)-(7.11) in Theorem 7.1, along with the minimum total cost.

Case 2: Even $\lfloor r \rfloor - \lfloor k \rfloor$: The Lagrangian function for this case is the same as (7.191), with the respective total cost C from (7.190). The KKT conditions for this case are,

$$\frac{\partial J}{\partial \alpha} = \frac{\beta(4\lfloor r \rfloor - 2)}{\lfloor r \rfloor - \lfloor k \rfloor - 2} + \frac{4(1-\beta)\lceil r \rceil}{\lceil r \rceil - \lfloor k \rfloor - 1} - \frac{4\delta\lfloor r \rfloor}{\lfloor r \rfloor - \lceil k \rceil - 1} - \frac{(1-\delta)(4\lceil r \rceil - 2)}{\lceil r \rceil - \lceil k \rceil - 2} + \lambda_1 - \lambda_2 + \lambda_4 \frac{\lfloor k \rfloor}{k\alpha^2}(r - \lfloor r \rfloor) - \lambda_7 \frac{\lceil k \rceil}{k(1-\alpha)^2}(r - \lfloor r \rfloor) + \lambda_9 \left(\frac{\lceil r \rceil - \beta}{\lfloor k \rfloor} - \frac{\lceil r \rceil - \delta}{\lceil k \rceil} \right) = 0 \tag{7.199}$$

$$\frac{\partial J}{\partial \beta} = \frac{\alpha(4\lfloor r \rfloor - 2)}{\lfloor r \rfloor - \lfloor k \rfloor - 2} - \frac{4\alpha\lceil r \rceil}{\lceil r \rceil - \lfloor k \rfloor - 1} + \lambda_3 - \lambda_4 - \lambda_5 - \lambda_9 \frac{\alpha}{\lfloor k \rfloor} = 0 \tag{7.200}$$

$$\frac{\partial J}{\partial \delta} = \frac{4(1-\alpha)\lfloor r \rfloor}{\lfloor r \rfloor - \lceil k \rceil - 1} - \frac{(1-\alpha)(4\lceil r \rceil - 2)}{\lceil r \rceil - \lceil k \rceil - 2} + \lambda_6 - \lambda_7 - \lambda_8 - \lambda_9 \frac{(1-\alpha)}{\lceil k \rceil} = 0 \tag{7.201}$$

$$\lambda_1(\alpha - 1) = 0, \quad \lambda_2 \left(\frac{\lfloor k \rfloor}{k}(\lceil k \rceil - k) - \alpha \right) = 0, \quad \lambda_3(\beta - 1) = 0, \tag{7.202}$$

$$\lambda_4 \left(1 - \frac{\lfloor k \rfloor}{k\alpha}(r - \lfloor r \rfloor) - \beta \right) = 0, \quad \lambda_5 \beta = 0, \quad \lambda_6(\delta - 1) = 0, \tag{7.203}$$

$$\lambda_7 \left(1 - \frac{\lceil k \rceil}{k(1-\alpha)}(r - \lfloor r \rfloor) - \delta \right) = 0, \quad \lambda_8 \delta = 0 \tag{7.204}$$

$$\lambda_i \geq 0, \quad i \in \{1, \ldots, 8\}, \tag{7.205}$$

and (7.186)-(7.189). Solving the above KKT conditions results in the optimum α, β, δ stated in (7.12)-(7.14) in Theorem 7.1.

■

CHAPTER 8

Deceptive Information Retrieval (DIR)

8.1 Introduction

In this chapter, we introduce the problem of deceptive information retrieval (DIR), in which a user wishes to download a required file out of multiple independent files stored in a system of databases while *deceiving* the databases by making the databases' predictions on the user-required file index incorrect with high probability. Conceptually, DIR is an extension of private information retrieval (PIR). In PIR, a user downloads a required file without revealing its index to any of the databases. The metric of deception is defined as the probability of error of databases' prediction on the user-required file, minus the corresponding probability of error in PIR. The problem is defined on time-sensitive data that keeps updating from time to time. In the proposed scheme, the user deceives the databases by sending *real* queries to download the required file at the time of the requirement and *dummy* queries at multiple distinct future time instances to manipulate the probabilities of sending each query for each file requirement, using which the databases' make the predictions on the user-required file index. The proposed DIR scheme is based on a capacity

achieving probabilistic PIR scheme, and achieves rates lower than the PIR capacity due to the additional downloads made to deceive the databases. When the required level of deception is zero, the proposed scheme achieves the PIR capacity.

8.2 Problem Formulation and System Model

We consider N non-colluding databases storing K independent files, each consisting of L uniformly distributed symbols from a finite field $\mathbb{F}_q$, i.e.,

$$H(W_1, \ldots, W_K) = \sum_{i=1}^{K} H(W_i) = KL, \tag{8.1}$$

where W_i is the ith file. The files keep updating from time to time, and a given user wants to download an arbitrary file at arbitrary time instances T_i, $i \in \mathbb{N}$. We assume that all files are equally probable to be requested by the user.

The user sends queries at arbitrary time instances to download the required file while *deceiving* the databases. We assume that the databases are only able to store data (files, queries from users, time stamps of received queries etc.) corresponding to the current time instance, and that the file updates at distinct time instances are mutually independent. Therefore, the user's file requirements and the queries sent are independent of the stored files at all time instances, i.e.,

$$I(\theta^{[t]}, Q_n^{[t]}; W_{1:K}^{[t]}) = 0, \quad n \in \{1, \ldots, N\}, \quad \forall t, \tag{8.2}$$

where $\theta^{[t]}$ is the user's file requirement, $Q_n^{[t]}$ is the query sent by the user to database

n, and $W_{1:K}^{[t]}$ is the set of K files, all at time t.[1] At any given time t when each database n, $n \in \{1, \ldots, N\}$, receives a query from the user, it sends the corresponding answer as a function of the received query and the stored files, thus,

$$H(A_n^{[t]} | Q_n^{[t]}, W_{1:K}^{[t]}) = 0, \quad n \in \{1, \ldots, N\}, \tag{8.3}$$

where $A_n^{[t]}$ is the answer received by the user from database n at time t. At each time T_i, $i \in \mathbb{N}$, the user must be able to correctly decode the required file, that is,

$$H(W_{\theta^{[T_i]}} | Q_{1:N}^{[T_i]}, A_{1:N}^{[T_i]}) = 0, \quad i \in \mathbb{N}. \tag{8.4}$$

At any given time t when each database n, $n \in \{1, \ldots, N\}$, receives a query from the user, it makes a prediction on the user-required file index using the maximum aposteriori probability (MAP) estimate as follows,

$$\hat{\theta}_{\tilde{Q}}^{[t]} = \arg\max_i P(\theta^{[t]} = i | Q_n^{[t]} = \tilde{Q}), \quad n \in \{1, \ldots, N\}, \tag{8.5}$$

where $\hat{\theta}_{\tilde{Q}}^{[t]}$ is the predicted user-required file index based on the realization of the received query $\tilde{Q}$ at time t. The probability of error of each database's prediction is defined as,

$$P_e = \mathbb{E}[P(\hat{\theta}_{\tilde{Q}}^{[T_i]} \neq \theta^{[T_i]})], \tag{8.6}$$

[1]The notation $1 : K$ indicates all integers from 1 to K.

where the expectation is taken across all $\tilde{Q}$ and T_i. Note that in PIR, $P(\theta_{\tilde{Q}}^{[t]} = i|Q_n^{[t]} = \tilde{Q}) = P(\theta_{\tilde{Q}}^{[t]} = j|Q_n^{[t]} = \tilde{Q})$ for all $i, j \in \{1, \ldots, N\}$, any $\tilde{Q}^{[t]}$, which results in $P_e^{\text{PIR}} = 1 - \frac{1}{K}$. Based on this information, we define the metric of deception as,

$$D = P_e - \left(1 - \frac{1}{K}\right). \tag{8.7}$$

For PIR, the amount of deception is $D = 0$, and for weakly PIR where some amount of information is leaked on the user-required file index, the amount of deception takes a negative value as the probability of error is smaller than $1 - \frac{1}{K}$. The goal of this work is to generate schemes that meet a given level of deception $D = d > 0$, while minimizing the normalized download cost defined as,

$$D_L = \frac{H(A_{1:N})}{L}, \tag{8.8}$$

where $A_{1:N}$ represents all the answers received by all N databases, corresponding to a single file requirement of the user. The DIR rate is defined as the reciprocal of D_L.

8.3 Main Result

In this section we present the main result of this work, along with some remarks. Consider a system of N non-colluding databases containing K identical files. A user is able to retrieve any file k, while deceiving the databases by leaking information about some other file k' to the databases.

Theorem 8.1 *Consider a system of N non-colluding databases storing K independent files. A required level of deception d, satisfying $0 \leq d < \frac{(K-1)(N-1)}{K(N^K-N)}$ is achievable at a DIR rate,*

$$R = \left(\frac{1 + \left(\frac{N^K-N}{N-1}\right) e^{\epsilon}}{1 + (N^{K-1}-1)e^{\epsilon}} + \left(\frac{N}{N-1}\right)(2u - u(u+1)\alpha) \right)^{-1}, \tag{8.9}$$

where

$$\epsilon = \ln\left(\frac{dKN + (K-1)(N-1)}{dKN + (K-1)(N-1) - dKN^K}\right) \tag{8.10}$$

$$\alpha = \frac{N + (N^K - N)e^{\epsilon}}{(N-1)e^{2\epsilon} + (N^K - N)e^{\epsilon} + 1} \tag{8.11}$$

$$u = \lfloor \frac{1}{\alpha} \rfloor \tag{8.12}$$

Remark 8.1 *For given N and K, $\epsilon \geq 0$ is a one-to-one continuous function of d, the required level of deception, and $\alpha \in (0, 1]$ is a one-to-one continuous function of ϵ. For a given $u \in \mathbb{Z}^+$, there exists a range of values of α, specified by $\frac{1}{u+1} < \alpha \leq \frac{1}{u}$, which corresponds to a unique range of values of ϵ, for which (7.3) is valid. Since $(0, 1] = \cup\{\alpha : \frac{1}{u+1} < \alpha \leq \frac{1}{u}, u \in \mathbb{Z}^+\}$, there exists an achievable rate (as well as an achievable scheme) for any $\epsilon \geq 0$ as well as for any d in the range $0 \leq d < \frac{(K-1)(N-1)}{K(N^K-N)}$.*

Remark 8.2 *When the user specified amount of deception is zero, i.e., $d = 0$, the corresponding values of α and u are $\alpha = 1$ and $u = 1$. The achievable rate for this case is $\frac{1-\frac{1}{N}}{1-\frac{1}{N^K}}$, which is equal to the PIR capacity.*

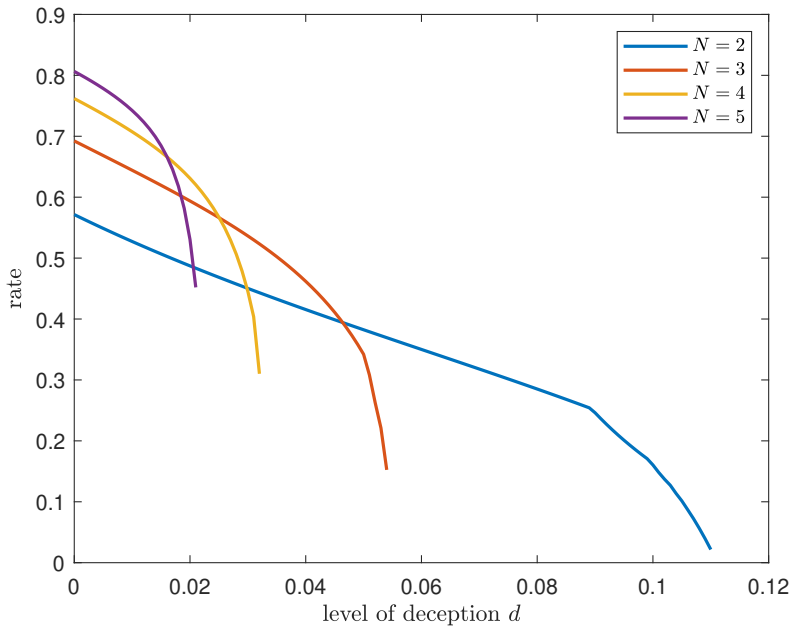


Figure 8.1: Achievable DIR rate for varying levels of deception and different number of databases when $K = 3$.

Remark 8.3 *The achievable DIR rate monotonically decreases with increasing amount of deception d for any given N and K.*

Remark 8.4 *The variation of the achievable DIR rate with the level of deception for different number of databases when the number of files fixed at $K = 3$ is shown in Fig. 8.1. The achievable rate for different number of files when the number of databases is fixed at $N = 2$ is shown in Fig. 8.2. For any given N and K, the rate decreases exponentially when the level of deception is close to the respective upper bound, i.e., $d < \frac{(K-1)(N-1)}{K(N^K-N)}$.*

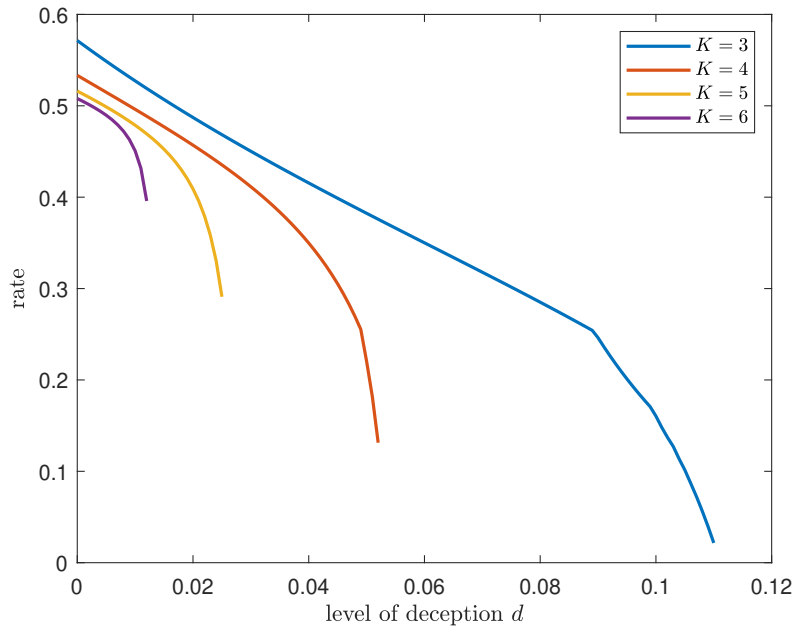


Figure 8.2: Achievable DIR rate for varying levels of deception and different number of files when $N = 2$.

8.4 DIR Scheme

The DIR scheme introduced in this section is designed for a system of N non-colluding databases containing K independent files, with a pre-determined amount of deception $d > 0$ required. For each file requirement at time T_i, $i \in \mathbb{N}$, the user chooses a set of $M + 1$ queries to be sent to database n, $n \in \{1, \ldots, N\}$, at time T_i as well as at future time instances $t_{i,j}$, $j \in \{1, \ldots, M\}$, such that each $t_{i,j} > T_i$. The query sent at time T_i is used to download the required file, while the rest of the M queries are sent to deceive the databases. The queries sent at times T_i, $i \in \mathbb{N}$ and $t_{i,j}$, $j \in \{1, \ldots, M\}$, $i \in \mathbb{N}$ are known as real and dummy queries, respectively. The binary random variable R is used to specify whether a query sent by the user is real or dummy, i.e., $R = 1$ corresponds to a real query sent at time T_i, and

$R = 0$ corresponds to a dummy query sent at time $t_{i,j}$. Next, we define another classification of queries used in the proposed scheme.

Definition 8.1 (ϵ-deceptive query) *An ϵ-deceptive query $\tilde{Q}$ with respect to file k is defined as a query that always satisfies,*

$$\frac{P(Q_n = \tilde{Q}|\theta = k, R = 1)}{P(Q_n = \tilde{Q}|\theta = \ell, R = 1)} = e^{-\epsilon}, \quad \frac{P(\theta = k|Q_n = \tilde{Q})}{P(\theta = \ell|Q_n = \tilde{Q})} = e^{\epsilon}, \quad \forall \ell \in \{1, \ldots, K\},\ \ell \neq k, \tag{8.13}$$

for some $\epsilon > 0$, where Q_n and θ are the random variables representing a query sent to database n, $n \in \{1, \ldots, N\}$, and the user-required file index. An equivalent representation of (8.13) *is given by,*

$$\frac{P(R = 1|\theta = \ell) + \frac{P(Q_n = \tilde{Q}|\theta = \ell, R = 0)}{P(Q_n = \tilde{Q}|\theta = \ell, R = 1)} P(R = 0|\theta = \ell)}{P(R = 1|\theta = k) + \frac{P(Q_n = \tilde{Q}|\theta = k, R = 0)}{P(Q_n = \tilde{Q}|\theta = k, R = 1)} P(R = 0|\theta = k)} = e^{-2\epsilon}, \quad \forall \ell \in \{1, \ldots, K\},\ \ell \neq k. \tag{8.14}$$

Definition 8.2 (PIR query) *A query $\tilde{Q}$ that satisfies* (8.13) *with $\epsilon = 0$ for all $k \in \{1, \ldots, K\}$, i.e., a 0-deceptive query, is known as a PIR query.*

Remark 8.5 *The intuition behind the definition of an ϵ-deceptive query with respect to message k in Definition 8.1 is as follows. Note that the second equation in* (8.13) *fixes the databases' prediction on the user's requirement as W_k for the query $\tilde{Q}$. This is because the aposteriori probability corresponding to message k, when $\tilde{Q}$ is received by the databases, is greater than that of any other message ℓ, $\ell \neq k$. However, the first equation in* (8.13)*, which is satisfied at the same time, ensures that the user*

sends the query $\tilde{Q}$ with the least probability when the user requires to download message k, compared to the probabilities of sending $\tilde{Q}$ for other message requirements. In other words, since we assume equal priors, the query $\tilde{Q}$ is mostly sent when the user requires to download W_ℓ for $\ell \neq k$, and is rarely sent to download W_k, while the databases' prediction on the user-required message upon receiving query $\tilde{Q}$ is fixed at W_k, which is incorrect with high probability, hence, the deception.

At a given time t, there exists a set of queries consisting of both deceptive and PIR queries, sent to the N databases. Database n, $n \in \{1, \ldots, N\}$, is aware of the probability of receiving each query, for each file requirement, i.e., $P(Q_n = \tilde{Q}|\theta = k)$, for $k \in \{1, \ldots, K\}$, $\tilde{Q} \in \mathcal{Q}$, where $\mathcal{Q}$ is the set of all queries. However, the databases are unaware of being deceived, and are unable to determine whether the received query $\tilde{Q}$ is real or dummy or deceptive or PIR. The proposed scheme generates a list of real and dummy queries for a given N and K along with the probabilities of using them as ϵ-deceptive and PIR queries, based on the required level of deception d. The scheme also characterizes the optimum number of dummy queries M to be sent to the databases for each file requirement, to minimize the download cost. As an illustration of the proposed scheme, consider the following representative examples.

8.4.1 Example 1: Two Databases and Two Files, $N = K = 2$

In this example, we present how the proposed DIR scheme is applied in a system of two databases containing two files each. In the proposed scheme, the user generates $M + 1$ queries for any given file-requirement which consists of one real query and

M dummy queries. The user sends the real query at the time of the requirement T_i, and the rest of the M dummy queries at M different future time instances $t_{i,j}$. Tables 8.1 and 8.2 give possible pairs of real queries that are sent to the two databases to retrieve W_1 and W_2, respectively, at time T_i, $i \in \mathbb{N}$. The probability of using each pair of queries is indicated in the first columns of Tables 8.1 and 8.2. Note that the correctness condition in (8.4) is satisfied at each time T_i as each row of Tables 8.1 and 8.2 decodes files W_1 and W_2, respectively, with no error.

$P(Q\|\theta = 1, R = 1)$	DB 1	DB 2
p	W_1	ϕ
p	ϕ	W_1
p'	W_2	$W_1 + W_2$
p'	$W_1 + W_2$	W_2

Table 8.1: Real query table – W_1.

$P(Q\|\theta = 2, R = 1)$	DB 1	DB 2
p	W_2	ϕ
p	ϕ	W_2
p'	W_1	$W_1 + W_2$
p'	$W_1 + W_2$	W_1

Table 8.2: Real query table – W_2.

$P(Q\|\theta = 1, R = 0)$	DB 1	DB 2
1	W_1	W_1

Table 8.3: Dummy query table – W_1.

$P(Q\|\theta = 2, R = 0)$	DB 1	DB 2
1	W_2	W_2

Table 8.4: Dummy query table – W_2.

The dummy queries sent to each database at time $t_{i,j}$ are given in Tables 8.3 and 8.4. The purpose of the dummy queries sent at future time instances is to deceive the databases by manipulating the aposteriori probabilities, which impact their predictions. For example, if the user wants to download W_1 at time T_i, the user selects one of the four query options in Table 8.1 based on the probabilities in column 1,[2] and sends the corresponding queries to database 1 and 2 at time T_i. Based on the information in Table 8.3, the user sends the query W_1 to both databases at M distinct future time instances $t_{i,j}$, $j \in \{1, \ldots, M\}$.

[2]The values of p and p' are derived later in this section.

Based on the information in Tables 8.1-8.4, when the user-required file is W_1, the probability of each query being received by database n, $n \in \{1, 2\}$, at an arbitrary time instance t is calculated as follows. Let $P(R = 1|\theta = i) = \alpha$ for $i \in \{1, 2\}$.[3] Then,

$$P(Q_n = W_1|\theta = 1) = P(Q_n = W_1|\theta = 1, R = 1)P(R = 1|\theta = 1)$$

$$+ P(Q_n = W_1|\theta = 1, R = 0)P(R = 0|\theta = 1) \quad (8.15)$$

$$= p\alpha + 1 - \alpha \quad (8.16)$$

$$P(Q_n = W_2|\theta = 1) = P(Q_n = W_2|\theta = 1, R = 1)P(R = 1|\theta = 1)$$

$$+ P(Q_n = W_2|\theta = 1, R = 0)P(R = 0|\theta = 1) \quad (8.17)$$

$$= p'\alpha \quad (8.18)$$

$$P(Q_n = W_1 + W_2|\theta = 1) = P(Q_n = W_1 + W_2|\theta = 1, R = 1)P(R = 1|\theta = 1)$$

$$+ P(Q_n = W_1 + W_2|\theta = 1, R = 0)P(R = 0|\theta = 1)$$

$$(8.19)$$

$$= p'\alpha \quad (8.20)$$

$$P(Q_n = \phi|\theta = 1) = P(Q_n = \phi|\theta = 1, R = 1)P(R = 1|\theta = 1)$$

$$+ P(Q_n = \phi|\theta = 1, R = 0)P(R = 0|\theta = 1) \quad (8.21)$$

$$= p\alpha \quad (8.22)$$

[3]The intuition behind $P(R = 1|\theta = i)$ is the probability of a query received by any database being real when the user-required file index is i. For a fixed M, $P(R = 1|\theta = i) = \frac{1}{M+1}$.

Thus, writing these probabilities compactly, we have,

$$P(Q_n = W_1|\theta = 1) = p\alpha + 1 - \alpha \tag{8.23}$$

$$P(Q_n = W_2|\theta = 1) = p'\alpha \tag{8.24}$$

$$P(Q_n = W_1 + W_2|\theta = 1) = p'\alpha \tag{8.25}$$

$$P(Q_n = \phi|\theta = 1) = p\alpha. \tag{8.26}$$

Similarly, when the user-required file is W_2, the corresponding probabilities are,

$$P(Q_n = W_1|\theta = 2) = p'\alpha \tag{8.27}$$

$$P(Q_n = W_2|\theta = 2) = p\alpha + 1 - \alpha \tag{8.28}$$

$$P(Q_n = W_1 + W_2|\theta = 2) = p'\alpha \tag{8.29}$$

$$P(Q_n = \phi|\theta = 2) = p\alpha. \tag{8.30}$$

These queries and the corresponding probabilities of sending them to each database for each message requirement are known to the databases. However, the decomposition of these probabilities based on whether the query is real or dummy, i.e., Tables 8.1-8.4, is not known by the databases. When database n, $n \in \{1, \ldots, N\}$, receives a query $\tilde{Q}$ at time t, it calculates the aposteriori probability distribution of the user-required file index, to predict the user's requirement using (8.5). The aposteriori probabilities corresponding to the four queries received by database n,

$n \in \{1, 2\}$, are calculated as follows,

$$P(\theta = i | Q_n = \tilde{Q}) = \frac{P(Q_n = \tilde{Q} | \theta = i) P(\theta = i)}{P(Q_n = \tilde{Q})}. \tag{8.31}$$

Then, the explicit a posteriori probabilities are given by,

$$P(\theta = 1 | Q_n = W_1) = \frac{\frac{1}{2}(p\alpha + 1 - \alpha)}{P(Q_n = W_1)} \tag{8.32}$$

$$P(\theta = 2 | Q_n = W_1) = \frac{\frac{1}{2}p'\alpha}{P(Q_n = W_1)} \tag{8.33}$$

$$P(\theta = 1 | Q_n = W_2) = \frac{\frac{1}{2}p'\alpha}{P(Q_n = W_2)} \tag{8.34}$$

$$P(\theta = 2 | Q_n = W_2) = \frac{\frac{1}{2}(p\alpha + 1 - \alpha)}{P(Q_n = W_2)} \tag{8.35}$$

$$P(\theta = 1 | Q_n = W_1 + W_2) = \frac{\frac{1}{2}p'\alpha}{P(Q_n = W_1 + W_2)} \tag{8.36}$$

$$P(\theta = 2 | Q_n = W_1 + W_2) = \frac{\frac{1}{2}p'\alpha}{P(Q_n = W_1 + W_2)} \tag{8.37}$$

$$P(\theta = 1 | Q_n = \phi) = \frac{\frac{1}{2}p\alpha}{P(Q_n = \phi)} \tag{8.38}$$

$$P(\theta = 2 | Q_n = \phi) = \frac{\frac{1}{2}p\alpha}{P(Q_n = \phi)}. \tag{8.39}$$

While queries ϕ and $W_1 + W_2$ are PIR queries as stated in Definition 8.2, queries W_1 and W_2 are ϵ-deceptive with respect to file indices 1 and 2, respectively, for an ϵ that depends on the required amount of deception d. The values of p and p' in Tables 8.1-8.4 are calculated based on the requirements in Definition 8.1 as follows. It is straightforward to see that $p' = pe^{\epsilon}$ follows from the first part of (8.13) for each query $\tilde{Q} = W_1$ and $\tilde{Q} = W_2$, which also gives $p = \frac{1}{2(1+e^{\epsilon})}$. The second part of (8.13) (as well as (8.14)) results in $\alpha = \frac{2}{1+e^{\epsilon}}$ for both ϵ-deceptive queries W_1 and

W_2. Based on the aposteriori probabilities (8.32)-(8.39) calculated by the databases using the information in (8.23)-(8.30), each database predicts the user's requirement at each time it receives a query from the user. The predictions corresponding to each query received by database n, $n = 1, 2$, which are computed using (8.5), are shown in Table 8.5.

query $\tilde{Q}$	$P(\hat{\theta}_{\tilde{Q}} = 1)$	$P(\hat{\theta}_{\tilde{Q}} = 2)$
W_1	1	0
W_2	0	1
$W_1 + W_2$	$\frac{1}{2}$	$\frac{1}{2}$
ϕ	$\frac{1}{2}$	$\frac{1}{2}$

Table 8.5: Probabilities of each database predicting the user-required file in Example 1.

Based on this information, when a database receives query $Q = W_1$, it always decides that the requested message is W_1, and when it receives query $Q = W_2$, it always decides that the requested message is W_2. For queries $Q = \phi$ and $Q = W_1 + W_2$, the databases flip a coin to choose either W_1 or W_2 as the requested message.

As the queries are symmetric across all databases, the probability of error corresponding to some query $\tilde{Q}$ received by database n at time T_i is given by,

$$
\begin{aligned}
&P(\hat{\theta}_{\tilde{Q}}^{[T_i]} \neq \theta^{[T_i]}) \\
&= P(\theta^{[T_i]} = 1, \hat{\theta}_{\tilde{Q}}^{[T_i]} = 2 | Q_n^{[T_i]} = \tilde{Q}) + P(\theta^{[T_i]} = 2, \hat{\theta}_{\tilde{Q}}^{[T_i]} = 1 | Q_n^{[T_i]} = \tilde{Q}) \qquad (8.40) \\
&= \frac{1}{P(Q_n^{[T_i]} = \tilde{Q})} \Big(P(\hat{\theta}_{\tilde{Q}}^{[T_i]} = 2 | \theta^{[T_i]} = 1, Q_n^{[T_i]} = \tilde{Q}) P(Q_n^{[T_i]} = \tilde{Q} | \theta^{[T_i]} = 1) P(\theta^{[T_i]} = 1)
\end{aligned}
$$

$$+P(\hat{\theta}^{[T_i]}_{\tilde{Q}} = 1|\theta^{[T_i]} = 2, Q_n^{[T_i]} = \tilde{Q})P(Q_n^{[T_i]} = \tilde{Q}|\theta^{[T_i]} = 2)P(\theta^{[T_i]} = 2)\Big) \tag{8.41}$$

$$= \frac{1}{P(Q_n^{[T_i]} = \tilde{Q})}\Big(P(\hat{\theta}^{[T_i]}_{\tilde{Q}} = 2|Q_n^{[T_i]} = \tilde{Q})P(Q_n^{[T_i]} = \tilde{Q}|\theta^{[T_i]} = 1)P(\theta^{[T_i]} = 1)$$

$$+P(\hat{\theta}_{\tilde{Q}} = 1|Q_n^{[T_i]} = \tilde{Q})P(Q_n^{[T_i]} = \tilde{Q}|\theta^{[T_i]} = 2)P(\theta^{[T_i]} = 2)\Big), \tag{8.42}$$

as the predictions only depend on the received queries. The explicit probabilities corresponding to the four queries are,[4]

$$P(\hat{\theta}^{[T_i]}_{W_1} \neq \theta^{[T_i]}) = \frac{1}{P(Q_n^{[T_i]} = W_1)}\frac{e^\epsilon}{4(1+e^\epsilon)} \tag{8.43}$$

$$P(\hat{\theta}^{[T_i]}_{W_2} \neq \theta^{[T_i]}) = \frac{1}{P(Q_n^{[T_i]} = W_2)}\frac{e^\epsilon}{4(1+e^\epsilon)} \tag{8.44}$$

$$P(\hat{\theta}^{[T_i]}_{W_1+W_2} \neq \theta^{[T_i]}) = \frac{1}{P(Q_n^{[T_i]} = W_1+W_2)}\frac{e^\epsilon}{4(1+e^\epsilon)} \tag{8.45}$$

$$P(\hat{\theta}^{[T_i]}_{\phi} \neq \theta^{[T_i]}) = \frac{1}{P(Q_n^{[T_i]} = \phi)}\frac{1}{4(1+e^\epsilon)}. \tag{8.46}$$

As the same scheme is used for all user-requirements at all time instances, the probability of error of each database's prediction for this example is calculated using (8.6) as,

$$P_e = \sum_{\tilde{Q}\in\mathcal{Q}} P(Q_n^{[T_i]} = \tilde{Q})P(\hat{\theta}^{[T_i]}_{\tilde{Q}} \neq \theta^{[T_i]}) \tag{8.47}$$

$$= \frac{3e^\epsilon + 1}{4(1+e^\epsilon)} \tag{8.48}$$

[4]Note that $P(Q_n = \tilde{Q}|\theta^{[T_i]} = i)$ implies $P(Q_n = \tilde{Q}|\theta = i, R = 1)$ as only real queries are sent at time T_i.

where $\mathcal{Q} = \{W_1, W_2, W_1 + W_2, \phi\}$, which results in a deception of $D = \frac{3e^\epsilon+1}{4(1+e^\epsilon)} - \frac{1}{2} = \frac{e^\epsilon-1}{4(1+e^\epsilon)}$. Therefore, for a required amount of deception $d < \frac{1}{4}$, the value of ϵ is chosen as $\epsilon = \ln\left(\frac{4d+1}{1-4d}\right)$.

The download cost of this scheme is computed as follows. As the scheme is symmetric across all file retrievals, and since the apriori probability distribution of the files is uniform, without loss of generality, we can calculate the download cost of retrieving W_1. The download cost of retrieving W_1 for a user specified amount of deception d is given by,

$$D_L = \frac{1}{L}\left(2Lp + 2(2L)pe^\epsilon + 2L\sum_{m=0}^{\infty} p_m m\right) \tag{8.49}$$

$$= \frac{1+2e^\epsilon}{1+e^\epsilon} + 2\mathbb{E}[M] \tag{8.50}$$

where p_m is the probability of sending m dummy queries per each file requirement. To minimize the download cost, we need to find the probability mass function (PMF) of M which minimizes $\mathbb{E}[M]$ such that $P(R = 1|\theta = i) = \alpha = \frac{2}{1+e^\epsilon}$ is satisfied for any i. Note that for any i, $P(R = 1|\theta = i)$ can be written as,

$$P(R = 1|\theta = i) = \alpha = \sum_{m=0}^{\infty} p_m \frac{1}{m+1} = \mathbb{E}\left[\frac{1}{M+1}\right], \tag{8.51}$$

where M is the random variable representing the number of dummy queries sent to each database per file requirement. Thus, the following optimization problem needs

to be solved, for a given ϵ, that is a function of the given value of d,

$$\begin{aligned} \min \quad & \mathbb{E}[M] \\ \text{s.t.} \quad & \mathbb{E}\left[\frac{1}{M+1}\right] = \frac{2}{1+e^{\epsilon}} = \alpha. \end{aligned} \tag{8.52}$$

The solution to this problem is given in Lemma 8.1, and the resulting minimum download cost is given by,

$$D_L = \frac{1+2e^{\epsilon}}{1+e^{\epsilon}} + 4u - 2u(u+1)\alpha, \tag{8.53}$$

where $u = \lfloor \frac{1}{\alpha} \rfloor$. When $d = 0$, it follows that $\epsilon = 0$ and $u = 1$, and the achievable rate is $\frac{2}{3}$, which is the same as the PIR capacity for $N = 2$ and $K = 2$.

8.4.2 Example 2: Three Databases and Three Files, $N = K = 3$

Similar to the previous example, the user sends real queries at time T_i and dummy queries at times $t_{i,j}$, $j \in \{1, \ldots, M\}$, for each $i \in \mathbb{N}$, based on the probabilities shown in Tables 8.7-8.12. The notation W_i^j in these tables correspond to the jth segment of W_i, where each file W_i is divided into $N - 1 = 2$ segments of equal size. Database n, $n \in \{1, \ldots, N\}$, only knows the overall probabilities of receiving each query for each file requirement of the user shown in Table 8.6. These overall probabilities which are calculated using,

$$P(Q_n = \tilde{Q}|\theta = k) = P(Q_n = \tilde{Q}|\theta = k, R = 1)P(R = 1|\theta = k)$$

$$+ P(Q_n = \tilde{Q}|\theta = k, R = 0)P(R = 0|\theta = k), \quad k \in \{1, \ldots, K\} \tag{8.54}$$

where $P(R = 1|\theta = i) = \alpha$ for any $i = 1, 2, 3$, are the same for each database as the scheme is symmetric across all databases.

query $\tilde{Q}$	$P(Q_n = \tilde{Q}\|\theta = 1)$	$P(Q_n = \tilde{Q}\|\theta = 2)$	$P(Q_n = \tilde{Q}\|\theta = 3)$
ϕ	$p\alpha$	$p\alpha$	$p\alpha$
W_1^1	$p\alpha + \frac{1}{2}(1-\alpha)$	$p'\alpha$	$p'\alpha$
W_1^2	$p\alpha + \frac{1}{2}(1-\alpha)$	$p'\alpha$	$p'\alpha$
W_2^1	$p'\alpha$	$p\alpha + \frac{1}{2}(1-\alpha)$	$p'\alpha$
W_2^2	$p'\alpha$	$p\alpha + \frac{1}{2}(1-\alpha)$	$p'\alpha$
W_3^1	$p'\alpha$	$p'\alpha$	$p\alpha + \frac{1}{2}(1-\alpha)$
W_3^2	$p'\alpha$	$p'\alpha$	$p\alpha + \frac{1}{2}(1-\alpha)$
other queries	$p'\alpha$	$p'\alpha$	$p'\alpha$

Table 8.6: Queries received by database n, $n \in \{1, \ldots, N\}$, at a given time t for each file requirement, and the corresponding probabilities.

The entry "other queries" in Table 8.6 includes all queries that have sums of two or three elements. Based on this available information, each database calculates the aposteriori probability of the user-required file index conditioned on each received query $\tilde{Q}$ using (8.31). Each query of the form W_k^j is an ϵ-deceptive query with respect to file k, where ϵ is a function of the required amount of deception, which is derived towards the end of this section. All other queries including the null query and all sums of two or three elements are PIR queries. As all ϵ-deceptive queries must satisfy (8.13), the value of p' is given by $p' = pe^{\epsilon}$, which results in $p = \frac{1}{3(1+8e^{\epsilon})}$, based on the same arguments used in the previous example. Using (8.13) and (8.31) for any given deceptive query, the value of α is calculated as follows. Note that for a query of the form W_k^j, for each database n, $n \in \{1, \ldots, N\}$,

using $P(\theta = k) = \frac{1}{K}$, we have

$$\frac{P(\theta = k|Q_n = W_k^j)}{P(\theta = \ell|Q = W_k^j)} = \frac{P(Q_n = W_k^j|\theta = k)}{P(Q_n = W_k^j|\theta = \ell)} = \frac{p\alpha + \frac{1}{2}(1-\alpha)}{p'\alpha}, \tag{8.55}$$

The value of α is computed as $\alpha = \frac{1}{2p(e^{2\epsilon}-1)+1}$, using (8.55) and (8.13) by solving $\frac{p\alpha+\frac{1}{2}(1-\alpha)}{p'\alpha} = e^{\epsilon}$.

$P(Q\|\theta = 1, R = 1)$	Database 1	Database 2	Database 3
p	W_1^1	W_1^2	ϕ
p	W_1^2	ϕ	W_1^1
p	ϕ	W_1^1	W_1^2
p'	$W_1^1 + W_2^1$	$W_1^2 + W_2^1$	W_2^1
p'	$W_1^2 + W_2^1$	W_2^1	$W_1^1 + W_2^1$
p'	W_2^1	$W_1^1 + W_2^1$	$W_1^2 + W_2^1$
p'	$W_1^1 + W_2^2$	$W_1^2 + W_2^2$	W_2^2
p'	$W_1^2 + W_2^2$	W_2^2	$W_1^1 + W_2^2$
p'	W_2^2	$W_1^1 + W_2^2$	$W_1^2 + W_2^2$
p'	$W_1^1 + W_3^1$	$W_1^2 + W_3^1$	W_3^1
p'	$W_1^2 + W_3^1$	W_3^1	$W_1^1 + W_3^1$
p'	W_3^1	$W_1^1 + W_3^1$	$W_1^2 + W_3^1$
p'	$W_1^1 + W_3^2$	$W_1^2 + W_3^2$	W_3^2
p'	$W_1^2 + W_3^2$	W_3^2	$W_1^1 + W_3^2$
p'	W_3^2	$W_1^1 + W_3^2$	$W_1^2 + W_3^2$
p'	$W_1^1 + W_2^1 + W_3^1$	$W_1^2 + W_2^1 + W_3^1$	$W_2^1 + W_3^1$
p'	$W_1^2 + W_2^1 + W_3^1$	$W_2^1 + W_3^1$	$W_1^1 + W_2^1 + W_3^1$
p'	$W_2^1 + W_3^1$	$W_1^1 + W_2^1 + W_3^1$	$W_1^2 + W_2^1 + W_3^1$
p'	$W_1^1 + W_2^2 + W_3^1$	$W_1^2 + W_2^2 + W_3^1$	$W_2^2 + W_3^1$
p'	$W_1^2 + W_2^2 + W_3^1$	$W_2^2 + W_3^1$	$W_1^1 + W_2^2 + W_3^1$
p'	$W_2^2 + W_3^1$	$W_1^1 + W_2^2 + W_3^1$	$W_1^2 + W_2^2 + W_3^1$
p'	$W_1^1 + W_2^1 + W_3^2$	$W_1^2 + W_2^1 + W_3^2$	$W_2^1 + W_3^2$
p'	$W_1^2 + W_2^1 + W_3^2$	$W_2^1 + W_3^2$	$W_1^1 + W_2^1 + W_3^2$
p'	$W_2^1 + W_3^2$	$W_1^1 + W_2^1 + W_3^2$	$W_1^2 + W_2^1 + W_3^2$
p'	$W_1^1 + W_2^2 + W_3^2$	$W_1^2 + W_2^2 + W_3^2$	$W_2^2 + W_3^2$
p'	$W_1^2 + W_2^2 + W_3^2$	$W_2^2 + W_3^2$	$W_1^1 + W_2^2 + W_3^2$
p'	$W_2^2 + W_3^2$	$W_1^1 + W_2^2 + W_3^2$	$W_1^2 + W_2^2 + W_3^2$

Table 8.7: Real query table – W_1.

Assume that the user wants to download W_2 at some time T_i. Then, at time

$P(Q\|\theta=1,R=0)$	DB 1	$P(Q\|\theta=1,R=0)$	DB 2	$P(Q\|\theta=1,R=0)$	DB 3
$\frac{1}{2}$	W_1^1	$\frac{1}{2}$	W_1^1	$\frac{1}{2}$	W_1^1
$\frac{1}{2}$	W_1^2	$\frac{1}{2}$	W_1^2	$\frac{1}{2}$	W_1^2

Table 8.8: Dummy query table – W_1.

T_i, the user picks a row of queries from Table 8.9 based on the probabilities in the first column, and sends them to each of the three databases. Note that correctness is satisfied as it is possible to decode W_2 from any row of Table 8.9. Next, the user picks M future time instances $t_{i,j}$, $j \in \{1, \ldots, M\}$, and at each time $t_{i,j}$ the user independently and randomly picks a row from Table 8.10 and sends the queries to the databases. This completes the scheme, and the value of M that minimizes the download cost is calculated at the end of this example.

The databases make predictions with the received query at each time t, based on the information available in Table 8.6. As the aposteriori probabilities $P(\theta = k|Q_n = \tilde{Q})$ are proportional to the corresponding probabilities given by $P(Q_n = \tilde{Q}|\theta = k)$ from (8.31), the databases' predictions (using (8.5)) and the corresponding probabilities are shown in Table 8.13.

The probability of error for each type of query is calculated as follows. First, consider the ϵ-deceptive queries with respect to file k, given by W_k^j, $j \in \{1, 2\}$. For these queries, the error probability from the perspective of database n, $n \in \{1, \ldots, N\}$, is given by,

$$P(\hat{\theta}^{[T_i]}_{W_k^j} \neq \theta^{[T_i]}) = P(\theta^{[T_i]} \neq k|Q_n^{[T_i]} = W_k^j) \tag{8.56}$$

$$= \sum_{\ell=1,\ell\neq k}^{3} P(\theta^{[T_i]} = \ell|Q_n^{[T_i]} = W_k^j) \tag{8.57}$$

$P(Q\|\theta=2,R=1)$	Database 1	Database 2	Database 3
p	W_2^1	W_2^2	ϕ
p	W_2^2	ϕ	W_2^1
p	ϕ	W_2^1	W_2^2
p'	$W_1^1+W_2^1$	$W_1^1+W_2^2$	W_1^1
p'	$W_1^1+W_2^2$	W_1^1	$W_1^1+W_2^1$
p'	W_1^1	$W_1^1+W_2^1$	$W_1^1+W_2^2$
p'	$W_1^2+W_2^1$	$W_1^2+W_2^2$	W_1^2
p'	$W_1^2+W_2^2$	W_1^2	$W_1^2+W_2^1$
p'	W_1^2	$W_1^2+W_2^1$	$W_1^2+W_2^2$
p'	$W_2^1+W_3^1$	$W_2^2+W_3^1$	W_3^1
p'	$W_2^2+W_3^1$	W_3^1	$W_2^1+W_3^1$
p'	W_3^1	$W_2^1+W_3^1$	$W_2^2+W_3^1$
p'	$W_2^1+W_3^2$	$W_2^2+W_3^2$	W_3^2
p'	$W_2^2+W_3^2$	W_3^2	$W_2^1+W_3^2$
p'	W_3^2	$W_2^1+W_3^2$	$W_2^2+W_3^2$
p'	$W_1^1+W_2^1+W_3^1$	$W_1^1+W_2^2+W_3^1$	$W_1^1+W_3^1$
p'	$W_1^1+W_2^2+W_3^1$	$W_1^1+W_3^1$	$W_1^1+W_2^1+W_3^1$
p'	$W_1^1+W_3^1$	$W_1^1+W_2^1+W_3^1$	$W_1^1+W_2^2+W_3^1$
p'	$W_1^1+W_2^1+W_3^2$	$W_1^1+W_2^2+W_3^2$	$W_1^1+W_3^2$
p'	$W_1^1+W_2^2+W_3^2$	$W_1^1+W_3^2$	$W_1^1+W_2^1+W_3^2$
p'	$W_1^1+W_3^2$	$W_1^1+W_2^1+W_3^2$	$W_1^1+W_2^2+W_3^2$
p'	$W_1^2+W_2^1+W_3^1$	$W_1^2+W_2^2+W_3^1$	$W_1^2+W_3^1$
p'	$W_1^2+W_2^2+W_3^1$	$W_1^2+W_3^1$	$W_1^2+W_2^1+W_3^1$
p'	$W_1^2+W_3^1$	$W_1^2+W_2^1+W_3^1$	$W_1^2+W_2^2+W_3^1$
p'	$W_1^2+W_2^1+W_3^2$	$W_1^2+W_2^2+W_3^2$	$W_1^2+W_3^2$
p'	$W_1^2+W_2^2+W_3^2$	$W_1^2+W_3^2$	$W_1^2+W_2^1+W_3^2$
p'	$W_1^2+W_3^2$	$W_1^2+W_2^1+W_3^2$	$W_1^2+W_2^2+W_3^2$

Table 8.9: Real query table – W_2.

$P(Q\|\theta=2,R=0)$	DB 1	$P(Q\|\theta=2,R=0)$	DB 2	$P(Q\|\theta=2,R=0)$	DB 3
$\frac{1}{2}$	W_2^1	$\frac{1}{2}$	W_2^1	$\frac{1}{2}$	W_2^1
$\frac{1}{2}$	W_2^2	$\frac{1}{2}$	W_2^2	$\frac{1}{2}$	W_2^2

Table 8.10: Dummy query table – W_2.

$$= \frac{\sum_{\ell=1,\ell\neq k}^{3} P(Q_n^{[T_i]}=W_k^j|\theta^{[T_i]}=\ell)P(\theta^{[T_i]}=\ell)}{P(Q_n^{[T_i]}=W_k^j)} \tag{8.58}$$

$$= \frac{1}{P(Q_n^{[T_i]}=W_k^j)}\frac{2}{3}pe^{\epsilon}, \tag{8.59}$$

$P(Q\|\theta=3,R=1)$	Database 1	Database 2	Database 3
p	W_3^1	W_3^2	ϕ
p	W_3^2	ϕ	W_3^1
p	ϕ	W_3^1	W_3^2
p'	$W_1^1+W_3^1$	$W_1^1+W_3^2$	W_1^1
p'	$W_1^1+W_3^2$	W_1^1	$W_1^1+W_3^1$
p'	W_1^1	$W_1^1+W_3^1$	$W_1^1+W_3^2$
p'	$W_1^2+W_3^1$	$W_1^2+W_3^2$	W_1^2
p'	$W_1^2+W_3^2$	W_1^2	$W_1^2+W_3^2$
p'	W_1^2	$W_1^2+W_3^2$	$W_1^2+W_3^1$
p'	$W_2^1+W_3^1$	$W_2^1+W_3^2$	W_2^1
p'	$W_2^1+W_3^2$	W_3^1	$W_2^1+W_3^1$
p'	W_2^1	$W_2^1+W_3^1$	$W_2^1+W_3^2$
p'	$W_2^2+W_3^1$	$W_2^2+W_3^2$	W_2^2
p'	$W_2^2+W_3^2$	W_2^2	$W_2^2+W_3^1$
p'	W_2^2	$W_2^2+W_3^1$	$W_2^2+W_3^2$
p'	$W_1^1+W_2^1+W_3^1$	$W_1^1+W_2^1+W_3^2$	$W_1^1+W_2^1$
p'	$W_1^1+W_2^1+W_3^2$	$W_1^1+W_2^1$	$W_1^1+W_2^1+W_3^1$
p'	$W_1^1+W_2^1$	$W_1^1+W_2^1+W_3^1$	$W_1^1+W_2^1+W_3^2$
p'	$W_1^2+W_2^1+W_3^1$	$W_1^2+W_2^1+W_3^2$	$W_1^2+W_2^1$
p'	$W_1^2+W_2^1+W_3^2$	$W_1^2+W_2^1$	$W_1^2+W_2^1+W_3^1$
p'	$W_1^2+W_2^1$	$W_1^2+W_2^1+W_3^1$	$W_1^2+W_2^1+W_3^2$
p'	$W_1^1+W_2^2+W_3^1$	$W_1^1+W_2^2+W_3^2$	$W_1^1+W_2^2$
p'	$W_1^1+W_2^2+W_3^2$	$W_1^1+W_2^2$	$W_1^1+W_2^2+W_3^1$
p'	$W_1^1+W_2^2$	$W_1^1+W_2^2+W_3^1$	$W_1^1+W_2^2+W_3^2$
p'	$W_1^2+W_2^2+W_3^1$	$W_1^2+W_2^2+W_3^2$	$W_1^2+W_2^2$
p'	$W_1^2+W_2^2+W_3^2$	$W_1^2+W_2^2$	$W_1^2+W_2^2+W_3^1$
p'	$W_1^2+W_2^2$	$W_1^2+W_2^2+W_3^1$	$W_1^2+W_2^2+W_3^2$

Table 8.11: Real query table – W_3.

$P(Q\|\theta=3,R=0)$	DB 1	$P(Q\|\theta=3,R=0)$	DB 2	$P(Q\|\theta=3,R=0)$	DB 3
$\frac{1}{2}$	W_3^1	$\frac{1}{2}$	W_3^1	$\frac{1}{2}$	W_3^1
$\frac{1}{2}$	W_3^2	$\frac{1}{2}$	W_3^2	$\frac{1}{2}$	W_3^2

Table 8.12: Dummy query table – W_3.

where (8.56) follows from the fact that the databases' prediction on a received query of the form W_k^j is file k with probability 1 from Table 8.13, and the probabilities in (8.59) are obtained from real query tables as they correspond to queries sent at time T_i. Next, the probability of error corresponding to each of the the other queries,

query $\tilde{Q}$	$P(\hat{\theta}_{\tilde{Q}}=1)$	$P(\hat{\theta}_{\tilde{Q}}=2)$	$P(\hat{\theta}_{\tilde{Q}}=3)$
W_1^1	1	0	0
W_1^2	1	0	0
W_2^1	0	1	0
W_2^2	0	1	0
W_3^1	0	0	1
W_3^2	0	0	1
other queries	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Table 8.13: Probabilities of each database predicting the user-required file in Example 2.

i.e., PIR queries that include the null query and sums of two or three elements, is given by,

$$
\begin{aligned}
&P(\hat{\theta}_{\tilde{Q}}^{[T_i]} \neq \theta^{[T_i]}) \\
&\quad = P(\hat{\theta}^{[T_i]} \neq \theta^{[T_i]} | Q_n^{[T_i]} = \tilde{Q}) && (8.60)\\
&\quad = \frac{\sum_{j=1}^{3}\sum_{m=1,m\neq j}^{3} P(\hat{\theta}^{[T_i]} = m, \theta^{[T_i]} = j, Q_n^{[T_i]} = \tilde{Q})}{P(Q_n^{[T_i]} = \tilde{Q})} && (8.61)\\
&\quad = \frac{\sum_{j=1}^{3}\sum_{m=1,m\neq j}^{3} P(\hat{\theta}^{[T_i]} = m | \theta^{[T_i]} = j, Q_n^{[T_i]} = \tilde{Q}) P(Q_n^{[T_i]} = \tilde{Q} | \theta^{[T_i]} = j) P(\theta^{[T_i]} = j)}{P(Q_n^{[T_i]} = \tilde{Q})} && (8.62)\\
&\quad = \frac{1}{P(Q_n^{[T_i]} = \tilde{Q})} \begin{cases} \frac{2p}{3}, & \text{if } \tilde{Q} = \phi \\ \frac{2pe^{\epsilon}}{3}, & \text{if } \tilde{Q} \text{ if of the form } \sum_{s=1}^{\ell} W_{k_s}^{j_s} \text{ for } \ell \in \{2,3\} \end{cases} && (8.63)
\end{aligned}
$$

where (8.63) follows from the fact that $\hat{\theta}^{[T_i]}$ is conditionally independent of $\theta^{[T_i]}$ given Q_n, from (8.5). The probability of error at each time T_i, $i \in \mathbb{N}$, is the same, as the scheme is identical at each T_i, and across all file requirements. Therefore, the

probability of error of each database's prediction, using (8.6) is given by,

$$
\begin{aligned}
P_e &= P(\hat{\theta}^{[T_i]} \neq \theta^{[T_i]}) && (8.64)\\
&= \sum_{\tilde{Q}\in\mathcal{Q}} P(Q_n = \tilde{Q}) P(\hat{\theta}^{[T_i]}_{\tilde{Q}} \neq \theta^{[T_i]}) && (8.65)\\
&= \sum_{k=1}^{3}\sum_{j=1}^{2} P(Q_n = W_k^j)\frac{1}{P(Q_n^{[T_i]} = W_k^j)}\frac{2}{3}pe^{\epsilon} + P(Q_n = \phi)\frac{1}{P(Q_n = \phi)}\frac{2p}{3}\\
&\quad + 20P(Q_n = \hat{Q})\frac{1}{P(Q_n = \hat{Q})}\frac{2pe^{\epsilon}}{3} && (8.66)\\
&= 4pe^{\epsilon} + \frac{2p}{3} + \frac{40pe^{\epsilon}}{3} && (8.67)\\
&= \frac{52e^{\epsilon}+2}{9(8e^{\epsilon}+1)}. && (8.68)
\end{aligned}
$$

where $\mathcal{Q}$ is the set of all queries and $\hat{Q}$ is a query of the form $\sum_{s=1}^{\ell} W_{k_s}^{j_s}$ for $\ell \in \{2,3\}$. The resulting amount of deception is,

$$
D = P_e - \left(1 - \frac{1}{K}\right) = \frac{52e^{\epsilon}+2}{9(8e^{\epsilon}+1)} - \frac{2}{3} = \frac{4(e^{\epsilon}-1)}{9(8e^{\epsilon}+1)}. \tag{8.69}
$$

Therefore, for a required amount of deception $d < \frac{1}{18}$, ϵ is chosen as $\epsilon = \ln\left(\frac{9d+4}{4(1-18d)}\right)$.

Without loss of generality, consider the cost of downloading W_1, which is the same as the expected download cost, as the scheme is symmetric across all file retrievals,

$$
D_L = \frac{1}{L}\left(L \times 3p + \frac{3L}{2} \times 24pe^{\epsilon} + \frac{3L}{2}\sum_{m=0}^{\infty} p_m m\right) = \frac{1+12e^{\epsilon}}{1+8e^{\epsilon}} + \frac{3}{2}\mathbb{E}[M] \tag{8.70}
$$

To find the scheme that achieves the minimum D_L we need to find the minimum

$\mathbb{E}[M]$ that satisfies $P(R=1|\theta=i)=\alpha=\mathbb{E}[\frac{1}{M+1}]=\frac{3(1+8e^{\epsilon})}{2e^{2\epsilon}+24e^{\epsilon}+1}$, i.e., the following optimization problem needs to be solved.

$$\begin{aligned} \min \quad & \mathbb{E}[M] \\ \text{s.t.} \quad & \mathbb{E}\left[\frac{1}{M+1}\right]=\frac{3e^{-2\epsilon}(1+8e^{\epsilon})}{2+e^{-2\epsilon}+24e^{-\epsilon}}. \end{aligned} \tag{8.71}$$

The solution to this problem is given in Lemma 8.1. The resulting minimum download cost for a given value of ϵ, i.e., required level of deception d, is given by,

$$\frac{D_\epsilon}{L}=\frac{1+12e^{\epsilon}}{1+8e^{\epsilon}}+\frac{3}{2}(2u-u(u+1)\alpha), \quad \alpha=\frac{3e^{-2\epsilon}(1+8e^{\epsilon})}{2+e^{-2\epsilon}+24e^{-\epsilon}}, \tag{8.72}$$

where $u=\lfloor\frac{1}{\alpha}\rfloor$. When $d=0$, it follows that $\epsilon=0$, $\alpha=1$ and $u=1$, and the achievable rate is $\frac{9}{13}$, which is equal the PIR capacity for the case $N=3, K=3$.

8.4.3 Generalized DIR Scheme for Arbitrary N and K

In the general DIR scheme proposed in this work, at each time T_i, $i\in\mathbb{N}$, when the user requires to download some file W_k, the user sends a set of real queries to each of the N databases. These queries are picked based on a certain probability distribution, defined on all possible sets of real queries. For the same file requirement, the user sends M dummy queries at future time instances $t_{i,j}$, $j\in\{1,\ldots,M\}$, where $t_{i,j}>T_i$. The dummy queries sent at each time $t_{i,j}$ are randomly selected from a subset of real queries. We assume that the databases are unaware of being deceived, and treat both real and dummy queries the same when calculating their

predictions on the user-required file index at each time they receive a query. The overall probabilities of a given user sending each query for each file requirement is known by the databases. However, the decomposition of these probabilities based on whether each query is used as a real or a dummy query is not known by the databases. It is also assumed that the databases only store the queries received at the current time instance.

The main components of the general scheme include 1) N^K possible sets of real queries to be sent to the N databases for each file requirement and their probabilities, 2) $N-1$ possible sets of dummy queries and their probabilities, 3) overall probabilities of sending each query for each of the K file requirements of the user. Note that 1) and 2) are only known by the user while 3) is known by the databases.

As shown in the examples considered, the set of all possible real queries takes the form of the queries in the probabilistic PIR scheme in Chapter 2 and [49], with a non-uniform probability distribution unlike in PIR. The real query table used when retrieving W_k consists of the following queries:

1. **Single blocks:** W_k is divided into $N-1$ parts, and each part is requested from $N-1$ databases, while requesting nothing ϕ from the remaining database. All cyclic shifts of these queries are considered in the real query table.

2. **Sums of two blocks/Single block:** One database is used to download W_j^l, $l \in \{1, \ldots, N-1\}, j \neq k$ and each one in the rest of the $N-1$ databases is used to download $W_k^r + W_j^l$ for each $r \in \{1, \ldots, N-1\}$. All cyclic shifts of these queries are also considered as separate possible sets of queries.

3. **Sums of three/Two blocks:** One database is used to download $W_{j_1}^{\ell_1} + W_{j_2}^{\ell_2}$, $\ell_1, \ell_2 \in \{1, \ldots, N-1\}$ and $j_1 \neq j_2 \neq k$. Each one in the rest of the $N-1$ databases is used to download $W_{j_1}^{l_1} + W_{j_2}^{l_2} + W_k^r$ for each $r \in \{1, \ldots, N-1\}$. All cyclic shifts of these queries are also considered as separate possible sets of queries.

4. **Sums of $K/K-1$ blocks:** The above process is repeated for all sums of blocks until $K/K-1$.

Out of the N^K different sets of queries described above in the real query table, the queries except ϕ in single blocks, i.e., queries of the form W_k^ℓ, $\ell \in \{1, \ldots, N-1\}$, are chosen as ϵ-deceptive ones with respect to file k, for each $k \in \{1, \ldots, K\}$, and are included in the set of dummy queries sent to databases when the user-required file index is k. The $N-1$ ϵ-deceptive queries W_k^r, $r \in \{1, \ldots, N-1\}$, corresponding to the kth file requirement must guarantee the condition in (8.13). For that, we assign,

$$P(Q_n = W_k^r | \theta = k, R = 1) = p, \quad r \in \{1, \ldots, N-1\} \tag{8.73}$$

and

$$P(Q_n = W_k^r | \theta = j, R = 1) = pe^\epsilon, \quad r \in \{1, \ldots, N-1\}, \quad j \neq k, \tag{8.74}$$

for each database n, $n \in \{1, \ldots, N\}$. The rest of the queries, i.e., ϕ and sums of ℓ blocks where $\ell \in \{2, \ldots, K\}$, are PIR queries in the proposed scheme. Note that

the query ϕ is always coupled with the ϵ-deceptive queries with respect to file index k (required file) for correctness (see Tables 8.7, 8.9, 8.11). Thus, ϕ is assigned the corresponding probability given by,

$$P(Q_n = \phi | \theta = m, R = 1) = p, \quad m \in \{1, \ldots, K\}, \quad n \in \{1, \ldots, N\}. \tag{8.75}$$

Similarly, as the rest of the PIR queries are coupled with ϵ-deceptive queries with respect to file indices j, $j \neq k$, or with other PIR queries, they are assigned the corresponding probability given by,

$$P(Q_n = \hat{Q} | \theta = m, R = 1) = pe^{\epsilon}, \quad m \in \{1, \ldots, K\}, \quad n \in \{1, \ldots, N\}, \tag{8.76}$$

where $\hat{Q}$ is any PIR query in the form of ℓ-sums with $\ell \in \{2, \ldots, K\}$. Since the probabilities of the real queries sent for each file requirement must add up to one, i.e., $\sum_{\tilde{Q} \in \mathcal{Q}} P(Q_n = \tilde{Q} | \theta = m, R = 1) = 1$ for each $m \in \{1, \ldots, K\}$, p is given by,

$$p = \frac{1}{N + (N^K - N)e^{\epsilon}}, \tag{8.77}$$

as there are N query sets in the real query table with probability p, and $N^K - N$ sets with probability pe^{ϵ}. Each ϵ-deceptive query with respect to file index k is chosen with equal probability to be sent to the databases as dummy queries at times $t_{i,j}$ when the file requirement at the corresponding time T_i is W_k. Since there are $N - 1$

deceptive queries,

$$P(Q_n = W_k^r | \theta = k, R = 0) = \frac{1}{N-1}, \quad r \in \{1, \ldots, N-1\}. \tag{8.78}$$

and

$$P(Q_n = W_k^r | \theta = j, R = 0) = 0, \quad r \in \{1, \ldots, N-1\}, \quad j \neq k. \tag{8.79}$$

for each database n, $n \in \{1, \ldots, N\}$. Therefore, for all ϵ-deceptive queries with respect to file index k of the form W_k^i, the condition in (8.14) can be written as,

$$\frac{\alpha}{\alpha + \frac{1}{p(N-1)}(1-\alpha)} = e^{-2\epsilon} \tag{8.80}$$

thus,

$$\alpha = \frac{1}{p(N-1)(e^{2\epsilon}-1)+1} = \frac{N + (N^K - N)e^{\epsilon}}{(N-1)e^{2\epsilon} + (N^K - N)e^{\epsilon} + 1}, \tag{8.81}$$

which characterizes $\alpha = \mathbb{E}\left[\frac{1}{M+1}\right]$. The information available to database n, $n \in \{1, \ldots, N\}$, is the overall probability of receiving each query for each file requirement of the user $P(Q_n = \tilde{Q} | \theta = k)$, $k \in \{1, \ldots, K\}$, given by,

$$\begin{aligned} P(Q_n = \tilde{Q} | \theta = k) &= P(Q_n = \tilde{Q} | \theta = k, R = 1) P(R = 1 | \theta = k) \\ &\quad + P(Q_n = \tilde{Q} | \theta = k, R = 0) P(R = 0 | \theta = k). \end{aligned} \tag{8.82}$$

For ϵ-deceptive queries with respect to file index k, i.e., W_k^j, $j \in \{1, \ldots, N-1\}$, the overall probability in (8.82) from the perspective of database n, $n \in \{1, \ldots, N\}$, is given by,

$$P(Q_n = W_k^j | \theta = \ell) = \begin{cases} \alpha p + \frac{1-\alpha}{N-1} = \frac{e^{2\epsilon}}{(N-1)(e^{2\epsilon}-1)+N+(N^K-N)e^{\epsilon}}, & \ell = k \\ \alpha p e^{\epsilon} = \frac{e^{\epsilon}}{(N-1)(e^{2\epsilon}-1)+N+(N^K-N)e^{\epsilon}}, & \ell \neq k. \end{cases} \tag{8.83}$$

The probability of sending the null query ϕ to database n, $n \in \{1, \ldots, N\}$, for each file-requirement k, $k \in \{1, \ldots, K\}$, is,

$$P(Q_n = \phi | \theta = k) = \alpha p = \frac{1}{(N-1)(e^{2\epsilon}-1) + N + (N^K - N)e^{\epsilon}}. \tag{8.84}$$

For the rest of the PIR queries denoted by $\hat{Q}$, i.e., queries of the form $\sum_{s=1}^{\ell} W_{i_s}^{j_s}$ for $\ell \in \{2, \ldots, K\}$, the overall probability in (8.82), known by each database n, $n \in \{1, \ldots, N\}$ for each file requirement k, $k \in \{1, \ldots, K\}$ is given by,

$$P(Q_n = \hat{Q} | \theta = k) = \alpha p e^{\epsilon} = \frac{e^{\epsilon}}{(N-1)(e^{2\epsilon}-1) + N + (N^K - N)e^{\epsilon}}. \tag{8.85}$$

Based on the query received at a given time t, each database n, $n \in \{1, \ldots, N\}$, calculates the aposteriori probability of the user-required file index being k, $k \in \{1, \ldots, K\}$, using,

$$P(\theta = k | Q_n = \tilde{Q}) = \frac{P(Q_n = \tilde{Q} | \theta = k) P(\theta = k)}{P(Q_n = \tilde{Q})}. \tag{8.86}$$

Since we assume uniform priors, i.e., $P(\theta = k) = \frac{1}{K}$ for all $k \in \{1, \ldots, K\}$, the posteriors are directly proportional to $P(Q_n = \tilde{Q}|\theta = k)$ for each $\tilde{Q}$. Therefore, the databases predict the user-required file index for each query received using (8.5) and (8.83)-(8.85). For example, when the query W_1^1 is received, it is clear that the maximum $P(\theta = k|Q_n = W_1^1)$ in (8.5) is obtained for $k = 1$ from (8.83) and (8.86). The prediction corresponding to any query received is given in Table 8.14 along with the corresponding probability of choosing the given prediction.[5]

Based on the information in Table 8.14, the probability of error when a database n, $n \in \{1, \ldots, N\}$, receives the query W_k^ℓ at some time T_i is given by,

$$P(\hat{\theta}_{W_k^\ell}^{[T_i]} \neq \theta^{[T_i]}) = P(\theta^{[T_i]} \neq k|Q_n^{[T_i]} = W_k^\ell) \tag{8.87}$$

$$= \sum_{j=1, j\neq k}^{K} P(\theta^{[T_i]} = j|Q_n^{[T_i]} = W_k^\ell) \tag{8.88}$$

$$= \frac{\sum_{j=1,j\neq k}^{K} P(Q_n^{[T_i]} = W_k^\ell|\theta^{[T_i]} = j)P(\theta^{[T_i]} = j)}{P(Q_n^{[T_i]} = W_k^\ell)} \tag{8.89}$$

$$= \frac{\frac{1}{K}pe^{\epsilon}(K-1)}{P(Q_n^{[T_i]} = W_k^\ell)}, \tag{8.90}$$

where (8.90) follows from the fact that the user sends real queries based on the probabilities $P(Q_n = \tilde{Q}|\theta = k, R = 1)$ at time T_i.

For all other queries $\tilde{Q}$, the corresponding probability of error is given by,

$$P(\hat{\theta}_{\tilde{Q}}^{[T_i]} \neq \theta^{[T_i]}) = P(\hat{\theta}^{[T_i]} \neq \theta^{[T_i]}|Q_n^{[T_i]} = \tilde{Q}) \tag{8.91}$$

[5]The superscript j in the first column of Table 8.14 corresponds to any index in the set $\{1, \ldots .N-1\}$.

query $\tilde{Q}$	$P(\hat{\theta}_{\tilde{Q}}=1)$	$P(\hat{\theta}_{\tilde{Q}}=2)$	$P(\hat{\theta}_{\tilde{Q}}=3)$	$\dots$	$P(\hat{\theta}_{\tilde{Q}}=K)$
W_1^j	1	0	0	$\dots$	0
W_2^j	0	1	0	$\dots$	0
W_3^j	0	0	1	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
W_K^j	0	0	0	$\dots$	1
other queries	$\frac{1}{K}$	$\frac{1}{K}$	$\frac{1}{K}$	$\dots$	$\frac{1}{K}$

Table 8.14: Probabilities of each database predicting the user-required file.

$$= \frac{\sum_{j=1}^{K}\sum_{m=1,m\neq j}^{K} P(\hat{\theta}^{[T_i]}=m, \theta^{[T_i]}=j, Q_n^{[T_i]}=\tilde{Q})}{P(Q_n^{[T_i]}=\tilde{Q})} \tag{8.92}$$

$$= \frac{\sum_{j=1}^{K}\sum_{m=1,m\neq j}^{K} P(\hat{\theta}^{[T_i]}=m|\theta^{[T_i]}=j, Q_n^{[T_i]}=\tilde{Q})P(Q_n^{[T_i]}=\tilde{Q}|\theta^{[T_i]}=j)P(\theta^{[T_i]}=j)}{P(Q_n^{[T_i]}=\tilde{Q})} \tag{8.93}$$

$$= \frac{1}{P(Q_n^{[T_i]}=\tilde{Q})} \begin{cases} \frac{(K-1)p}{K}, & \text{if } \tilde{Q}=\phi \\ \frac{(K-1)pe^{\epsilon}}{K}, & \text{if } \tilde{Q} \text{ of the form } \sum_{s=1}^{\ell} W_{i_s}^{j_s},\ \ell\in\{2,\dots,K\} \end{cases} \tag{8.94}$$

where (8.94) follows from the fact that $\hat{\theta}^{[T_i]}$ is conditionally independent of $\theta^{[T_i]}$ given Q from (8.5). The probability of error of each database's prediction is given by,

$$P_e = \sum_{\tilde{Q}} P(Q_n^{[T_i]}=\tilde{Q})P(\hat{\theta}^{[T_i]}\neq\theta^{[T_i]}|Q^{[T_i]}=\tilde{Q}) \tag{8.95}$$

$$= \sum_{k=1}^{K}\sum_{\ell=1}^{N-1} P(Q_n^{[T_i]}=W_k^{\ell})\frac{\frac{1}{K}pe^{\epsilon}(K-1)}{P(Q_n^{[T_i]}=W_k^{\ell})} + P(Q_n^{[T_i]}=\phi)\frac{\frac{1}{K}(K-1)p}{P(Q_n^{[T_i]}=\phi)}$$
$$+ (N^K-1-K(N-1))P(P(Q_n^{[T_i]}=\hat{Q})\frac{\frac{1}{K}(K-1)pe^{\epsilon}}{P(Q_n^{[T_i]}=\hat{Q})}) \tag{8.96}$$

$$= pe^{\epsilon}(K-1)(N-1) + \frac{(K-1)p}{K} + \frac{(K-1)pe^{\epsilon}(N^K-1-K(N-1))}{K} \tag{8.97}$$

$$= \frac{(K-1)(1+e^{\epsilon}(N^K-1))}{K(N+(N^K-N)e^{\epsilon})}, \tag{8.98}$$

where $\hat{Q}$ in (8.96) represents the queries of the form $\sum_{s=1}^{\ell} W_{i_s}^{j_s}$ for $\ell \in \{2, \ldots, K\}$. Note that $P(Q_n^{[T_i]} = \hat{Q})$ is the same for each $\hat{Q}$ as $P(Q_n^{[T_i]} = \hat{Q}|\theta = j) = pe^{\epsilon}$ for each $\hat{Q}$ and all $j \in \{1, \ldots, K\}$ from (8.76). Thus, the amount of deception achieved by this scheme for a given ϵ is given by,

$$D = P_e - \left(1 - \frac{1}{K}\right) = \frac{(K-1)(N-1)(e^{\epsilon}-1)}{K(N + (N^K - N)e^{\epsilon})}. \tag{8.99}$$

Therefore, for a required amount of deception d, satisfying $d < \frac{(K-1)(N-1)}{K(N^K-N)}$, the value of ϵ must be chosen as,

$$\epsilon = \ln\left(\frac{dKN + (K-1)(N-1)}{dKN + (K-1)(N-1) - dKN^K}\right). \tag{8.100}$$

The download cost of the general scheme is,

$$D_L = \frac{1}{L}\left(NpL + (N^K - N)pe^{\epsilon}\frac{NL}{N-1} + \frac{NL}{N-1}\mathbb{E}[M]\right) \tag{8.101}$$

$$D_L = Np + \frac{N(N^K - N)}{N-1}pe^{\epsilon} + \left(\frac{N}{N-1}\right)\mathbb{E}[M] \tag{8.102}$$

$$D_L = \frac{N}{N-1}\left(1 - \frac{1}{N + (N^K - N)e^{\epsilon}} + \mathbb{E}[M]\right). \tag{8.103}$$

Following optimization problem needs to be solved to minimize the download cost while satisfying $\alpha = \frac{N+(N^K-N)e^{\epsilon}}{(N-1)e^{2\epsilon}+(N^K-N)e^{\epsilon}+1}$, from (8.51),

$$\begin{aligned} \min \quad & \mathbb{E}[M] \\ \text{s.t.} \quad & \mathbb{E}\left[\frac{1}{M+1}\right] = \frac{N + (N^K - N)e^{\epsilon}}{(N-1)e^{2\epsilon} + (N^K - N)e^{\epsilon} + 1} = \alpha. \end{aligned} \tag{8.104}$$

Lemma 8.1 *The solution to the optimization problem in* (8.104) *is given by,*

$$\mathbb{E}[M] = 2u - u(u+1)\alpha, \tag{8.105}$$

where $u = \lfloor \frac{1}{\alpha} \rfloor$ *for a given value of* α*, which is specified by the required level of deception* d*.*

The proof of Lemma 8.1 is given in the Appendix. The minimum download cost for the general case with N databases, K files and a deception requirement d, is obtained by (8.103) and (8.105). The corresponding maximum achievable rate is given in (7.3).

8.5 Conclusions

We introduced the problem of deceptive information retrieval (DIR), in which a user retrieves a file from a set of independent files stored in multiple databases, while revealing fake information about the required file to the databases, which makes the probability of error of the databases' prediction on the user-required file index high. The proposed scheme achieves rates lower than the PIR capacity when the required level of deception is positive, as it sends dummy queries at distinct time instances to deceive the databases. When the required level of deception is zero, the achievable DIR rate is the same as the PIR capacity.

The probability of error of the databases' prediction on the user-required file

index is calculated at the time of the user's requirement, as defined in Section 8.2. In the proposed scheme, the user sends dummy queries at other (future) time instances as well. As the databases are unaware of being deceived, and are unable to distinguish between the times corresponding to real and dummy queries, they make predictions on the user-required file indices every time a query is received. Note that whenever a query of the form W_k^ℓ is received, the databases prediction is going to be $\hat{\theta} = k$ from Table 8.14. Although this is an incorrect prediction with high probability at times corresponding to user's real requirements, these predictions are correct when W_k^ℓ is used as a dummy query, as W_k^ℓ is only sent as a dummy query when the user requires to download file k. However, the databases are only able to obtain these correct predictions at future time instances, after which the user has already downloaded the required file while also deceiving the databases.

The reason for the requirement of the time dimension is also explained as follows. An alternative approach to using the time dimension is to select a subset of databases to send the dummy queries and to send the real queries to rest of the databases. As explained above, whenever a database receives a query of the form W_k^ℓ as a dummy query, the database predicts the user-required file correctly. Therefore, this approach leaks information about the required file to a subset of databases, right at the time of the retrieval, while deceiving the rest. Hence, to deceive all databases at the time of retrieval, we exploit the time dimension that is naturally present in information retrieval applications that are time-sensitive.

A potential future direction of this work is an analysis on the time dimension. Note that in this work we assume that the databases do not keep track of the previous

queries and only store the information corresponding to the current time instance. Therefore, as long as the dummy queries are sent at distinct time instances that are also different from the time of the user's requirement, the calculations presented in this chapter are valid. An extension of basic DIR can be formulated by assuming that the databases keep track of all queries received and their time stamps. This imposes additional constraints on the problem as the databases now have extra information along the time dimension, which requires the scheme to choose the time instances at which the dummy queries are sent, in such a way that they do not leak any information about the existence of the two types (real and dummy) queries. Another direction is to incorporate the freshness and age of information into DIR, where the user may trade the age of the required file for a reduced download cost, by making use of the previous dummy downloads present in DIR.

8.6 Appendix

8.6.1 Proof of Lemma 8.1

Proof: The solution to the optimization problem in (8.104) for the general case with N databases and K files is as follows. The optimization problem in (8.104), for a required amount of deception d and the corresponding ϵ with $\alpha = \frac{N+(N^K-N)e^{\epsilon}}{(N-1)e^{2\epsilon}+(N^K-N)e^{\epsilon}+1}$ is given by,

$$\min \quad \mathbb{E}[M] = \sum_{m=0}^{\infty} m p_m$$

$$\text{s.t.} \quad \mathbb{E}\left[\frac{1}{m+1}\right] = \sum_{m=0}^{\infty}\left(\frac{1}{m+1}\right)p_m = \alpha$$

$$\sum_{m=0}^{\infty} p_m = 1$$

$$p_m \geq 0, \quad m \in \{0, 1, \dots\}. \tag{8.106}$$

We need to determine the optimum PMF of M that minimizes $\mathbb{E}[M]$ while satisfying the given condition. The Lagrangian L of this optimization problem is given by,

$$L = \sum_{m=0}^{\infty} m p_m + \lambda_1 \left(\sum_{m=0}^{\infty}\left(\frac{1}{m+1}\right)p_m - \alpha\right) + \lambda_2\left(\sum_{m=0}^{\infty} p_m - 1\right) - \sum_{m=0}^{\infty}\mu_m p_m. \tag{8.107}$$

Then, the following set of equations need to be solved to find the minimum $\mathbb{E}[M]$,

$$\frac{\partial L}{\partial p_m} = m + \lambda_1\left(\frac{1}{m+1}\right) + \lambda_2 - \mu_m = 0, \quad m \in \{0, 1, \dots\} \tag{8.108}$$

$$\sum_{m=0}^{\infty}\left(\frac{1}{m+1}\right)p_m = \alpha \tag{8.109}$$

$$\sum_{m=0}^{\infty} p_m = 1 \tag{8.110}$$

$$\mu_m p_m = 0, \quad m \in \{0, 1, \dots\} \tag{8.111}$$

$$\mu_m, p_m \geq 0, \quad m \in \{0, 1, \dots\}. \tag{8.112}$$

Case 1: Assume that the PMF of M contains at most two non-zero probabilities, i.e., $p_0, p_1 \geq 0$ and $p_i = 0$, $i \in \{2, 3, \dots\}$. Then, the conditions in (8.108)-(8.112)

are simplified as,

$$\frac{\partial L}{\partial p_0} = \lambda_1 + \lambda_2 - \mu_0 = 0 \tag{8.113}$$

$$\frac{\partial L}{\partial p_1} = \frac{1}{2}\lambda_1 + \lambda_2 - \mu_1 = -1 \tag{8.114}$$

$$p_0 + \frac{1}{2}p_1 = \alpha \tag{8.115}$$

$$p_0 + p_1 = 1 \tag{8.116}$$

$$\mu_0 p_0 = 0 \tag{8.117}$$

$$\mu_1 p_1 = 0 \tag{8.118}$$

$$\mu_0, \mu_1, p_0, p_1 \geq 0. \tag{8.119}$$

From (8.115) and (8.116) we obtain,

$$p_0 + \frac{1}{2}(1 - p_0) = \alpha \tag{8.120}$$

and thus,

$$p_0 = 2\alpha - 1, \qquad p_1 = 2 - 2\alpha, \tag{8.121}$$

which along with (8.119) implies that this solution is only valid for $\frac{1}{2} \leq \alpha \leq 1$. The corresponding optimum value of $\mathbb{E}[M]$ is given by,

$$\mathbb{E}[M] = 1 - p_0 = 2 - 2\alpha, \qquad \frac{1}{2} \leq \alpha \leq 1. \tag{8.122}$$

Case 2: Now consider the case where at most three probabilities of the PMF of M are allowed to be non-zero. i.e., $p_0, p_1, p_2 \geq 0$ and $p_i = 0$, $i \in \{3, 4, \dots\}$. The set of conditions in (8.108)-(8.112) for this case is,

$$\frac{\partial L}{\partial p_m} = m + \lambda_1 \left(\frac{1}{m+1}\right) + \lambda_2 - \mu_m = 0, \quad m \in \{0, 1, 2\} \tag{8.123}$$

$$\sum_{m=0}^{2} \left(\frac{1}{m+1}\right) p_m = \alpha \tag{8.124}$$

$$\sum_{m=0}^{2} p_m = 1 \tag{8.125}$$

$$\mu_m p_m = 0, \quad m \in \{0, 1, 2\} \tag{8.126}$$

$$\mu_m, p_m \geq 0, \quad m \in \{0, 1, 2\}. \tag{8.127}$$

The set of conditions in (8.123)-(8.127) can be written in a matrix form as,

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \mu_0 \\ \mu_1 \\ \mu_2 \\ p_0 \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -2 \\ \alpha \\ 1 \end{bmatrix}. \tag{8.128}$$

Three of the above eight variables, i.e., either μ_i or p_i for each i, are always zero according to (8.126). We consider all choices of $\{\mu_i, p_i\}$ pairs such that one element

of the pair is equal to zero, and the other one is a positive variable, and solve the system for the non-zero variables. Then we calculate the resulting $\mathbb{E}[M]$, along with the corresponding regions of u for which the solutions are applicable. For each region of u, we find the solution to (8.128) that results in the minimum $\mathbb{E}[M]$. Based on this process, the optimum values of p_i, $i \in \{0, 1, 2\}$, the corresponding ranges of u and the minimum values of $\mathbb{E}[M]$ are given in Table 8.15.

range of α	p_0	p_1	p_2	$\mathbb{E}[M]$
$\frac{1}{3} \leq \alpha \leq \frac{1}{2}$	0	$6\alpha - 2$	$3 - 6\alpha$	$4 - 6\alpha$
$\frac{1}{2} \leq \alpha \leq 1$	$2\alpha - 1$	$2 - 2\alpha$	0	$2 - 2\alpha$

Table 8.15: Solution to Case 2: Optimum PMF of M, valid ranges of α and minimum $\mathbb{E}[M]$.

As an example, consider the calculations corresponding to the case where $\mu_0 > 0$, $\mu_1 = \mu_2 = 0$ which implies $p_0 = 0$, $p_1, p_2 > 0$. Note that for this case, (8.128) simplifies to,

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \mu_0 \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -2 \\ \alpha \\ 1 \end{bmatrix}. \tag{8.129}$$

The values of p_1 and p_2, from the solution of the above system, and the corresponding

range of α, from (8.127), along with the resulting $\mathbb{E}[M]$ are given by,

$$p_1 = 6\alpha - 2, \quad p_2 = 3 - 6\alpha, \quad \frac{1}{3} \leq \alpha \leq \frac{1}{2}, \quad \mathbb{E}[M] = 4 - 6\alpha. \tag{8.130}$$

Case 3: At most four non-zero elements of the PMF of M are considered in this case, i.e., $p_0, p_1, p_2, p_3 \geq 0$ and $p_i = 0$, $i \in \{4, 5, \dots\}$. The conditions in (8.108)-(8.112) can be written in a matrix form as,

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -2 \\ -3 \\ \alpha \\ 1 \end{bmatrix}. \tag{8.131}$$

Using the same method described in Case 2, the optimum values of p_i, $i \in \{0, 1, 2, 3\}$, corresponding ranges of α and the resulting minimum $\mathbb{E}[M]$ for Case 3 are given in Table 8.16.

Case 4: At most five non-zero elements of the PMF of M are considered in this case, i.e., $p_0, p_1, p_2, p_3, p_4 \geq 0$ and $p_i = 0$, $i \in \{5, 6, \dots\}$. The conditions in

range of α	p_0	p_1	p_2	p_3	$\mathbb{E}[M]$
$\frac{1}{4} \leq \alpha \leq \frac{1}{3}$	0	0	$12\alpha - 3$	$4 - 12\alpha$	$6 - 12\alpha$
$\frac{1}{3} \leq \alpha \leq \frac{1}{2}$	0	$6\alpha - 2$	$3 - 6\alpha$	0	$4 - 6\alpha$
$\frac{1}{2} \leq \alpha \leq 1$	$2\alpha - 1$	$2 - 2\alpha$	0	0	$2 - 2\alpha$

Table 8.16: Solution to Case 3: Optimum PMF of M, valid ranges of α and minimum $\mathbb{E}[M]$.

(8.108)-(8.112) can be written in a matrix form as,

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \frac{1}{2} & 1 & 0 & -1 & 0 & 0 & 0 & 0 & \dots & 0 \\ \frac{1}{3} & 1 & 0 & 0 & -1 & 0 & 0 & 0 & \dots & 0 \\ \frac{1}{4} & 1 & 0 & 0 & 0 & -1 & 0 & 0 & \dots & 0 \\ \frac{1}{5} & 1 & 0 & 0 & 0 & 0 & -1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 0 & \dots & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -2 \\ -3 \\ -4 \\ \alpha \\ 1 \end{bmatrix}. \tag{8.132}$$

Using the same method as before, the optimum values of p_i, $i \in \{0, 1, 2, 3, 4\}$, the corresponding ranges of α and the resulting minimum $\mathbb{E}[M]$ for Case 4 are given in Table 8.17.

Note that the PMF of M and the resulting $\mathbb{E}[M]$ are the same for a given

range of α	p_0	p_1	p_2	p_3	p_4	$\mathbb{E}[M]$
$\frac{1}{5} \leq \alpha \leq \frac{1}{4}$	0	0	0	$20\alpha - 4$	$5 - 20\alpha$	$8 - 20\alpha$
$\frac{1}{4} \leq \alpha \leq \frac{1}{3}$	0	0	$12\alpha - 3$	$4 - 12\alpha$	0	$6 - 12\alpha$
$\frac{1}{3} \leq \alpha \leq \frac{1}{2}$	0	$6\alpha - 2$	$3 - 6\alpha$	0	0	$4 - 6\alpha$
$\frac{1}{2} \leq \alpha \leq 1$	$2\alpha - 1$	$2 - 2\alpha$	0	0	0	$2 - 2\alpha$

Table 8.17: Solution to Case 4: Optimum PMF of M, valid ranges of α and minimum $\mathbb{E}[M]$.

α in all cases (see Tables 8.15-8.17) irrespective of the support of the PMF of M considered. Therefore, we observe from the above cases that, for a given α in the range $\frac{1}{\ell+1} \leq \alpha \leq \frac{1}{\ell}$, $\mathbb{E}[M]$ is minimized when the PMF of M is such that,

$$p_\ell, p_{\ell-1} > 0, \text{ and } p_i = 0 \text{ for } i \in \mathbb{Z}^+ \setminus \{\ell, \ell-1\}, \tag{8.133}$$

which requires p_ℓ and $p_{\ell-1}$ to satisfy,

$$p_\ell + p_{\ell-1} = 1 \tag{8.134}$$

$$\mathbb{E}\left[\frac{1}{M+1}\right] = p_\ell \frac{1}{\ell+1} + p_{\ell-1}\frac{1}{\ell} = \alpha. \tag{8.135}$$

Therefore, for a given α in the range $\frac{1}{\ell+1} \leq \alpha \leq \frac{1}{\ell}$, the optimum PMF of M and the resulting minimum $\mathbb{E}[M]$ are given by,

$$p_\ell = (\ell+1)(1-\ell\alpha), \quad p_{\ell-1} = \ell((\ell+1)\alpha - 1), \quad \mathbb{E}[M] = 2\ell - \alpha\ell(\ell+1). \tag{8.136}$$

■

CHAPTER 9

Conclusions

In this dissertation, we used information and coding theoretic tools to achieve user-privacy in information retrieval and transmission. We first considered the problem of PIR, extended the concepts to PRUW, which has applications in distributed learning, and further extended beyond privacy to deception in DIR.

In Chapter 2, we investigated the problem of semantic PIR, where the messages in the classical PIR setting are allowed to have different semantics such as different message sizes and arbitrary popularity profiles. The goal of this work was to investigate how the different semantics in a given practical PIR setting can be exploited to improve the corresponding classical PIR rate. As the main result of this work, we characterized the capacity of semantic PIR with achievable schemes and a converse proof. This result implies that the semantic PIR capacity is equal to the classical PIR capacity if all messages are equal in size irrespective of the popularity profile. We derived a necessary and sufficient condition for the semantic PIR capacity to exceed the classical PIR capacity. In particular, if the longer messages are retrieved more often, there is a strict retrieval rate gain from exploiting

the message semantics. Furthermore, the results show that for all message sizes and priors, the semantic PIR capacity exceeds the achievable rate of classical PIR with zero-padding, which zero-pads all messages to equalize their sizes. The extensions of semantic PIR to coded databases and colluding databases were also analyzed separately, where the complete characterizations of the capacities of the two cases were presented along with the corresponding optimal schemes.

In Chapters 3-5 we investigated the problem of PRUW in relation to private FSL. In Chapter 3, we provided a basic PRUW scheme that performs private FSL by only downloading and uploading twice as many bits as the size of a submodel. This scheme achieves the lowest known total communication cost thus far for FSL that guarantees information-theoretic privacy of users. The basic scheme was extended to private FSL with top r sparsification in Chapter 4, which further reduces the reading and writing costs to $\approx 2r$ times the size of a submodel, where r is the sparsification rate, typically in the range of 10^{-2} to 10^{-3}, while guaranteeing information-theoretic privacy of the updating submodel index, the values of the sparse updates, and the positions of the sparse updates. This is achieved using a permutation technique which is based on Shannon's one time pad theorem and certain properties of Lagrange polynomials.

In Chapter 5, we considered random sparsification in private FSL. The problem setting was formulated in terms of a rate-distortion characterization, and the resulting asymptotic normalized reading and writing costs are both equal to $\approx 2r$, where $r = 1 - \tilde{D}$, where $\tilde{D}$ is the distortion allowed in the corresponding phase, i.e., a linear rate-distortion relation is achievable in PRUW. Random sparsification out-

performs (or performs equally) top r sparsification in terms of the communication cost when similar sparsification rates are considered. However, random sparsification may not be as effective as top r sparsification since it does not capture the most significant variations of the gradients in the stochastic gradient descent (SGD) process of the underlying learning task. This may have an adverse effect on the accuracy and the convergence time of the model.

The ideas of PRUW were extended to private FL with top r sparsification in Chapter 6, where the privacy concern is only on the values and the positions of the sparse updates/parameters. We proposed four schemes with different properties to perform FL with top r sparsification without revealing the values or the positions of the sparse updates/parameters to the databases. The schemes follow the same permutation technique introduced in Chapter 4, which requires a large storage space in FL due to the large model sizes. To this end, we generalized the schemes to incur a reduced storage cost at the expense of a certain amount of information leakage, using a model segmentation mechanism. Furthermore, we introduced single-stage and two-stage permutation techniques to control the information leakage, based on the user's requirements. In general, we characterized the trade-off between the communication cost, storage complexity and information leakage in private FL with top r sparsification.

In Chapter 7, we considered the problem of information-theoretically private FSL with storage constrained databases. In this work, we provided a storage mechanism and a PRUW scheme that efficiently utilizes the available storage in the databases for any given set of arbitrary storage constraints, while achieving the

minimum total communication cost within the algorithm. The storage mechanism combines MDS coding and submodel division, which proves to be more effective in terms of the communication cost compared to what is achieved with MDS coding and submodel division individually. We provided different achievable schemes for homogeneous and heterogeneous storage constraints. The scheme proposed for heterogeneous constraints is based on finding the optimum MDS codes and the respective fractions of submodels stored using them for a given set of arbitrary storage constraints, while guaranteeing the information-theoretic privacy of the updating submodel index and the values of the updates. For homogeneous storage constraints, the proposed scheme characterizes the minimum achievable costs within the algorithm for any given constraint using the lower convex hull boundary of a set of achievable (constraint, cost) pairs determined by the scheme itself.

In Chapter 8, we introduced the problem of DIR, in which a user downloads a required file out of multiple files stored in a system of databases, while deceiving the databases by revealing fake information (information about a different file) about the user-required file index. We proposed a scheme that performs DIR for a given level of deception required, with the goal of minimizing the download cost, which makes use of the time-dimension present in the problem setting. The proposed scheme achieves a rate that decreases with increasing amount of deception required, and achieves the PIR capacity when the required level of deception is zero. In this work, we assume that the databases only keep track of the current information. A potential future direction would be to relax this assumption and carry out a privacy analysis when the databases have access to past data as well. This may open up

new problems introducing freshness and age of information into DIR.

The contents of Chapter 2 are published in [108–110], Chapter 3 in [111, 112], Chapter 4 in [111, 113], Chapter 5 in [111, 114], Chapter 6 in [115–117], Chapter 7 in [118–120] and Chapter 8 in [121].

Bibliography

[1] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, November 1998.

[2] S. Yekhanin. Private information retrieval. *Communications of the ACM*, 53(4):68–73, April 2010.

[3] H. Sun and S. A. Jafar. The capacity of private information retrieval. *IEEE Transactions on Infomation Theory*, 63(7):4075–4088, July 2017.

[4] 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.

[5] 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. In *IEEE ISIT*, June 2017.

[6] Z. Jia, H. Sun, and S. A. Jafar. The capacity of private information retrieval with disjoint colluding sets. In *IEEE Globecom*, December 2017.

[7] X. Yao, N. Liu, and W. Kang. The capacity of private information retrieval under arbitrary collusion patterns. In *IEEE ISIT*, June 2020.

[8] R. Bitar and S. El Rouayheb. Staircase-PIR: Universally robust private information retrieval. In *IEEE ITW*, pages 1–5, November 2018.

[9] K. Banawan and S. Ulukus. The capacity of private information retrieval from coded databases. *IEEE Transactions on Information Theory*, 64(3):1945–1956, March 2018.

[10] S. Kumar, H.-Y. Lin, E. Rosnes, and A. G. i Amat. Achieving maximum distance separable private information retrieval capacity with linear codes. *IEEE Transactions on Information Theory*, 65(7):4243–4273, July 2019.

[11] 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, August 2020.

[12] 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.

[13] Y. Zhang and G. Ge. A general private information retrieval scheme for MDS coded databases with colluding servers. *Designs, Codes and Cryptography*, 87(11), November 2019.

[14] 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.

[15] L. Holzbaur, R. Freij-Hollanti, J. Li, and C. Hollanti. Towards the capacity of private information retrieval from coded and colluding servers. Available at arXiv: 1903.12552.

[16] R. Tandon, M. Abdul-Wahid, F. Almoualem, and D. Kumar. PIR from storage constrained databases – coded caching meets PIR. In *IEEE ICC*, May 2018.

[17] 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, November 2020.

[18] 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, June 2020.

[19] Y.-P. Wei, B. Arasli, K. Banawan, and S. Ulukus. The capacity of private information retrieval from decentralized uncoded caching databases. *Information*, 10, December 2019.

[20] K. Banawan, B. Arasli, and S. Ulukus. Improved storage for efficient private information retrieval. In *IEEE ITW*, August 2019.

[21] C. Tian. On the storage cost of private information retrieval. *IEEE Transactions on Information Theory*, 66(12):7539–7549, December 2020.

[22] N. Raviv and I. Tamo. Private information retrieval in graph based replication systems. In *IEEE ISIT*, June 2018.

[23] K. Banawan and S. Ulukus. Private information retrieval from non-replicated databases. In *IEEE ISIT*, pages 1272–1276, July 2019.

[24] N. Woolsey, R. Chen, and M. Ji. Uncoded placement with linear sub-messages for private information retrieval from storage constrained databases. *IEEE Trans. on Comm.*, 68(10):6039–6053, October 2020.

[25] 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, February 2019.

[26] 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.

[27] X. Yao, N. Liu, and W. Kang. The capacity of multi-round private information retrieval from Byzantine databases. In *IEEE ISIT*, July 2019.

[28] 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, October 2018.

[29] Y. Zhang and G. Ge. Multi-file private information retrieval from MDS coded databases with colluding servers. Available at arXiv: 1705.03186.

[30] H. Sun and S. A. Jafar. The capacity of symmetric private information retrieval. *IEEE Transactions on Information Theory*, 65(1):322–329, January 2019.

[31] Q. Wang, H. Sun, and M. Skoglund. Symmetric private information retrieval with mismatched coded messages and randomness. In *IEEE ISIT*, pages 365–369, July 2019.

[32] Q. Wang and M. Skoglund. Symmetric private information retrieval from MDS coded distributed storage with non-colluding and colluding servers. *IEEE Transactions on Information Theory*, 65(8):5160–5175, August 2019.

[33] Z. Wang, K. Banawan, and S. Ulukus. Private set intersection: A multi-message symmetric private information retrieval perspective. *IEEE Trans. on Info. Theory*, 68(3):2001–2019, March 2022.

[34] Z. Wang, K. Banawan, and S. Ulukus. Multi-party private set intersection: An information-theoretic approach. *IEEE Journal on Selected Areas in Information Theory*, 2(1):366–379, 2021.

[35] 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, April 2020.

[36] Z. Chen, Z. Wang, and S. Jafar. The capacity of T-private information retrieval with private side information. Available at arXiv:1709.03022.

[37] Y.-P. 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, December 2019.

[38] S. P. Shariatpanahi, M. J. Siavoshani, and M. A. Maddah-Ali. Multi-message private information retrieval with private side information. In *IEEE ITW*, pages 1–5, November 2018.

[39] A. Heidarzadeh, B. Garcia, S. Kadhe, S. E. Rouayheb, and A. Sprintson. On the capacity of single-server multi-message private information retrieval with side information. In *Allerton Conference*, pages 180–187, October 2018.

[40] S. Li and M. Gastpar. Single-server multi-message private information retrieval with side information. In *Allerton Conference*, pages 173–179, October 2018.

[41] S. Li and M. Gastpar. Converse for multi-server single-message PIR with side information. In *CISS*, March 2020.

[42] Y.-P. Wei and S. Ulukus. The capacity of private information retrieval with private side information under storage constraints. *IEEE Transactions on Information Theory*, 66(4):2023–2031, April 2020.

[43] R. Tandon. The capacity of cache aided private information retrieval. In *Allerton Conference*, October 2017.

[44] M. Kim, H. Yang, and J. Lee. Cache-aided private information retrieval. In *Asilomar Conference*, October 2017.

[45] Y.-P. 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.

[46] Y.-P. Wei, K. Banawan, and S. Ulukus. Cache-aided private information retrieval with partially known uncoded prefetching: Fundamental limits. *IEEE JSAC*, 36(6):1126–1139, June 2018.

[47] S. Kumar, A. G. i Amat, E. Rosnes, and L. Senigagliesi. Private information retrieval from a cellular network with caching at the edge. *IEEE Transactions on Communications*, 67(7):4900–4912, July 2019.

[48] T. Guo, R. Zhou, and C. Tian. On the information leakage in private information retrieval systems. *IEEE Transactions on Information Forensics and Security*, 15:2999–3012, December 2020.

[49] I. Samy, R. Tandon, and L. Lazos. On the capacity of leaky private information retrieval. In *IEEE ISIT*, pages 1262–1266, July 2019.

[50] I. Samy, M. A. Attia, R. Tandon, and L. Lazos. On the capacity of latent variable private information retrieval. In *IEEE ISIT*, July 2021.

[51] J. Xu and Z. Zhang. Building capacity-achieving PIR schemes with optimal sub-packetization over small fields. In *IEEE ISIT*, pages 1749–1753, June 2018.

[52] C. Tian, H. Sun, and J. Chen. Capacity-achieving private information retrieval codes with optimal message size and upload cost. *IEEE Transactions on Infomation Theory*, 65(11):7613–7627, Nov 2019.

[53] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication efficient learning of deep networks from decentralized data. *AISTATS*, April 2017.

[54] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, January 2019.

[55] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, and M. Bennis *et al.* Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, June 2021.

[56] T. Li, A. K. Sahu, A. S. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37:50–60, May 2020.

[57] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, May 2019.

[58] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, May 2017.

[59] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, May 2019.

[60] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX*, April 2019.

[61] J. Geiping, H. Bauermeister, H. Droge, and M. Moeller. Inverting gradients–how easy is it to break privacy in federated learning? Available online at arXiv:2003.14053.

[62] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *NeurIPS*, December 2019.

[63] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Infocom*, April-May 2019.

[64] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, October 2017.

[65] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, August 2014.

[66] H. Ono and T. Takahashi. Locally private distributed reinforcement learning. Available online at arXiv:2001.11718.

[67] Y. Li, T. Chang, and C. Chi. Secure federated averaging algorithm with differential privacy. *IEEE MLSE*, September 2020.

[68] N. Agarwal, A. Suresh, F. Yu, S. Kumar, and H. B. McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *NeurIPS*, December 2018.

[69] B. Balle, G. Barthe, and M. Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *NeurIPS*, December 2018.

[70] H. Ono and T. Takahashi. Differentially private cross-silo federated learning. Available online at arXiv:2007.05553.

[71] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *ICLR*, May 2018.

[72] S. Asoodeh and F. Calmon. Differentially private federated learning: An information-theoretic perspective. In *ICML-FL*, July 2020.

[73] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. Available online at arXiv:1811.12469.

[74] B. Balle, J. Bell, A. Gascon, , and K. Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, August 2019.

[75] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. Suresh. Shuffled model of differential privacy in federated learning. In *AISTAT*, April 2021.

[76] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. In *NeurIPS*, December 2017.

[77] S. Ulukus, S. Avestimehr, M. Gastpar, S. A. Jafar, R. Tandon, and C. Tian. Private retrieval, computing and learning: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications*, 40(3):729–748, January 2022.

[78] C. Naim, R. D'Oliveira, and S. El Rouayheb. Private multi-group aggregation. *IEEE ISIT*, July 2021.

[79] J. Wangni, J. Wang, et al. Gradient sparsification for communication-efficient distributed optimization. In *NeurIPS*, December 2018.

[80] S. Li, Q. Qi, et al. GGS: General gradient sparsification for federated learning in edge computing. In *IEEE ICC*, June 2020.

[81] P Han, S. Wang, and K. Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In *IEEE ICDCS*, November 2020.

[82] S. Shi, K. Zhao, Q. Wang, Z. Tang, and X. Chu. A convergence analysis of distributed SGD with communication-efficient gradient sparsification. In *IJCAI*, August 2019.

[83] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli. The convergence of sparsified gradient methods. In *NeurIPS*, December 2018.

[84] Y. Sun, S. Zhou, Z. Niu, and D. Gunduz. Time-correlated sparsification for efficient over-the-air model aggregation in wireless federated learning. Available online at arXiv:2202.08420.

[85] L. Barnes, H. Inan, B. Isik, and A. Ozgur. rTop-k: A statistical estimation approach to distributed SGD. *IEEE JSAIT*, 1(3):897–907, November 2020.

[86] E. Ozfatura, K. Ozfatura, and D. Gunduz. Time-correlated sparsification for communication-efficient federated learning. In *IEEE ISIT*, July 2021.

[87] D. Basu, D. Data, C. Karakus, and S. Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations. *IEEE JSAIT*, 1(1):217–226, May 2020.

[88] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *AISTATS*, August 2020.

[89] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *NeurIPS*, December 2017.

[90] N. Shlezinger, M. Chen, Y. Eldar, H. Poor, and S. Cui. Federated learning with quantization constraints. In *IEEE ICASSP*, May 2020.

[91] 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 *MobiCom*, April 2020.

[92] M. Kim and J. Lee. Information-theoretic privacy in federated submodel learning. *ICT express*, February 2022.

[93] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen. Secure federated submodel learning. Available online at arXiv:1911.02254.

[94] Z. Jia and S. A. Jafar. X-secure T-private federated submodel learning with elastic dropout resilience. *IEEE Trans. on Info. Theory*, 68(8):5418–5439, August 2022.

[95] Z. Jia and S. A. Jafar. X-secure T-private federated submodel learning. In *IEEE ICC*, June 2021.

[96] Z. Wang and S. Ulukus. Private federated submodel learning via private set union. Available online at arXiv:2301.07686.

[97] 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, September 2019.

[98] F. Sattler, S. Wiedemann, K. Müller, and W. Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE trans. on neural networks and learning systems*, 31(9):3400–3413, November 2019.

[99] I. Samy, M. Attia, R. Tandon, and L. Lazos. Asymmetric leaky private information retrieval. *IEEE Transactions on Information Theory*, 67(8):5352–5369, August 2021.

[100] D. Liebowitz, S. Nepal, K. Moore, C. Christopher, S. Kanhere, D. Nguyen, R. Timmer, M. Longland, and K. Rathakumar. Deception for cyber defence: Challenges and opportunities. In *TPS-ISA*, December 2021.

[101] A. Yarali and F. Sahawneh. Deception: Technologies and strategy for cybersecurity. In *SmartCloud*, December 2019.

[102] C. Faveri and A. Moreira. Designing adaptive deception strategies. In *QRS-C*, August 2016.

[103] W. Tounsi. Cyber deception, the ultimate piece of a defensive strategy - proof of concept. In *CSNet*, October 2022.

[104] A. Sarr, A. Anwar, C. Kamhoua, N. Leslie, and J. Acosta. Software diversity for cyber deception. In *IEEE Globecom*, December 2020.

[105] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, October 1949.

[106] A. Yener and S. Ulukus. Wireless physical layer security: Lessons learned from information theory. *Proceedings of the IEEE*, 103(10):1814–1825, October 2015.

[107] J. Xie and S. Ulukus. Secure degrees of freedom of multi-user networks: One-time-pads in the air via alignment. *Proceedings of the IEEE*, 103(10):1857–1873, October 2015.

[108] S. Vithana, K. Banawan, and S. Ulukus. Semantic private information retrieval. *IEEE Transactions on Information Theory*, 68(4):2635–2652, April 2022.

[109] S. Vithana, K. Banawan, and S. Ulukus. Semantic private information retrieval from MDS-coded databases. In *IEEE ISIT*, July 2021.

[110] S. Vithana, K. Banawan, and S. Ulukus. Semantic private information retrieval: Effects of heterogeneous message sizes and popularities. In *IEEE Globecom*, December 2020.

[111] S. Vithana and S. Ulukus. Private read update write (PRUW) in federated submodel learning (FSL): Communication efficient schemes with and without sparsification. *IEEE Transactions on Information theory*, 2023.

[112] S. Vithana and S. Ulukus. Efficient private federated submodel learning. In *IEEE ICC*, May 2022.

[113] S. Vithana and S. Ulukus. Private federated submodel learning with sparsification. In *IEEE ITW*, November 2022.

[114] S. Vithana and S. Ulukus. Rate distortion tradeoff in private read update write in federated submodel learning. In *Asilomar Conference*, October 2022.

[115] S. Vithana and S. Ulukus. Rate-privacy-storage tradeoff in federated learning with top r sparsification. In *IEEE ICC*, May 2023.

[116] S. Vithana and S. Ulukus. Model segmentation for storage efficient private federated learning with top r sparsification. In *CISS*, March 2023.

[117] S. Vithana and S. Ulukus. Private read-update-write with controllable information leakage for storage-efficient federated learning with top r sparsification. Available online at arXiv:2303.04123.

[118] S. Vithana and S. Ulukus. Private read update write (PRUW) with storage constrained databases. In *IEEE ISIT*, June 2022.

[119] S. Vithana and S. Ulukus. Private read update write (PRUW) with heterogeneous databases. In *IEEE ISIT*, June 2023.

[120] S. Vithana and S. Ulukus. Information-theoretically private federated submodel learning with storage constrained databases. Available online at arXiv:2307.06323.

[121] S. Vithana and S. Ulukus. Deceptive information retrieval. Available online at arXiv:2307.04727.