Privacy, Security, and Repair in Distributed Storage Systems

Siddhartha Kumar

Thesis for the Degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2018

UNIVERSITY OF BERGEN

Privacy, Security, and Repair in Distributed Storage Systems

Siddhartha Kumar

Thesis for the Degree of Philosophiae Doctor (PhD)
at the University of Bergen

2018

Date of defence: 17.10.2018

Year: 2018

Title: Privacy, Security, and Repair in Distributed Storage Systems

Name: Siddhartha Kumar

Print: Skipnes Kommunikasjon / University of Bergen

Privacy, Security, and Repair in Distributed Storage Systems

Siddhartha Kumar

Dissertation for the degree of Philosophiae Doctor (PhD)

Department of Informatics
University of Bergen

August 2018

Acknowledgements

Eirik and Àlex, Suman and Bjørn, Mum and Dad, Simula UiB.

Thanks, for making the journey enjoyable.

Abstract

We are living in the age of information where our lives are shaped by information and communication technologies. As a consequence, there is an explosion in the amount of generated data. Distribute storage systems (DSSs) are a storage technology wherein large amounts of data are stored on a network of inexpensive storage nodes in a distributed fashion, in an efficient and inexpensive way. This thesis explores three aspects of DSSs: efficient storage, security, and privacy.

We start with the description of DSSs as they are the underlying theme for the majority of the thesis. For such systems, we propose a code construction that performs efficient repair of failed systematic nodes with low repair complexity. We construct such codes by concatenating two different classes of codes. The first code serves to provide the ability to tolerate node failures, while the second allows for efficient and low complexity repair. The proposed codes achieve better repair bandwidth compared to maximum distance separable (MDS) codes, codes constructed using piggybacks, and local reconstruction/Pyramid codes, while they have a better repair complexity compared to MDS, Zigzag, Pyramid codes, and codes constructed using piggybacks.

Next, we consider the notion of information-theoretic security in DSSs. To this end, we assume an eavesdropper model that has access to the data stored on a subset of nodes and a subset of the data exchanged during the repair of nodes. We present a secure coding scheme that is secure against this adversary. Furthermore, the proposed scheme ensures an efficient node repair. The security is achieved by appending the data with random symbols, and encoding the resulting sequence using a concatenated coding scheme consisting of a Gabidulin code and a repairable fountain code.

We then move toward the problem of private information retrieval (PIR) in DSSs. PIR is the problem of retrieving data from the nodes without revealing the identity of the requested data (or files) to the nodes. A scheme that achieves PIR is referred to as a PIR protocol, and its efficiency is determined by its PIR rate, i.e., the ratio of downloaded data and the requested file size. For the DSS scenario, we present PIR protocols for two cases: first, where some nodes act as spies, which do not collaborate, with a goal of determining what the user is requesting. We refer to this scenario as the noncolluding case, and, second, where these nodes collaborate together in order to determine the request of a user, which we refer to as the colluding case. For the noncolluding case, we show that it is possible to achieve MDS-PIR capacity, i.e., the maximum PIR rate achieved by any protocol when the DSS stores the data using an MDS code, even when the codes used by the DSS are non-MDS. We go on to give a necessary and a sufficient condition based on the structure of the storage code for our PIR protocol to achieve this capac-

ity. We refer to such codes as MDS-PIR capacity-achieving codes. Furthermore, we show that the rates achieved by these codes using the proposed protocol are indeed equal to the maximum PIR rate achieved by these codes under any protocol. Subsequently, we show that cyclic codes, Reed-Muller (RM) codes, and a class of distance-optimal local reconstruction codes are MDS-PIR capacity-achieving codes. For the colluding case, we present a PIR protocol for DSSs that store data using arbitrary linear codes. Following this, as in the noncolluding case, we give a necessary and a sufficient condition based on the structure of the underlying codes that allow the proposed protocol to achieve an upper bound on the PIR rate and present codes that allow this. In particular, we show that if the underlying codes are RM codes, then the protocol achieves the maximum PIR rate.

The concepts learned while finding solutions to the privacy issues for DSSs are then applied to distributed caching in wireless networks, which is one of the fundamental techniques that will be implemented in future 5G wireless networks. In the considered distributed caching scenario, the files are encoded using MDS codes and then cached on the small-cell base stations (SBSs), with the goal to reduce the backhaul usage. We introduce the notion of PIR in this setting and present a protocol that achieves PIR. One essential difference between DSSs and distributed caching is that in distributed caching the files to be cached are typically cached using codes with different rates that depend on the file popularity distribution. We assume that the SBSs can collude and as such the protocol is a generalization of the protocol considered for the colluding case in DSSs. As a surprising result, we show that to minimize backhaul usage, uniform content placement, i.e., the files are cached using a single code, is optimal.

Keywords: Concatenated codes, distributed caching, distributed storage systems, fountain codes, information-theoretic security, linear codes, node repair, private information retrieval.

List of Papers

This thesis is based on the following publications:

Paper I
S. Kumar, A. Graell i Amat, I. Andriyanova, F. Brännström, E. Rosnes, *Code Constructions for Distributed Storage With Low Repair Bandwidth and Low Repair Complexity*, IEEE Transactions on Communications, to appear.

Paper II
S. Kumar, E. Rosnes, A. Graell i Amat, *Secure Repairable Fountain Codes*, IEEE Communications Letters, vol. 20, no. 8, August 2016.

Paper III
S. Kumar, H.-Y. Lin, E. Rosnes, A. Graell i Amat, *Achieving Maximum Distance Separable Private Information Retrieval Capacity With Linear Codes*, submitted to IEEE Transactions on Information Theory, revised in August 2018.

Paper IV
H.-Y. Lin, S. Kumar, E. Rosnes, A. Graell i Amat, *Asymmetry Helps: Improved Private Information Retrieval Protocols for Distributed Storage*, in Proc. IEEE Information Theory Workshop (ITW), Guangzhou, China, November 2018.

Paper V
S. Kumar, A. Graell i Amat, E. Rosnes, L. Senigagliesi, *Private Information Retrieval From a Cellular Network With Caching at the Edge*, submitted to IEEE Transactions on Communications.

Other publications by the author which are not included in this thesis, but are included within the papers of the thesis, are:

- S. Kumar, A. Graell i Amat, I. Andriyanova, F. Brännström, *A Family of Erasure Correcting Codes with Low Repair Bandwidth and Low Repair Complexity*, in Proc. IEEE Global Communications Conf. (GLOBECOM), San Diego, CA, December 2015.
- S. Kumar, E. Rosnes, A. Graell i Amat, *Private Information Retrieval in Distributed Storage Systems Using an Arbitrary Linear Code*, in Proc. IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, June 2017.
- H.-Y. Lin, S. Kumar, E. Rosnes, A. Graell i Amat, *An MDS-PIR Capacity-Achieving PIR Protocol for Distributed Storage Using Non-MDS Linear Codes*, in Proc. IEEE International Symposium on Information Theory (ISIT), Vail, CO, June 2018.
- S. Kumar, H.-Y. Lin, E. Rosnes, A. Graell i Amat, *Local Reconstruction Codes: A class of MDS-PIR Capacity-Achieving Codes*, in Proc. IEEE Information Theory Workshop (ITW), Guangzhou, China, November 2018.

Contents

List of Figures

List of Tables

Part A

Introduction

Chapter 1

Background

At the outset of the 21st century there was a remarkable trend: integrated circuits–used in consumer products such as mobile phones, tablets, and personal computers–were getting smaller, cheaper, and computationally faster. It continued to a point where these consumer products became easily accessible to the general public. In parallel, the internet expanded due to technological advances in fiber optics and microprocessors, giving easy and efficient access to the population. This marked the start of the *information age*.

The information age is characterized by the fact that information and communication technologies shape our lives, culture, and civic discourse [1]. In other words, producing, digesting and processing information has become a norm. This, in turn, has led to an explosion of digital data. In 2009, it was estimated that there is at least an average of 200 terabytes of stored data per company with more than 1000 employees across all sectors of the US economy [2]. Furthermore, according to an EMC report, by the year 2020, 40000 exabytes[1] of data will have been either created, replicated, or consumed worldwide annually [3]. In fact, companies such as Google and Facebook handle terabytes of data each day. For example, Facebook stores petabytes of data on its analytics cluster [4]. Storing such large amounts of data on a single device is inefficient, unreliable, and expensive.

Distributed storage system (DSS) is a new concept introduced by the storage industry to overcome such challenges. Distributed storage is a technique of storing large amounts of data across a network of relatively inexpensive storage devices (from hereon referred to as nodes). Storing data in such a manner introduces scalability, which is to say that the storage capacity of such systems can be scaled without destabilizing the DSS [5]. This can be achieved by simply adding more nodes to the network whenever required. Furthermore, because nodes that store data are inexpensive, it makes the whole DSS inexpensive when compared to storing data on a single storage device with the same capacity.

On the other hand, by the same virtue, the nodes are vulnerable to failures. Failures can occur due to many reasons. Prime among them are mechanical and electrical issues [6]. They may also occur due to network outages when there is a sudden influx of data flow in the network. Thus, it is essential to prevent data

[1] 1 exabyte equals 1000 petabytes, where one petabyte equals 1000 terabytes (TB), and $1\,\text{TB} = 10^{12}$ bytes.

loss from node failures.

A common way to achieve this is by adding redundancy into the system by encoding the stored data using an erasure correcting code. Addition of redundancy increases the storage overhead. For example, DSSs like Google File System (GFS) [7] and Hadoop Distributed File System (HDFS) [8] employ $(3, 1)$ repetition codes, which ensures that the system can tolerate two node failures but at the cost of large additional storage overhead. In simple words, every bit of data stored on the system incurs a penalty of storing two extra bits. As an alternative to GFS and HDFS, DSSs such as HDFS-Redundant Array of Independent Disks (RAID[2]) and Distributed RAID were introduced. They can tolerate at most 2 node failures while achieving significantly lower storage overhead [9]. On the downside, such systems cannot tolerate many node failures. Current DSSs employed by Google (GFS II) and Facebook use variants of Reed-Solomon (RS) codes to store data, which can tolerate multiple node failures [10]. These codes are a class of maximum distance separable (MDS) codes, which when used on a DSS, allow them to tolerate an optimal number of node failures for a given storage overhead.

Another issue that arises in DSSs is the *repair problem*. Nodes need to be repaired periodically to maintain the initial state of reliability of these systems. In order for DSSs to achieve high throughputs, it is important to have efficient repair of failed nodes. In other words, the node repair should involve as little data transfer across the DSS network as possible. Thus, an efficient DSS involves designing erasure correcting codes that not only balance reliability with storage overhead but should also perform efficient node repair. Although DSSs that use MDS codes have the best reliability for a given storage overhead, they are inefficient in repair. In contrast, local reconstruction codes (LRCs) [11, 12], minimum storage regenerating (MSR) codes [13, 14], and minimum bandwidth regenerating (MBR) codes [13, 15] are some state-of-the-art erasure correcting codes for DSSs that are also repair efficient.[3] A further way to improve data throughputs is to reduce the repair complexity, i.e., the number of operations required in the repair process. Generally speaking, most state-of-the-art codes require a high number of operations for repairing nodes.

The main purpose of this thesis is to study distributed storage techniques for storing large amounts of data. In particular, we study how to store large amounts of data efficiently, securely, and privately access them in such systems. In this thesis, we present a new class of codes for DSSs that have efficient repair and low repair complexity. These codes are constructed by concatenating two different classes of codes, where the first class of codes is a modified version of the classical MDS codes with the goal of providing resilience against node failures, while the second class of codes in conjunction with the first class provides efficient repair and low repair complexity. We go on to characterize their fault tolerance, storage overhead, efficiency in repair, and repair complexity.

With the increasing importance of DSSs in real-world applications, security of these systems is becoming more and more crucial. The issue is further compounded when data is stored on untrusted systems like peer-to-peer DSSs. Se-

[2]RAID is a separate class of storage technology that use repetition, single parity check, or Reed Solomon codes to achieve reliability. Besides, they are highly parallelized to ensure high throughputs.

[3]In a parallel line of work, schemes for efficient repair of RS codes have been proposed [16, 17].

curity for distributed storage is classified into data storage, access to data, movement of data, and management of data [18]. In this thesis, we focus on the first class: security for data storage. Currently, the industry achieves security by applying cryptographic protocols to the systems [19, 20]. However, the downside of such schemes is first, complex key management and secondly, a weak sense of security as it is dependent upon the computational power of the system [21]. As an antithesis to cryptographic protocols, in this dissertation we look at security from an information-theoretic perspective that provides security even if the system has infinite computational power. Furthermore, we assume an eavesdropper adversary that has access to the stored data on a subset of nodes of the DSS. The central idea to achieve security in this scenario is by encoding the data and random bits together and then storing them on the DSS [21–23]. This strong sense of security comes at the cost of larger storage overhead. Thus, there is an interest in determining the best possible efficiency and coding schemes that can provide security. With the aim of providing security and efficient repair of failed nodes in DSSs, the authors in [21] presented secure MSR codes and secure LRCs. Furthermore, they characterized the maximum achievable storage efficiency while maintaining the security of the data stored and showed that their schemes achieved this. Repairable fountain codes (RFCs) are another class of codes for DSSs. While providing efficient repair of failed nodes, unlike the codes above for DSSs, they can efficiently encode data over a large number of nodes and parallelize access of data from the nodes [24]. Since these are large codes, they can achieve higher storage efficiency and fault tolerance. In this thesis, we present secure RFCs that retain all the properties of RFCs while also providing security. The scheme is similar to the one presented in [21], but the proofs involved are a further generalization of the ones presented in [21, 22]. In particular, we provide a necessary and a sufficient condition for any code to achieve security, unlike [21, 22], which give only a sufficient condition.

Closely related to security is the notion of *privacy*, which is the ability to keep oneself or information about oneself from others. With the explosion of information in the world, it has gained significant traction in recent years [25]. DSSs are employed in data centers around the world which are owned by third party entities. Thus, in some cases, it is important to hide which data from the DSS is requested by the users from such parties. Consider the example where stock traders are trying to retrieve stock prices from a DSS that is hosted by a third party. In this case, hiding the identity of the requested stock is crucial. A natural question to ask is as follows. Is it possible for a user to retrieve data from storage devices (therefore, the 3rd party entities) without letting them know what the user asked? We refer to this as *private information retrieval* (PIR) and the schemes that achieve this as PIR protocols. Achieving PIR comes at the cost of lower communication efficiency, i.e., an increase in the cost of communication (upload and download) [26]. PIR protocols can be classified into computational PIR protocols and information-theoretic PIR protocols, where the difference lies in the assumption of the computational power of the adversaries (here, the storage devices). The former achieves privacy when the adversaries have limited computational power, while the latter provides privacy even when the adversaries have infinite computational power. In this thesis, we focus on information-theoretic PIR. Typ-

ically, the communication efficiency of a PIR protocol is given by its PIR rate, which is defined as the ratio of the amount of requested data and the amount of data that needs to be downloaded to ensure privacy.[4] Most works in the literature address PIR under the assumption that the data on the DSS is encoded using MDS codes. Under the assumption that no nodes collude, in [27] the authors provided the largest achievable PIR rate, referred to as MDS-PIR capacity, for any PIR protocol. As mentioned earlier, MDS codes perform inefficient repair of node failures.

In the thesis, we study PIR for the case where data is stored using an arbitrary linear code. We assume that some nodes in the DSS may act as spies, which can collaborate (the colluding case) or cannot collaborate (the noncolluding case) in order to determine the identity of the requested file. Subsequently, we provide 3 PIR protocols for such a scenario. Protocols 1 and 2 provide privacy for the noncolluding case. They differ in the fact that Protocol 1 requires the files stored to have a size that is exponential in the number of files, while Protocol 2 is independent of the file size, thus making it more practical. We are first to show that the MDS-PIR capacity is achievable even when the underlying storage code is a non-MDS code. In particular, we prove that important classes of non-MDS codes, namely cyclic codes, Reed-Muller (RM) codes, and distance-optimal LRCs, achieve the MDS-PIR capacity under Protocol 1. Assuming that the number of files in the system tends to infinity, for the same class of underlying storage codes, Protocol 2 achieves the MDS-PIR capacity asymptotically. Subsequently, we show that these rates are indeed equal to the maximum PIR rate achieved by these codes under any protocol. For such codes, we provide partial fundamental reasons for why this happens, and in doing so, we connect the fields of PIR and algebraic codes. For the remaining codes, we provide protocols that improve on the PIR rates provided by Protocols 1 and 2. Finally, Protocol 3 provides privacy when a subset of nodes in the DSS collude in order to obtain the identity of the requested file. To the best of our knowledge, for most parameters, our protocol provides the highest asymptotic PIR rates among all available protocols when a subset of nodes collude.

In the current age of information, parallel to the evolution of the storage technology there has also been an evolution of the cellular wireless technology standard from 2G at the start of the millennium to current 4G. As of 2017, 4G technology has been deployed in most countries[5] and the technology itself has matured. Furthermore, there is strong evidence that wireless data is exploding [28]. Accordingly, the future standard (referred to as 5G) must allow higher data rates and lower latencies, while being energy and cost-efficient [29]. Caching content at the edge of the wireless network is considered to be an attractive solution that could be implemented to achieve lower latencies through lower backhaul rates [30]. A future wireless network will most likely consist of densely populated small-cell base stations (SBSs) which can cache content. Users accessing this wireless network can download content from the neighboring SBSs, thus drastically reducing

[4] In the traditional information-theoretic sense, the size of the requested data is much larger than the size of the query, thus download dominates the upload. As a consequence, the upload cost is commonly neglected in the formulation of PIR rate.

[5] https://opensignal.com/reports/2017/11/state-of-lte

the backhaul usage, and hence the latency. Just like for DSSs, attaining privacy in such networks is important. In this thesis, we consider PIR in a wireless network where SBS cache content for a scenario where the SBSs can be adversaries. In light of this, we present a PIR scheme that is a generalization of Protocol 3. Essentially, this scheme assumes that files stored on the network can be encoded using codes with different code rates, which is in contrast to DSSs where all files are stored by encoding them using the same code. We also show that for the caching technique to be most effective it is best for the most popular files to be encoded using a single code, which is in contrast to what is the best when privacy is not considered. Furthermore, in most cases, it is seen that popular content caching, i.e., the most popular files are replicated and then cached on the SBSs, is optimal.

1.1 Organization

The thesis is an *omnibus* of published papers, or papers that are submitted for publication. It is divided into two parts where Part 1 deals with the introduction to the concepts in Papers I-V, while Part 2 contains these papers.

Part 1 is further divided into numerous chapters, where Chapters 2 and 3 are concerned with Paper I and Paper II, respectively, and Chapter 4 deals with concepts in Papers III-V. Chapter 2 is concerned with the evolution of codes for DSSs. It further describes how to store data on these systems, which has relevance to Papers I-IV. It also describes some of the important families of codes for DSSs. The chapter ends by briefly introducing distributed caching in wireless networks, and describing the system model that is relevant to Paper V. Chapter 3 describes the security model that is used in Paper II. In other words, it characterizes the adversary. It goes on to provide the brief idea on how to achieve information-theoretic security against such an adversary. The last chapter, Chapter 4, explains the PIR system model and proceeds to provide the main ideas used by different PIR protocols in the literature. It provides the intuition behind the construction of PIR protocols for DSSs that use arbitrary linear codes. Finally, it describes the PIR problem in the distributed caching scenario, and mentions the differences relative to PIR in DSSs.

1.2 Notations

Notations used in Part 1 of the thesis are listed as follows.

- We use lowercase bold letters, uppercase bold letters, and uppercase calligraphic letters to denote vectors, matrices, and sets, respectively. For example, $\boldsymbol{x}$ denotes a vector while $\boldsymbol{X}$ denotes a matrix, and $\mathcal{X}$ denotes a set.
- The operator $|\cdot|$ denotes the cardinality of a set.
- The operator $(\cdot)^\top$ denotes the transpose of a vector.
- The symbol $\mathbb{N}_a$ represents the set of natural numbers $\{1, \ldots, a\}$, while $\mathbb{N}_{a:b}$ represents the set of natural numbers $\{a, \ldots, b\}$, where $a < b$.

- Consider the column vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, then $(\boldsymbol{x}_1 \mid \cdots \mid \boldsymbol{x}_n)$ denotes the horizontal concatenation of the vectors. Similarly the horizontal concatenation of matrices $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ with the same row dimension is denoted as $(\boldsymbol{X}_1 \mid \cdots \mid \boldsymbol{X}_n)$.
- The calligraphic letter $\mathcal{C}$ denotes an $[n, k, d^{\mathcal{C}}_{\mathsf{min}}]$ linear code of dimension k, length n, and minimum distance $d^{\mathcal{C}}_{\mathsf{min}}$ over a Galois field GF(q), where q is a prime or a power of a prime. As a shorthand notation, we sometimes neglect $d^{\mathcal{C}}_{\mathsf{min}}$ and represent the code $\mathcal{C}$ with parameters $[n, k]$.
- We represent a submatrix of $\boldsymbol{X}$ that is restricted by the column set $\mathcal{J}$ and row set $\mathcal{I}$ as $\boldsymbol{X}|^{\mathcal{I}}_{\mathcal{J}}$. Since $\mathcal{C}$ can be seen as a codebook matrix, the shortened and punctured codes are denoted by $\mathcal{C}^{\mathcal{I}}_{\mathcal{J}}$, with column coordinates $\mathcal{J}$ and row coordinates $\mathcal{I}$.
- The function $\mathsf{H}(\cdot)$ represents the entropy of its argument. The function $\mathsf{I}(X;Y)$ represents the mutual information between random variables X and Y.

Notational Inconsistencies

The reader should be aware of the following notational inconsistencies occurring throughout the manuscript.

- In Paper II, the length, dimension, and minimum distance of a Gabidulin code are denoted by N, K, and D_{min}, respectively, whereas the respective parameters of the remaining codes in all papers are denoted by n, k, and $d^{\mathcal{C}}_{\mathsf{min}}$.
- In the same paper, q strictly represents a prime number, whereas in this thesis and remaining papers it denotes either a prime number or a prime number raised to some arbitrary integer exponent.
- β represents the subpacketization of a storage code in Paper I, while in the remaining papers and in this manuscript the same is denoted by α, and β represents the striping of the stored files.
- The entropy function is represented by $H(\cdot)$ in Paper II, while in this manuscript and the remaining papers it is represented as $\mathsf{H}(\cdot)$.
- The number of files in the system in Papers III and IV is denoted by f, while it is denoted by F in Paper V.
- The acronym "LRC" in Paper I represents a specific code family as defined in [12], whereas in Paper III it represents a class of codes that includes the aforementioned codes.
- The codes are represented as (n, k) in Papers I, II, and V, while in the remaining papers, they are either represented as $[n, k, d^{\mathcal{C}}_{\mathsf{min}}]$ or $[n, k]$.

Chapter 2

Distributed Storage Systems and Distributed Caching

The majority of this thesis deals with practical issues faced in DSSs, and therefore this chapter serves to enlighten the readers toward DSSs. It briefly describes the need for erasure coding in DSSs and the reasons why traditional codes are inefficient in such systems. It goes on to explain the system model, the state-of-the-art codes, and the various parameters used to evaluate their performance.

In particular, Section 2.1 explains the evolution of erasure correcting codes in DSSs and the motivation for going toward current state-of-the-art codes. Section 2.2 then describes the general system model that is used in Papers I-IV. It also describes the different parameters that are used to evaluate the performance of storage codes. Finally, in Section 2.3, we give a brief description of some popular classes of storage codes.

2.1 Coding in Distributed Storage

In coding theory, an $[n, k]$ erasure correcting code $\mathcal{C}$ over GF(q) is a mathematical construct that encodes k message symbols to obtain a codeword containing $n > k$ code symbols, where all mathematical operations occur over the finite field GF(q). The code has a code rate of $R^{\mathcal{C}} = k/n$, and a minimum distance of $d^{\mathcal{C}}_{\mathsf{min}}$. The former denotes the efficiency, while the latter relates to the erasure correcting capability of the code, which equates to $d^{\mathcal{C}}_{\mathsf{min}} - 1$. For given values of n and k, the largest minimum distance (ergo, the largest erasure correcting capability) is given by the Singleton bound, i.e., $d^{\mathcal{C}}_{\mathsf{min}} \leq n - k + 1$. The $[n, 1]$ repetition codes are the most trivial codes, where a single message symbol is replicated n times. Moreover, they achieve the Singleton bound. Motivated by the triviality of such codes, early DSSs such as GFS and HDFS used $[3, 1]$ repetition codes where each message symbol is replicated on $n = 3$ nodes. Such systems can tolerate at most two node failures. However, such a coding scheme is inefficient as the code rate is $1/3$. To counter this, recent DSSs such as GFS II and Quick File System by Google and Oracle, respectively, have started to adopt classical MDS codes [10]. In particular, these systems employ the $[9, 6]$ RS code that has code rate $2/3$. Such class of codes has the distinction that their minimum distances achieve the Singleton bound, hence achieving the best erasure correcting capability for a given code rate.

Traditionally, the n code symbols obtained after the encoding using an $[n, k]$ code are distributed among the n nodes. As a consequence, the erasure correcting capability translates to the number of node failures the DSS can tolerate. This allows the DSS to be reliable as long as the number of nodes failed is less than the number of node failures the code can tolerate. Thus, to ensure reliability over long periods, the systems should repair failed nodes such that the number of node failures does not exceed the tolerance level of the system. It may seem that MDS codes are the best codes for DSSs as they achieve the best fault tolerance/storage efficiency tradeoff. However, there is a caveat: the $n - k$ redundant symbols are a function of the k message symbols. As a result, the repair of any failed node requires downloading symbols from k nodes in the system, thus making repair costly. There are two aspects related to efficient repair, which are repair locality and repair bandwidth. The repair locality determines the number of nodes that need to be contacted in the repair of a failed node, while repair bandwidth is the number of bits that need to be transferred over the network in order to repair the failed node. MDS codes have locality equal to k and repair bandwidth of kv, where v is the symbol size in bits. This is in high contrast to the repetition code, where the repair locality and bandwidth are 1 and v, respectively.

During the repair of a failed node, individual nodes are not accessible for other network processes such as data download by users. Thus, a large repair locality means that a significantly large number of nodes are inaccessible during the repair, thereby affecting the data availability. On the other hand, larger repair bandwidth results in high network data traffic, which in turn makes the repair process dominate other network processes, hence reducing the performance of the DSS. Accordingly, in recent years, the research has been focused on finding erasure correcting codes for DSSs that are not only efficient in storage and can tolerate node failures, but allow to repair node failures efficiently as well.

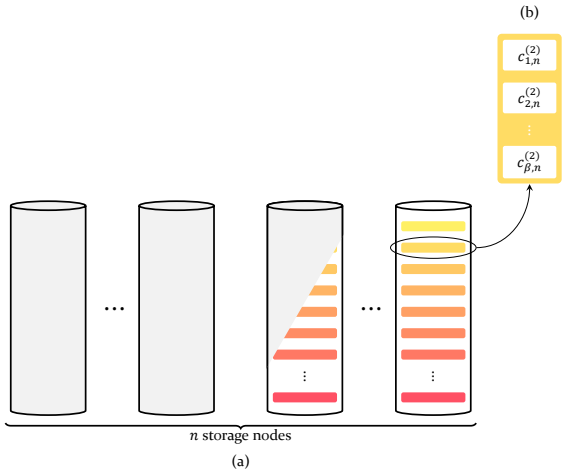

Figure 2.1: Illustration representing a typical DSS that stores f files.

2.2 System Model for Distributed Storage

We consider a DSS that stores f files $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(f)}$, where each file $\boldsymbol{X}^{(m)} = (x_{i,j}^{(m)})$, $m \in \mathbb{N}_f$, can be seen as a $\beta \times k$ matrix over GF($p^{\alpha\ell}$), with β, k, α, $\ell \in \mathbb{N}$, and p some prime number. Each file is encoded using a linear code as follows. Let $\boldsymbol{x}_i^{(m)} = (x_{i,1}^{(m)}, \ldots, x_{i,k}^{(m)})$, $i \in \mathbb{N}_\beta$, be a message vector corresponding to the i-th row of $\boldsymbol{X}^{(m)}$. Each $\boldsymbol{x}_i^{(m)}$ is encoded by an $[n, k]$ code $\mathcal{C}$ over GF(q) with $q \stackrel{\text{def}}{=} p^\alpha$, having *subpacketization* α, into a length-n codeword $\boldsymbol{c}_i^{(m)} = \left(c_{i,1}^{(m)}, \ldots, c_{i,n}^{(m)}\right)$, where $c_{i,j}^{(m)} \in$ GF(q^ℓ), $j \in \mathbb{N}_n$. For $\alpha = 1$, the code $\mathcal{C}$ is referred to as a *scalar* code. Otherwise, the code is called a *vector* code [31]. The βf generated codewords $\boldsymbol{c}_i^{(m)}$ are then arranged in the array $\boldsymbol{C} = \left((\boldsymbol{C}^{(1)})^\top | \ldots | (\boldsymbol{C}^{(f)})^\top\right)^\top$ of dimensions $\beta f \times n$, where $\boldsymbol{C}^{(m)} = \left((\boldsymbol{c}_1^{(m)})^\top | \ldots | (\boldsymbol{c}_\beta^{(m)})^\top\right)^\top$ for $m \in \mathbb{N}_f$. For a given column j of $\boldsymbol{C}$, we denote the vector $\left(c_{1,j}^{(m)}, \ldots, c_{\beta,j}^{(m)}\right)$ as a coded chunk pertaining to file $\boldsymbol{X}^{(m)}$. The f coded chunks in column j are stored on the j-th storage node as shown in Fig. 2.1(a), and Fig. 2.1(b) shows a coded chunk corresponding to the 2nd file in the n-th node, which consists of β code symbols, $c_{i,n}^{(2)}$, $i \in \mathbb{N}_\beta$. In case the $[n, k]$ code $\mathcal{C}$ is systematic, the nodes that store the systematic code symbols are referred to as systematic nodes, while the nodes storing redundant symbols are referred to as parity nodes.

We consider specific instances of the above system model in Papers I and II in contrast to Papers III and IV where we consider the exact model. In Paper I, we keep $\beta = 1$ and $\alpha = k$, whereas in Paper II, we keep $\beta = 1$ and $\alpha = 1$. In the following, for DSSs that use the code $\mathcal{C}$, we briefly explain the code parameters that parameterize the system performance.

1. **Fault tolerance.** The fault tolerance of a code determines the number of node failures the corresponding DSS can tolerate at a time. It depends on

the minimum distance of the code and is given as $d^{\mathcal{C}}_{\min} - 1$.

2. **Storage overhead.** The storage overhead of a DSS parameterizes the storage efficiency of the system. In particular, it is the ratio of total coded data and the message size and is given as $n/k = 1/R^{\mathcal{C}}$. If the system stores uncoded data, then it has a storage overhead of one. Since the storage overhead is just the inverse of the code rate of the code used, throughout the thesis, we alternate between storage overhead and code rate when talking about the efficiency of the system.

3. **Repair locality.** It is a measure of the repair efficiency that parameterizes the number of nodes that the DSS should connect to in order for it to repair a single failed node. If $\mathcal{C}$ is an MDS code, then the repair locality of the DSS is k, on the other hand, if the code is a repetition code, then it has locality 1.

4. **Repair bandwidth.** The repair bandwidth determines the number of bits that are communicated by the DSS in order to repair a failed node. In particular, let ν be the size of a code symbol in bits and λ the total number of code symbols needed to repair a single code symbol in the failed node, then the repair bandwidth of the system equals $\nu\lambda\alpha$ bits. For an efficient repair, it is necessary that the repair bandwidth is small.

5. **Complexity.** In this thesis, the term complexity determines the number of elementary operations required to complete a process. In computer systems, the complexity of a process is directly proportional to the hardware costs. As such for DSSs it is important to have low complexity. In DSSs, there are two important processes–encoding and repair. The complexity of the respective processes is referred to as repair and encoding complexity. Formally, repair complexity is defined as the number of elementary operations required in order to repair a failed node, whereas encoding complexity is the number of elementary operations required to encode a file. Mathematically, these processes occur over GF(q), and thus addition and multiplication require $O(\nu)$ and $O(\nu^2)$ elementary operations, respectively,[1] where $\nu = \alpha\ell\lceil\log_2 p\rceil$.

2.3 Codes for Distributed Storage

In the past decade, there have been a plethora of codes designed specifically for DSSs. In this section, we discuss erasure correcting codes that are relevant to this thesis. We will be brief and refer the reader to the cited papers for their detailed construction.

[1]It should be noted that the complexity of multiplication is quite pessimistic. However, for the sake of simplicity, we assume it to be $O(\nu^2)$. When the field is GF(2^ν) there exist algorithms such as the Karatsuba-Ofman algorithm [32, 33] and the Fast Fourier Transform [34–36] that lower the complexity to $O(\nu^{\log_2 3})$ and $O(\nu \log_2 \nu)$, respectively.

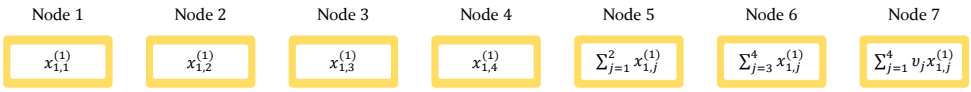

Figure 2.2: A DSS where $f = \beta = 1$ that uses a $[7, 4, 3]$ erasure correcting code with $\alpha = 1$ over GF(q) that is an example of a $(2, 2)$ information locality code, where $v_j \in$ GF$(q), j \in \mathbb{N}_4$.

2.3.1 Codes With Locality

These codes are the ones that focus on reducing the repair locality. They can be classified into two subclasses: (r, δ) information locality codes and (r, δ) code symbol locality codes [37]. The former class of codes achieves a repair locality of r when at most $\delta - 1$ systematic nodes have failed, while the latter class can achieve the same for $\delta - 1$ arbitrary node failures. The thesis is primarily focused on (r, δ) information locality codes, which leads us to formally defining them as follows.

Definition 1 ((r, δ) information locality code [37, Def. 2])**.** *An $[n, k]$ code is said to be an (r, δ) information locality code if there exist L_c punctured codes $\mathcal{C}_j \stackrel{\text{def}}{=} \mathcal{C}|_{\mathcal{S}_j}$ of $\mathcal{C}$ with column coordinate set $\mathcal{S}_j \subset \mathbb{N}_n$ for $j \in \mathbb{N}_{L_c}$. Furthermore, $\{\mathcal{C}|_{\mathcal{S}_j}\}_{j \in \mathbb{N}_{L_c}}$ must satisfy the following conditions:*

1. $|\mathcal{S}_j| \leq r + \delta - 1, \forall\, j \in \mathbb{N}_{L_c}$,
2. $d_{\min}^{\mathcal{C}_j} \geq \delta, \forall\, j \in \mathbb{N}_{L_c}$, *and*
3. $\operatorname{rank}\left(\boldsymbol{G}|_{\cup_j \mathcal{S}_j}\right) = k$.

The ensuing codes have $d_{\min}^{\mathcal{C}} \leq n - k + 1 - (\lceil k/r \rceil - 1)(\delta - 1)$.

In other words, Definition 1 says that there are L_c local codes in $\mathcal{C}$ each having a block length of at most $r + \delta - 1$, a minimum distance of at least δ, and the union of all local codes forms an information set. Examples of such codes out of many in literature are Pyramid codes [11], LRCs [12], and locally repairable codes [4]. These codes fall under the class of $(r, 2)$ information locality codes and are known to be distance optimal, i.e, for a given locality r they achieve the largest minimum distance. Notably, LRCs and the locally repairable codes in [4], were implemented by Microsoft and Facebook on their Windows Azure and HDFS-XORBAS systems. Such codes achieve low repair locality by ensuring that a subset of parity symbols are functions of a small number of message symbols.

Fig. 2.2 depicts a toy example of a DSS where $f = \alpha = \beta = 1$. The file $\boldsymbol{X}^{(1)}$ is encoded using a $[7, 4, 3]$ code that is a $(2, 2)$ information locality code and then stored on seven nodes. Thus, the code has a repair locality of 2. For example, consider that Node 1 has failed. Then, it can be repaired by downloading $x_{1,2}^{(1)}$ and $\sum_{j=1}^{2} x_{1,j}^{(1)}$. On the other hand, the codes presented by the authors in [38, 39], and RFCs [24] are examples of code symbol locality codes. Necessarily, in such codes, each code symbol is a function of at most r other symbols. As RFCs are extensively used in Paper II, we briefly describe their construction below.

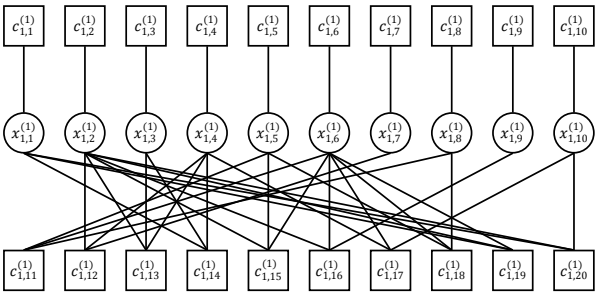

Figure 2.3: A [20, 10] RFC over GF(q) with $\alpha = 1$.

Repairable Fountain Codes

RFCs are scalar codes, and therefore they have $\alpha = 1$. An $[n, k, d^{\mathcal{C}}_{\min}]$ systematic RFC over GF(q) encodes a message $\boldsymbol{x}_i^{(m)} = (x_{i,1}^{(m)}, \ldots, x_{i,k}^{(m)})$ into a codeword $\boldsymbol{c}_i^{(m)} = (c_{i,1}^{(m)}, \ldots, c_{i,n}^{(m)})$, where $i \in \mathbb{N}_\beta$. Because the codes are systematic, we have $c_{i,j}^{(m)} = x_{i,j}^{(m)}, j \in \mathbb{N}_k$. For a fixed i, the parity symbols $c_{i,j}^{(m)}$, $j \in \mathbb{N}_{k+1:n}$, are constructed according to the following three-step procedure.

1. Successively select $\xi = O(\log k)$ message symbols $x_{i,j}^{(m)}$ independently and uniformly at random with replacement.
2. For each of the ξ message symbols, a coefficient is drawn uniformly at random from GF(q).
3. The parity symbol is then obtained as the linear combination of the ξ chosen message symbols, weighted by the corresponding coefficients.

Each of the n code symbols is stored in a different storage node. From the code construction, each parity symbol is a weighted sum of at most ξ message symbols. A parity symbol and the corresponding (at most) ξ message symbols are referred to as a local group. The existence of local groups is a hallmark of any erasure correcting code having low locality. Unlike most codes with locality, which have only disjoint local groups, RFCs also have overlapping local groups [24]. Furthermore, for each systematic symbol there exist a number of disjoint local groups from which it can be reconstructed. This allows multiple parallel reads of the systematic symbol, accessing the disjoint local groups. When a storage node fails, it is repaired from one of its local groups. This requires the download of at most ξ symbols (from the other at most ξ nodes of the local group). Thus, RFCs have low locality, ξ, and their repair bandwidth is $\xi \log p$. Also, RFCs are near MDS codes.

Consider a DSS with $f = \beta = 1$, and where the file that is to be stored is encoded using a [20, 10] RFC. Fig. 2.3 shows the encoding of this file using a bipartite graph, where circles represent the symbols of the file $\boldsymbol{X}^{(1)}$ and the squares represent the code symbols. Each code symbol is stored on an individual storage node and is a weighted linear combination of its neighbors as shown in the figure. In particular, each parity symbol is a sum of at most $\xi = 3$ message symbols.

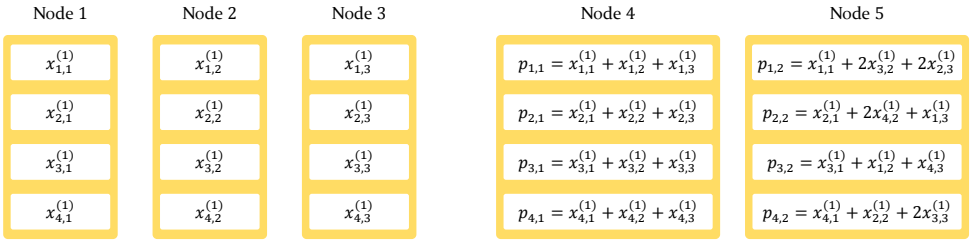

Figure 2.4: A $[5, 3, 3]$ Zigzag code over GF(3^4) with $f = \beta = 1$ and $\alpha = 4$. With some abuse of notation we represent the code as a $[20, 12]$ array code over GF(3) where we split each code symbol $c_{1,j}^{(1)} = x_{i,j}^{(1)} = (x_{1,j}, \ldots, x_{4,j})^{\mathsf{T}}, j \in \mathbb{N}_k$, and $x_{i,j} \in \text{GF}(3), i \in \mathbb{N}_\alpha$.

2.3.2 Regenerating Codes

In the seminal paper [13], Dimakis *et al.* presented regenerating codes, a class of codes that significantly reduce the repair bandwidth. Additionally, they showed that there is a fundamental tradeoff between storage and repair bandwidth. For a given r, the number of bits downloaded per contacted node ψ, and code parameters n and k, the upper bound on the file size M is given as

$$M \leq \sum_{i=0}^{k-1} \min\{(r-i)\psi, \alpha\}. \tag{2.1}$$

By fixing M, r, and k in (2.1) and plotting the resulting repair bandwidth as a function of α , i.e., the number of symbols stored per node, gives us the tradeoff. There exist two extremum points: the MSR point and the MBR point. As the names suggest, the MSR point is the point where α is minimum, while the MBR point is at the other end of the spectrum, where ψ is minimized, i.e., the repair bandwidth is minimized. Codes that achieve these two points are referred to as MSR and MBR codes, respectively. Codes such as Zigzag [14], Product matrix MSR [15], and minimum disk input/output repairable [40] codes are examples of MSR codes. Such codes are also MDS codes. Fractional repetition [41] and Product matrix MBR [15] codes are examples of MBR codes.

Fig. 2.4 illustrates a $[5, 3, 3]$ Zigzag code. This code achieves a repair bandwidth of 16 bits for the repair of any systematic symbol. For the same file size, an MDS code has a repair bandwidth of 24 bits. Consider the repair of Node 2, then the symbols $x_{1,2}^{(1)}$ and $x_{2,2}^{(1)}$ are recovered by downloading $x_{1,1}^{(1)}, x_{2,1}^{(1)}, x_{1,3}^{(1)}, x_{2,3}^{(1)}, p_{1,1}$, and $p_{2,1}$ from Nodes 1, 3, and 4. The last two symbols $x_{3,2}^{(1)}$ and $x_{4,2}^{(1)}$ can be recovered by downloading just $p_{1,2}$ and $p_{2,2}$ from Node 5 as the other required symbols are already downloaded during the recovery of previous symbols. In total, 8 symbols of size 2 bits are downloaded. Thus, in total 16 bits are downloaded.

2.3.3 Codes From the Piggybacking Framework

The piggybacking framework was first introduced in [42] and is applied over a pre-existing erasure correcting code. Most commonly, the pre-existing code is an

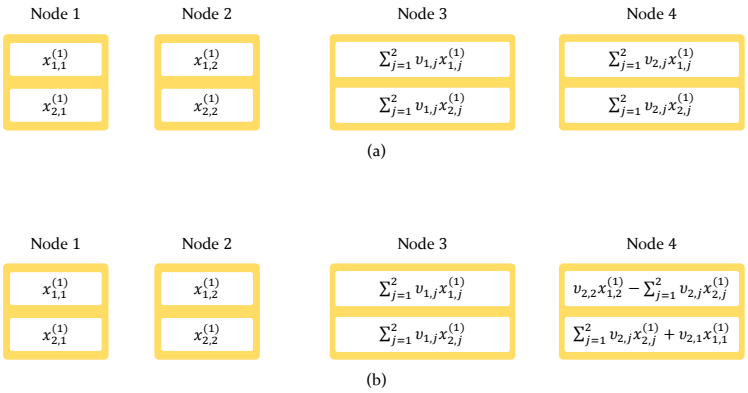

Figure 2.5: DSS with $f = \beta = 1$ that uses $[4, 2, 3]$ erasure correcting codes over GF(p^2) with $\alpha = 2$ and $v_{1,j}, v_{2,j} \in$ GF(p). (a) Classical MDS code. (b) Piggybacking framework applied to the MDS code.

MDS code. The concept involves taking carefully chosen linear combinations of message symbols, referred to as *piggybacks*, and then add them to the parity symbols of the code. This allows for lower repair bandwidth at the expense of higher encoding and repair complexity. A rule of thumb is that both the encoding and repair complexity are larger than those of the pre-existing code, while the repair bandwidth is smaller than that of the pre-existing code. The codes presented in [42–44] are some examples that use the piggybacking framework on top of an MDS code, and have fault tolerance equal to that of the pre-existing MDS code.

Fig. 2.5(b) illustrates a $[4, 2, 3]$ code from [42] that is able to reduce the repair bandwidth of the systematic nodes. It starts by taking the $[4, 2, 3]$ MDS code in Fig. 2.5(a) and then adding piggybacks to it. In the above example, the piggybacks $v_{2,1}x_{1,1}^{(1)}$ and $\sum_{j=1}^{2} v_{2,j}x_{2,j}^{(1)} + v_{2,1}x_{1,1}^{(1)}$ are added and subtracted, respectively to the parity symbols in Node 4. This modifies the code to the piggybacked code shown in Fig. 2.5(b). Consider the repair of Node 1. The repair procedure is as follows: recover $x_{2,1}^{(1)}$ by downloading $x_{2,2}^{(1)}$ and $\sum_{j=1}^{2} v_{1,j}x_{2,j}^{(1)}$ from Nodes 2 and 3. Then, download $\sum_{j=1}^{2} v_{2,j}x_{2,j}^{(1)} + v_{2,1}x_{1,1}^{(1)}$ to recover $x_{1,1}^{(1)}$ as the symbols $x_{2,j}^{(1)}, j \in \mathbb{N}_2$, are already known to the user. The DSS downloads 4 symbols for the repair of Node 1. The repair of Node 2 also requires downloading the same number of symbols. Thus, for systematic nodes, the code has a repair bandwidth of $4\lceil \log_2 p \rceil$ bits.

2.4 Coding in Distributed Caching for Wireless Networks

Content delivery has evolved from the traditional broadcasting scenario where the same content was broadcasted to all users to a content-on-demand scenario characterized by the fact that users place requests in a highly asynchronous manner. This, in addition to the fact that in recent years wireless networks have seen an exponential increase in the number of users, has lead to an explosion of the wireless traffic. To cope with this, a technique referred to as distributed caching

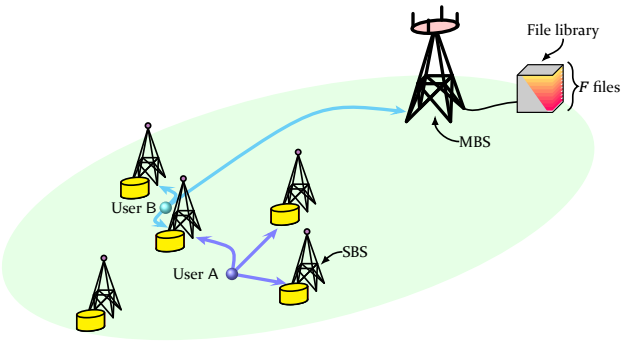

Figure 2.6: A wireless network for content delivery consisting of an MBS and five SBSs. Users download files from a library of F files. The MBS has access to the library through a backhaul link. Some files are also cached at SBSs using a $(5, 3)$ MDS code. User A retrieves a cached file from the three SBSs within range. User B retrieves a fraction $2/3$ of a cached file from the two SBSs within range and the remaining fraction from the MBS.

has been proposed. The concept of distributed caching is similar to that of distributed storage, where content is cached closer to the end users (in densely deployed SBSs [30, 45, 46] or directly in the mobile devices [46–48]) in a distributed manner. Similar to distributed storage, error correcting codes can be used to make caching more efficient. However, unlike distributed storage where the codes are used for storage, here coding reduces the latency in the content delivery to the users, as well as the backhaul traffic [30]. In this thesis, we consider the distributed coded caching scenario where content is cached on SBSs.

In a distributed coded caching scenario, each file (or piece of data stored at the macro base station (MBS)) is fragmented into k packets, depending on its popularity, and then encoded using an $[n, k]$ erasure correcting code to obtain n coded packets. These are then stored across a subset of n SBSs. A user who wishes to download a file downloads a fraction of it from neighboring SBSs, while the remaining fraction is downloaded from the MBS. The easiest way to achieve the benefits of caching is to perform popular content caching, which amounts to cache a copy of the most popular files on the SBSs. This is equivalent to using a repetition code to cache the content. The authors of [49, 50] were one of the first to show that caching content on wireless networks is beneficial. In addition, they showed that popular content caching is suboptimal. Later on, in [30, 45] it was shown that using MDS codes can significantly improve the performance of caching systems. In particular, [45] showed that using MDS codes in distributed caching reduces the download delay, and [30] showed that it reduces the amount of data downloaded from an MBS.

2.5 System Model for Distributed Caching

We consider a cellular network where a macro-cell is served by an MBS. Mobile users wish to download files from a library of F files that is always available at the

MBS through a backhaul link. We assume all files of equal size.[2] In particular, each file consists of βL bits and is represented by a $\beta \times L$ matrix $\boldsymbol{X}^{(i)}$,

$$\boldsymbol{X}^{(i)} = \begin{pmatrix} \tilde{\boldsymbol{x}}_1^{(i)} \\ \vdots \\ \tilde{\boldsymbol{x}}_\beta^{(i)} \end{pmatrix}$$

where upperindex $i = 1, \dots, F$ is the file index. Therefore, each file can be seen as divided into β stripes $\tilde{\boldsymbol{x}}_1^{(i)}, \dots, \tilde{\boldsymbol{x}}_\beta^{(i)}$ of L bits each. The file library has popularity distribution $\boldsymbol{p} = (p_1, \dots, p_F)$, where file $\boldsymbol{X}^{(i)}$ is requested with probability p_i. We also assume that N_{SBS} SBSs are deployed to serve requests and offload traffic from the MBS whenever possible. To this purpose, each SBS has a cache size equivalent to M files. The considered scenario is depicted in Fig. 2.6.

2.5.1 Content Placement

File $\boldsymbol{X}^{(i)}$ is partitioned into βk_i packets of size L/k_i bits and encoded before being cached in the SBSs. In particular, each packet is mapped onto a symbol of the field $\mathrm{GF}(q^{\delta_i})$, with $\delta_i \geq \frac{L}{k_i \log_2 q}$. For simplicity, we assume that $\frac{L}{k_i \log_2 q}$ is integer and set $\delta_i = \frac{L}{k_i \log_2 q}$. Thus, stripe $\tilde{\boldsymbol{x}}_a^{(i)}$ can be equivalently represented by a stripe $\boldsymbol{x}_a^{(i)}$, $a \in \mathbb{N}_\beta$, of symbols over $\mathrm{GF}(q^{\delta_i})$. Each stripe $\boldsymbol{x}_a^{(i)}$ is then encoded using an (N_{SBS}, k_i) MDS code $\mathcal{C}_i$ over $\mathrm{GF}(q)$ into a codeword $\boldsymbol{c}_a^{(i)} = (c_{a,1}^{(i)}, \dots, c_{a,N_{\mathsf{SBS}}}^{(i)})$, where code symbols $c_{a,j}^{(i)}$, $j \in \mathbb{N}_{N_{\mathsf{SBS}}}$, are over $\mathrm{GF}(q^{\delta_i})$.

The encoded file can be represented by a $\beta \times N_{\mathsf{SBS}}$ matrix $\boldsymbol{C}^{(i)} = (c_{a,j}^{(i)})$. Code symbols $c_{a,j}^{(i)}$, are then stored in the j-th SBS (the ordering is unimportant). Thus, for each file $\boldsymbol{X}^{(i)}$, each SBS caches one coded symbol of each stripe of the file, i.e., a fraction $\mu_i = 1/k_i$ of the i-th file. As $k_i \in \mathbb{N}_{N_{\mathsf{SBS}}}$,

$$\mu_i \in \mathcal{M}_{\mathsf{noPIR}} \triangleq \{0, 1/N_{\mathsf{SBS}}, 1/(N_{\mathsf{SBS}} - 1), \dots, 1/2, 1\}, \tag{2.2}$$

where $\mu_i = 0$ implies that file $\boldsymbol{X}^{(i)}$ is not cached.

Since each SBS can cache the equivalent of M files, the μ_i's must satisfy

$$\sum_{i=1}^{F} \mu_i \leq M.$$

We define the vector $\boldsymbol{\mu} = (\mu_1, \dots, \mu_F)$ and refer to it as the *content placement*. Also, we denote by $\mathcal{C}_{\mathsf{MDS}}^{\boldsymbol{\mu}}$ the caching scheme that uses MDS codes $\{\mathcal{C}_i\}$ according to the content placement $\boldsymbol{\mu}$.

[2] Assuming files of equal size is without loss of generality, since content can always be divided into chunks of equal size.

2.5.2 File Request

Mobile devices request files according to the popularity distribution $\boldsymbol{p} = (p_1, \ldots, p_F)$. Without loss of generality, we assume $p_1 \geq p_2 \geq \ldots \geq p_F$. The user request is initially served by the SBSs within communication range. We denote by γ_b the probability that the user is served by b SBSs and define $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{N_{\mathrm{SBS}}})$. If the user is not able to completely retrieve $\boldsymbol{X}^{(i)}$ from the SBSs, the additional required symbols are fetched from the MBS. Using the terminology in [30], the average fraction of files that are downloaded from the MBS is referred to as the backhaul rate, denoted by R_{bh}, and defined as

$$\mathsf{R}_{\mathsf{bh}} \triangleq \frac{\text{average no. of bits downloaded from the MBS}}{\beta L}.$$

Note that for the case of no caching $\mathsf{R}_{\mathsf{bh}} = 1$.

As in [30, 45, 47, 48], we assume that the communication is error free.

Chapter 3

Security

This chapter serves as the introduction to the notion of information-theoretic security in DSSs. Then, it goes on to define and characterize the adversary assumed in this thesis. Finally, it briefly explains the idea of how to achieve security against such an adversary.

Specifically, Section 3.1 serves to introduce the concept of security in DSSs. In Section 3.2, the system model for security is described. In particular, it defines and characterizes the adversary in detail. The last section, Section 3.3, provides a brief idea of how security is achieved in this scenario.

3.1 Introduction

Security in DSSs is analyzed through their resilience towards active or passive attacks [51, 52]. Active attacks are those attacks where the adversary actively injects errors into the stored data or modifies it, whereas passive attacks include an adversary who eavesdrops the stored or transmitted data. There are two ways to achieve security against such attacks: the cryptographic and the information-theoretic approach. For DSSs, the cryptographic approach entails key management, which can become complicated and cumbersome.

On the contrary, the information-theoretic approach involves coding schemes that are easier to handle. In DSSs, coding schemes are used to ensure reliability against node failures. Combined with the fact that the information-theoretic approach has a stronger sense of security, it is natural to look for information-theoretic security solutions for DSSs. In particular, in this thesis, we consider information-theoretic solutions against a passive attack adversary (from hereon referred to as the eavesdropper).

The authors of [23] were the first to consider the security aspect of DSSs. They considered adversaries which performed either an active attack, a passive attack, or a combination of the two. In particular, for the passive attack, they considered an eavesdropper model where the eavesdropper has access to at most $\ell_1 < k$ arbitrary nodes. They derived the *secrecy capacity*, i.e., the maximum size of the file such that security against the eavesdropper is maintained, at the MBR point. Furthermore, they provided constructions of secure MBR codes. These codes are MBR codes with the added benefit of being secure against the eavesdropper. In [22], the authors generalized the eavesdropper model, where in addition to having access to the data in ℓ_1 nodes, the eavesdropper has access to the data downloaded in the repair of ℓ_2 distinct nodes where $\ell_1 + \ell_2 < k$. They presented secure MBR and MSR code constructions that allowed the DSS to achieve security against the eavesdropper. For the same eavesdropper model, in [21], the authors presented the secrecy capacity at the MSR point for linear schemes, and an upper bound on the maximum file size that can be stored in the DSS using distance-optimal LRCs such that it is secure against the eavesdropper. As such, they also provided secure MSR codes that achieved the secrecy capacity, and secure LRCs that can store files having file size equal to the upper bound they derived. Such codes achieve efficient repair while being secure against the eavesdropper.

3.2 Eavesdropper Model

In this thesis (and more particularly in Paper II), we consider the (ℓ_1, ℓ_2) eavesdropper model [21, 22], where the eavesdropper can passively observe, but not modify, the content of $\ell = \ell_1 + \ell_2 < k$ storage nodes. Out of the ℓ nodes, the eavesdropper can observe the symbols stored in a subset of ℓ_1 storage nodes, which we denote by $\mathcal{S}_1$ ($|\mathcal{S}_1| = \ell_1$). Furthermore, it can observe the data downloaded during the repair of a subset of ℓ_2 storage nodes, denoted by $\mathcal{S}_2$ ($|\mathcal{S}_2| = \ell_2$), where $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. This model is relevant in the scenario where nodes are located at different geographical locations. Peer-to-peer storage systems are exam-

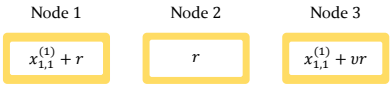

Figure 3.1: Illustration of an information-theoretic secure scheme for a DSS with $f = \beta = 1$. Here $r \in \mathrm{GF}(q)$ is the key and it is chosen uniformly at random from $\mathrm{GF}(q)$, and $v \in \mathrm{GF}(q)$.

ples of such DSSs [22]. We denote the subset of storage nodes from which data is downloaded to repair storage nodes in $\mathcal{S}_2$ by $\mathcal{S}_\mathrm{d}$. We will refer to the symbols $\boldsymbol{e}$ the eavesdropper obtains as the *eavesdropped symbols*. We also assume that the eavesdropper has perfect knowledge of the erasure correcting code used for encoding. In the following, we formally define when a DSS is secure against the aforementioned eavesdropper model.

Definition 2 ([21, 22])**.** *Let $\boldsymbol{e}$ be the vector of eavesdropped symbols that the eavesdropper obtains from the storage nodes in $\mathcal{S}_1 \cup \mathcal{S}_d$. A DSS storing a message $\boldsymbol{x}$ (possibly encoded by an erasure correcting code) is said to be* completely secure *against an (ℓ_1, ℓ_2) eavesdropper if the mutual information between the message and the eavesdropped symbols is zero, i.e., $\mathrm{I}(\boldsymbol{x}; \boldsymbol{e}) = 0$.*

3.3 Basic Principle

Any information-theoretic solution involves coding of the file and the key (which is a random sequence of symbols that are uniformly picked at random from a Galois field). Thus, in essence the security comes at the cost of larger storage overhead. Fig. 3.1 illustrates this. The scheme uses a $[3, 2]$ code $\mathcal{C}$ of rate $R^{\mathcal{C}} = 2/3$, which encodes a modified message $(x_{1,1}^{(1)} + r, r)$. Notice that the storage overhead increases from $3/2$ to 3. This simple scheme ensures security against an $(\ell_1 = 1, \ell_2 = 0)$ eavesdropper, while the retrieval of the file securely requires downloading symbols from any 2 nodes. For security against larger values of ℓ_1 and ℓ_2, one must look towards nontrivial solutions.

3.3.1 Shamir's Secret Sharing Scheme

Shamir provided one nontrivial solution in [53] that can be adapted to DSSs (more specifically, to DSSs described in this thesis). In the following, we describe the scheme in [53].

Consider that the DSS has to store the file $z \in \mathrm{GF}(q)$ securely. The user chooses a polynomial $q(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}$ of degree $k - 1$, where $a_0 = z$ and $a_i \in \mathrm{GF}(q), i \in \mathbb{N}_{k-1}$, is chosen uniformly at random. It then evaluates the polynomial at n points that are then stored on n nodes. In particular, the j-th node stores $q(x_j), x_j \in \mathrm{GF}(q)$. Given any subset of these k evaluations, the authenticated user can recover z by polynomial interpolation. More importantly, an adversary who has access to at most $k - 1$ evaluations cannot recover the file z. However, implementing such a solution on DSSs leads to inefficient repair of

nodes as one needs to download data from k nodes. So the question arises, is it possible to construct a secure scheme that allows efficient repair?

Consider a concatenated coding scheme (consisting of an outer and an inner code), where the code symbols from the inner code are stored on n storage nodes. Then, the repair of any storage node depends on the inner code. Furthermore, notice that in the secret sharing scheme, the security is achieved using the evaluation of polynomials. Therefore, the concatenation scheme can be made secure if the code symbols of the inner code are evaluations of a polynomial. To this end, we assign two criteria: 1) The outer code is an evaluation of a linearized polynomial, and 2) the inner code is a repair-efficient code for a DSS. We construct a concatenation scheme where the outer code is a Gabidulin code [54] as it meets criterion 1), and the inner code is either an LRC or an MSR code. In [21], the authors used this concatenation scheme to achieve security against an (ℓ_1, ℓ_2) eavesdropper. Motivated by the advantages of RFCs presented in Section 2.3, in Paper II we use the same concatenation scheme albeit with an inner RFC. As a result, in the following subsection, we present the construction of an $[N, K, D^{\mathcal{C}}_{\mathsf{min}}]$ Gabidulin code $\mathcal{C}$.

3.3.2 Gabidulin Codes

Gabidulin codes are a class of rank codes [54]. An $[N, K, D^{\mathcal{C}}_{\mathsf{min}}]$ Gabidulin code $\mathcal{C}$ (over GF(p^α), $\alpha \in \mathbb{N}$) of length N, dimension K, and minimum rank distance $D^{\mathcal{C}}_{\mathsf{min}}$, can correct up to $D^{\mathcal{C}}_{\mathsf{min}} - 1$ rank erasures. Gabidulin codes are maximum rank distance codes, i.e., they achieve the Singleton bound, $D^{\mathcal{C}}_{\mathsf{min}} \leq N - K + 1$, and are obtained by evaluations of polynomials. More specifically, Gabidulin codes use linearized polynomials.

Definition 3. *A linearized polynomial $f(y)$ of degree $t > 0$ over GF(p^α) has the form*

$$f(y) = \sum_{i=0}^{t} a_i y^{p^i},$$

where $a_i \in GF(p^\alpha)$ and $a_t \neq 0$.

A message $\boldsymbol{x} = (x_1, \ldots, x_K)$ is encoded using an (N, K) Gabidulin code as follows.

1. Construct a polynomial $f(y) = \sum_{i=1}^{K} x_i y^{p^{i-1}}$.
2. Evaluate $f(y)$ at N linearly independent (over GF(p)) points $\{y_1, \ldots, y_N\} \subset$ GF(p^α) to obtain a codeword $(f(y_1), \ldots, f(y_N))$.

Decoding proceeds as follows.

1. Obtain any K evaluations at K linearly independent (over GF(p)) points. Otherwise, decoding fails.
2. Perform polynomial interpolation on the K evaluations and recover the original message $\boldsymbol{x}$ by solving a system of linear equations.

Chapter 4

Private Information Retrieval

In this chapter, we introduce the concept of information-theoretic PIR in DSSs. The PIR problem in DSSs refers to the problem of downloading data from the nodes in a DSS without letting them know the identity of the requested file. The chapter starts by bringing to attention the evolution of PIR schemes that allow the user to achieve PIR. It then provides a system model that characterizes the adversary and the intuition behind the PIR schemes in the literature. Subsequently, it explains the PIR problem in distributed caching, and how it is different from the DSS model.

The chapter is divided into four sections. Section 4.1 provides the background that traces the evolution of PIR schemes. Section 4.2 then goes on to describe the privacy model that is used in Papers III and IV, and provides information-theoretic conditions for any scheme (or protocol) to achieve PIR. In Section 4.3, a brief intuition behind PIR protocols is provided where it is assumed that the DSS stores data using MDS codes. Section 4.4 discusses the need to provide PIR protocols for DSSs that store data using an arbitrary linear code. Furthermore, it provides a brief glimpse into the details of the construction of such protocols. Finally, Section 4.5 introduces the PIR problem to distributed caching.

4.1 Introduction

Chor *et al.* [26, 55] first introduced the so-called PIR problem in the theoretical computer science literature. They introduced the concept of an n-server PIR protocol, where it was assumed that the (binary) data was replicated on n nodes (or servers). From a coding theory perspective, it means that the PIR protocol was conditioned on the fact that the DSS uses an $[n, 1, n]$ storage code. Furthermore, they assumed that none of the nodes collude to obtain the identity of the file. The efficiency of the protocol was judged by the protocol's communication efficiency that accounted for the upload and the download cost. Subsequently, for the same storage model [56–58] presented PIR protocols that further improved the communication cost. In [59], the authors presented a PIR protocol where data was coded and then stored on the n storage nodes.

With the rise in popularity of DSSs and the knowledge that coding provides benefits, the study of PIR protocols has attracted the information theory community. Most information-theoretic works on PIR assume that the size of the data is significantly larger than the sizes of the queries sent to the nodes of the DSS. Therefore, the download cost dominates the upload cost, and it determines the efficiency of a PIR protocol. The efficiency is characterized by the PIR rate, which is defined as the ratio of the size of the requested data and the amount of data downloaded by the protocol, where higher PIR rate means better efficiency.

The authors of [60] were the first to introduce PIR schemes for DSSs, where data is stored using two explicitly designed codes. In [61], the authors presented an upper bound on the PIR rate for a particular class of PIR protocols. In [62], the authors presented PIR codes that when used with a traditional n-server PIR protocol, allows the DSS to achieve the PIR property. These codes allow the DSS to achieve low storage overhead, and at the same time allow for the resulting PIR protocol to achieve the same communication efficiency as the n-server PIR protocol. Given an arbitrary number of files stored on a DSS that uses an $[n, 1, n]$ repetition code, the authors of [63] derived the *replicated-PIR capacity*, which is the largest PIR rate for any n-server PIR protocol, when the nodes do not collude. Additionally, they provided a PIR protocol that achieves this. Extending this work, the authors in [27] derived the MDS-PIR capacity for a DSS where files are stored using an MDS code and where its nodes do not collude. They also presented a PIR protocol that achieved it. Since the MDS-PIR capacity depends on the number of files, in this dissertation we refer to it as the *finite MDS-PIR capacity*. The authors in [64] presented a PIR protocol for a DSS that uses an $[n, k, n - k + 1]$ MDS code to store data. When no nodes collude, they showed that their protocol achieves the *asymptotic MDS-PIR capacity*, i.e., the MDS-PIR capacity for an infinite number of files. The MDS-PIR capacity when the nodes are allowed to collude is still an open problem except in few particular cases [65] and for repetition codes [66]. In [67], the authors presented a PIR protocol for DSSs that stored data using generalized RS codes and ensured privacy even when the nodes collaborated. When the nodes collude, their protocol achieved a better PIR rate than the protocol in [64].

Parallel to the works on traditional PIR protocols, there has been some exciting research on variations of PIR for DSSs. In [68], the authors studied the

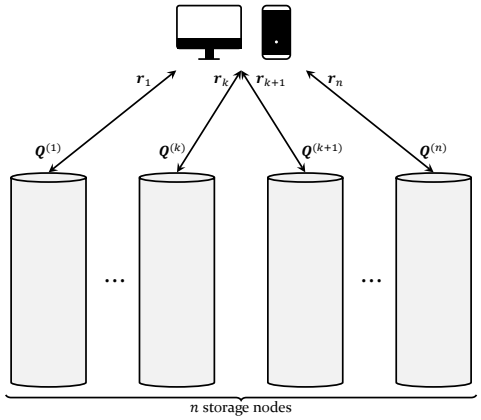

Figure 4.1: Illustration of a DSS containing n nodes that store f files. The user sends the queries $\boldsymbol{Q}^{(j)}$, $j \in \mathbb{N}_n$, to the storage nodes and receives responses $\boldsymbol{r}_j$.

problem of symmetric PIR, and presented the symmetric PIR capacity for linear schemes under the assumption that the DSS uses an MDS code and nodes are allowed to collude. In symmetric PIR, the user should not only retrieve the data privately but also not learn anything about other data that is stored on the DSS. Another interesting variation is the robust PIR introduced in [69], where a subset of nodes fail to respond back to the user, nodes can collude, and the DSS uses a repetition code. The authors of [70] presented a PIR protocol for the single server case, i.e., where all nodes in the DSS collude. In particular, they make use of side information available to the user to obtain an efficient protocol.

4.2 Privacy Model

We consider a DSS where a set of T nodes may act as spies. Further, they might collude and hence they are referred to as colluding nodes. In addition, it is assumed that the remaining nonspy nodes do not collaborate with the spy nodes. The scenario of a single spy node ($T = 1$) in the DSS is analogous to having a system with no colluding nodes. Let $\mathcal{T} \subset \mathbb{N}_n$, $|\mathcal{T}| = T$, denote the set of spy nodes in the DSS. The role of the spy nodes is to determine which file $\boldsymbol{X}^{(m)}$ is accessed by the user. We assume that the user does not know $\mathcal{T}$, since otherwise it can trivially achieve PIR by avoiding contacting the spy nodes. To retrieve file $\boldsymbol{X}^{(m)}$ from the DSS, the user sends a $d \times \beta f$ matrix query $\boldsymbol{Q}^{(l)}$ over $\mathrm{GF}(q) \subseteq \mathrm{GF}(q^\ell)$ to the l-th node for all $l \in \mathbb{N}_n$. The query matrices are represented in the form of d subquery vectors $\boldsymbol{q}_i^{(l)}$ of length βf as

$$\boldsymbol{Q}^{(l)} = \begin{pmatrix} \boldsymbol{q}_1^{(l)} \\ \boldsymbol{q}_2^{(l)} \\ \vdots \\ \boldsymbol{q}_d^{(l)} \end{pmatrix} = \begin{pmatrix} q_{1,1}^{(l)} & q_{1,2}^{(l)} & \cdots & q_{1,\beta f}^{(l)} \\ q_{2,1}^{(l)} & q_{2,2}^{(l)} & \cdots & q_{2,\beta f}^{(l)} \\ \vdots & \vdots & \cdots & \vdots \\ q_{d,1}^{(l)} & q_{d,2}^{(l)} & \cdots & q_{d,\beta f}^{(l)} \end{pmatrix}.$$

The i-th subqueries $\boldsymbol{q}_i^{(l)}$, $l \in \mathbb{N}_n$, of the n queries aim at recovering Γ unique code symbols[1] of the file $\boldsymbol{X}^{(m)}$. In response to the received query matrix, node l sends the column vector

$$\boldsymbol{r}_l = (r_{l,1}, \dots, r_{l,d})^\mathsf{T} = \boldsymbol{Q}^{(l)}\big(c_{1,l}^{(1)}, \dots, c_{\beta,l}^{(1)}, \dots, c_{\beta,l}^{(f)}\big)^\mathsf{T}, \tag{4.1}$$

referred to as the response vector, back to the user as illustrated in Fig. 4.1. We refer to $r_{l,i}$ as the i-th subresponse of the l-th node. Perfect information-theoretic PIR for such a scheme is defined in the following.

Definition 4. *Consider a DSS with n nodes storing f files in which a set of T nodes $\mathcal{T} = \{t_1, \dots, t_T\} \subset \mathbb{N}_n$, $1 \leq |\mathcal{T}| = T \leq n - k$, act as colluding spies. A user who wishes to retrieve the m-th file sends the queries $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_n$, to the storage nodes, which return the responses $\boldsymbol{r}_l$. This scheme achieves perfect information-theoretic PIR if and only if*

$$\textit{Privacy:} \qquad \mathsf{H}\big(m|\boldsymbol{Q}^{(t_1)}, \dots, \boldsymbol{Q}^{(t_T)}\big) = \mathsf{H}(m); \tag{4.2a}$$

$$\textit{Recovery:} \qquad \mathsf{H}\big(\boldsymbol{X}^{(m)}|\boldsymbol{r}_1, \dots, \boldsymbol{r}_n\big) = 0. \tag{4.2b}$$

Queries satisfying (4.2a) ensure that the file requested by the user is independent of the queries. Thus, the colluding nodes in $\mathcal{T}$ do not gain any additional information regarding which file is requested by the user by observing the queries. The recovery constraint in (4.2b) ensures that the user is able to recover the requested file from the responses sent by the DSS.

The efficiency of a PIR protocol, given by its PIR rate, is defined as the amount of retrieved data per unit of total amount of downloaded data, since it is assumed that the content of the retrieved file dominates the total communication cost [61, 64].

Definition 5. *The PIR rate of a PIR protocol, denoted by* R*, is the amount of information retrieved per downloaded symbol, i.e.,*

$$\mathsf{R} \triangleq \frac{\beta k}{nd}.$$

Since the size of each file is βk, the parameters d and Γ should be chosen such that $\beta k = \Gamma d$. For the (file-independent) Protocols 2 and 3 in Paper III to be practical, we may select

$$\beta = \frac{\mathsf{LCM}(k, \Gamma)}{k} \quad \text{and} \quad d = \frac{\mathsf{LCM}(k, \Gamma)}{\Gamma}, \tag{4.3}$$

as it ensures that we have the smallest values of β and d. This is not the case for Protocol 1 in Paper III, where β is exponential in the number of files in order achieve optimal PIR rates. By choosing the values above for β and d, the PIR rate for Protocols 2 and 3 become

$$\mathsf{R} = \frac{\Gamma}{n}.$$

[1] In general, the i-th subqueries recover Γ_i unique code symbols such that among the $\sum_i \Gamma_i$ recovered code symbols there are βk distinct information symbols. However, for the sake of simplicity, we assume $\Gamma_i = \Gamma$ for all i.

We will write $\mathrm{R}(\mathcal{C})$ to highlight that the PIR rate depends on the underlying storage code $\mathcal{C}$. The maximum achievable PIR rate is the PIR capacity. It was shown in [27] that for the noncolluding case and for a given number of files f stored using an $[n, k]$ MDS code, the MDS-PIR capacity, denoted by C_f, is

$$\mathrm{C}_f \triangleq \frac{n-k}{n}\left[1-\left(\frac{k}{n}\right)^f\right]^{-1}. \tag{4.4}$$

Throughout the paper we refer to the capacity in (4.4) as the *finite MDS-PIR capacity* as it depends on the number of files. When the number of files $f \to \infty$, the *asymptotic MDS-PIR capacity* is

$$\mathrm{C}_\infty \triangleq \frac{n-k}{n}. \tag{4.5}$$

It was shown in [61, Th. 3] that the PIR rate for a DSS with noncolluding nodes is upperbounded by C_∞ for a special class of linear retrieval schemes. In the case of colluding nodes, an explicit upper bound is currently unknown, as well as an expression for the MDS-PIR capacity. Some initial work for the case of two colluding nodes has recently been presented in [65], and when the storage codes are repetition codes the MDS-PIR capacity has been derived in [66].

4.3 Achieving PIR

The trivial way to achieve PIR is to download everything from the DSS. However, such a naive scheme is inefficient. In fact, when the DSS consists of just a single node, and when the user does not have any side information then the only way to achieve PIR is the trivial way. A way to achieve a more efficient PIR protocol is to add redundancy in the DSS. The simplest way to realize this is by adding more nodes (containing replicated data) into the system. Again, from a coding perspective, this is a DSS that uses an $[n, 1, n]$ repetition code. In this section, we provide the intuition behind two protocols. The first protocol, which we refer to as the finite PIR protocol, was given in [63] and achieves the MDS-PIR capacity in (4.4), while the second protocol, which we refer to as the asymptotic PIR protocol, was given in [64] and achieves the asymptotic MDS-PIR capacity in (4.5). For the sake of simplicity in the explanations, we consider a DSS (with parameter $\alpha = 1$) storing $f = 2$ files using a $[2, 1, 2]$ repetition code.

4.3.1 Finite PIR Protocol

The protocol is based on two main principles that are applied iteratively.

1. Enforcing symmetry within each node.
2. Exploiting side information in order to recover desired message symbols.

By the first principle, we mean that from each node an equal number of message symbols should be downloaded from each file. Doing this obfuscates the knowledge of the desired file to the nodes, thus ensuring privacy. Doing this means that

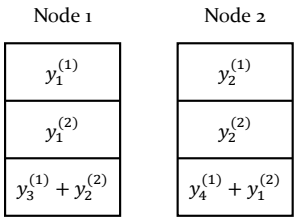

Figure 4.2: Query structure of the finite PIR protocol for the DSS that uses a $[2, 1, 2]$ repetition code with parameters $\alpha = 1$ and $\beta = 4$.

the user downloads undesired symbols from undesired files. The second principle says that these undesired symbols, which can be treated as side information, should be used to recover additional desired message symbols further, thus ensuring that the protocol is as efficient as possible. We explain these ideas using an example.

We consider a DSS that uses a $[2, 1, 2]$ repetition code to encode $f = 2$ files and then stores them across 2 storage nodes. The code is scalar ($\alpha = 1$), and $\beta = 4$. In particular the files $\boldsymbol{X}^{(m)}, m \in \mathbb{N}_2$, are of size 4×1, and as such $\boldsymbol{X}^{(m)} = (x_{1,1}^{(m)}, \cdots, x_{4,1}^{(m)})^\top \in \mathrm{GF}(q)^4$. Suppose that the user requests file $\boldsymbol{X}^{(1)}$. For such a system, the queries to the nodes are summarized in Fig. 4.2, where

$$y_i^{(m)} = x_{\pi(i),1}^{(m)}, \quad i \in \mathbb{N}_4.$$

Function $\pi(\cdot) : \mathbb{N}_4 \to \mathbb{N}_4$ represents a random permutation that is done privately and is known just to the user. Note that the goal of the user is to recover $\{y_1^{(1)}, \ldots, y_4^{(1)}\}$ as this ensures the complete recovery of $\boldsymbol{X}^{(1)}$. The protocol can be broken into three steps. In Step 1, from Nodes 1 and 2 it downloads $y_1^{(1)}$ and $y_2^{(1)}$, respectively. These are the symbols desired by the user. However, to ensure privacy, it needs to download an equal number of symbols from the remaining files (Principle 1). Thus, Step 2 involves downloading $y_1^{(2)}$ and $y_2^{(2)}$ from Node 1 and Node 2, respectively. These are the undesired symbols, which we would like to use in order to retrieve additional desired symbols (Principle 2). Step 3) involves downloading sums $y_3^{(1)} + y_2^{(2)}$ from Node 1 and $y_4^{(1)} + y_1^{(2)}$ from Node 2. Downloading these sums still ensures privacy as one symbol from each file participates in both sums. From previous steps, $y_2^{(2)}$ and $y_1^{(2)}$ are known. As a consequence the user can obtain $y_3^{(1)}$ and $y_4^{(1)}$, respectively. In this way, the user has obtained $y_1^{(1)}, \ldots, y_4^{(1)}$. The PIR rate of the resulting protocol is $\mathrm{R}([2, 1, 2]) = 4/6 = 2/3$. Though the protocol seems simple, it quickly explodes as the number of files increases, and when the DSS uses some nontrivial code.

4.3.2 Asymptotic PIR Protocol

Unlike the previous protocol, the asymptotic PIR protocol is much simpler, in the sense that the protocol is independent of the number of files in the system. Essentially the protocol entails queries that are a sum of a random vector and a

deterministic vector, where each vector in the query plays a crucial role. Adding the random vector to the deterministic vector makes this sum also random, thus ensuring that the queries are statistically independent of the desired file index. As a result, the protocol maintains privacy, whereas the deterministic vector allows for the recovery of the requested file. The recovery occurs in an ensuing way. The response generated by the respective nodes as in (4.1) from the queries above leads to the creation of response symbols that are linear combinations of some *interference symbols*, or linear combinations of such symbols with the desired message symbols. The message symbols (thus, the desired file) are then recovered by *interference cancellation* through solving systems of linear equations. We will explain this in more detail in the following through an example.

As mentioned at the start of the section, we assume that the DSS has two storage nodes and stores two files using a $[2, 1, 2]$ scalar ($\alpha = 1$) repetition code. Unlike the previous protocol, in this protocol we assume that $\beta = 1$ and as such, the files $\boldsymbol{X}^{(m)} = x_{1,1}^{(m)}, m \in \mathbb{N}_2$. Suppose the user wants to retrieve $\boldsymbol{X}^{(1)}$. It generates query matrices as follows,

$$\begin{aligned}\boldsymbol{Q}^{(1)} &= \boldsymbol{q}_1^{(1)} = (u_1, u_2) + (0, 0) = (u_1, u_2),\\ \boldsymbol{Q}^{(2)} &= \boldsymbol{q}_1^{(2)} = (u_1, u_2) + (1, 0) = (u_1 + 1, u_2),\end{aligned}$$

where $u_j \in \mathrm{GF}(q), j \in \mathbb{N}_2$, is chosen uniformly at random from the Galois field. Thereafter, it sends $\boldsymbol{Q}^{(l)}, l \in \mathbb{N}_2$, to the l-th node. To the nodes, their respective queries appear random. Thus, the nodes individually are unable to determine the requested file. Furthermore, in the second query $\boldsymbol{Q}^{(2)}$, a deterministic vector $(1, 0)$ is added to the random vector. The position of the element one in the deterministic vector dictates which file the user requests. In this case, since the goal is to retrieve $\boldsymbol{X}^{(1)}$, the one occurs at the first position while the remaining positions are filled with zeros. The nodes compute responses according to (4.1),

$$\begin{aligned}\boldsymbol{r}_1 = r_{1,1} &= u_1 x_{1,1}^{(1)} + u_2 x_{1,1}^{(2)},\\ \boldsymbol{r}_2 = r_{2,1} &= u_1 x_{1,1}^{(1)} + u_2 x_{1,1}^{(2)} + x_{1,1}^{(1)}.\end{aligned} \tag{4.6}$$

The user then obtains the requested file by performing $\boldsymbol{r}_2 - \boldsymbol{r}_1$. This can be interpreted as cancelling the interference $u_1 x_{1,1}^{(1)} + u_2 x_{1,1}^{(2)}$ to obtain $x_{1,1}^{(1)}$. Such a protocol, though simple, achieves a PIR rate of $\mathrm{R}([2, 1, 2]) = 1/2$, which is lower than the PIR rate of the finite PIR protocol. Although the number of symbols downloaded from the servers is low (it is 2 in this example), Chor *et al.* in [55] stated that this protocol is inefficient as the upload cost is high. However, if we assume that the files $\boldsymbol{X}^{(m)}$ are very big then the download cost dominates the upload cost and therefore the scheme is efficient.

4.4 Towards Arbitrary Linear Codes

All state-of-the-art PIR protocols ([27, 63, 64]) proposed in the literature are constructed using the principles mentioned in the previous section. Furthermore,

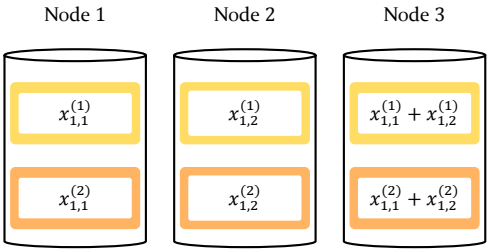

Figure 4.3: A DSS that uses a $[3, 2, 2]$ single parity check code with parameters $\alpha = \beta = 1$, and $f = 2$.

they consider that data is stored using an MDS code. As it was previously mentioned in Chapter 2, MDS codes are inefficient for DSSs. In particular, MDS-coded DSSs have inefficient repair. To counter this problem, codes based on locality have gained prominence in the DSS literature. Some examples of such codes were presented in [11, 12, 37–39]. For simplicity, we refer to such class of codes as LRCs. Another class of repair efficient codes are MBR codes [15, 41]. A common trait across these two classes of codes is that they are not MDS codes. That being the case, in this thesis we present PIR protocols where we relax the MDS property assumption.

To explain the intuition behind the construction of an optimal PIR protocol for a DSS that uses an arbitrary linear code, we make use of an example. More specifically, we consider the DSS in Fig. 4.3 that uses a single parity check (SPC) code. Although SPCs are MDS codes, we use it in the example to provide an alternative view of the construction that is more related to the structure of the code used in the DSS. This in turn provides an intuition for constructing PIR protocols for an arbitrary linear code. Assume that the user wants to retrieve $\boldsymbol{X}^{(1)}$ from the DSS in Fig. 4.3 and it uses the asymptotic PIR protocol. The user would generate query matrices $\boldsymbol{Q}^{(l)}, l \in \mathbb{N}_3$, of size 2×2, i.e., she would send $d = 2$ subqueries to each node. For explanation purposes, we focus on the first subqueries sent to the nodes,

$$\begin{aligned}\boldsymbol{q}_1^{(1)} &= (u_1, u_2) + (1, 0) = (u_1 + 1, u_2),\\ \boldsymbol{q}_1^{(2)} &= \boldsymbol{q}_1^{(3)} = (u_1, u_2).\end{aligned}$$

The user sends these subqueries to the nodes and then gets back the correspond-

ing subresponses as follows,

$$r_{1,1} = \boldsymbol{q}_1^{(1)} \begin{pmatrix} x_{1,1}^{(1)} \\ x_{1,1}^{(2)} \end{pmatrix} = u_1 x_{1,1}^{(1)} + u_2 x_{1,1}^{(2)} + x_{1,1}^{(1)} \triangleq I_1 + x_{1,1}^{(1)},$$
$$r_{2,1} = \boldsymbol{q}_1^{(2)} \begin{pmatrix} x_{1,2}^{(1)} \\ x_{1,2}^{(2)} \end{pmatrix} = u_1 x_{1,2}^{(1)} + u_2 x_{1,2}^{(2)} \triangleq I_2,$$
$$r_{3,1} = \boldsymbol{q}_1^{(3)} \begin{pmatrix} x_{1,1}^{(1)} + x_{1,2}^{(1)} \\ x_{1,1}^{(2)} + x_{1,2}^{(2)} \end{pmatrix} = u_1 x_{1,1}^{(1)} + u_2 x_{1,1}^{(2)} + u_1 x_{1,2}^{(1)} + u_2 x_{1,2}^{(2)} \triangleq I_1 + I_2,$$

where I_1 and I_2 are the interference symbols. To recover $x_{1,1}^{(1)}$, one needs to know what I_1 is. Of course one can get it through interference cancellation. However, we look at it from a different perspective. We consider (I_1, I_2) to be a random message vector that is encoded using a $[3, 2, 2]$ SPC to give a codeword $(I_1, I_2, I_1 + I_2)$. In the above system of equations, I_1 is unknown; thus we consider it as an erasure. In other words, the user has the erasure-corrupted codeword

$$(?, I_2, I_1 + I_2).$$

Solving for I_1 corresponds to a decoding problem of the $[3, 2, 2]$ SPC code $\mathcal{C}$. We know for a fact that this SPC code can correct a single erasure. As the codeword has a single erasure, the user can obtain I_1, and therefore obtain $x_{1,1}^{(1)}$. By swapping the queries to Nodes 1 and 2, one can recover $x_{1,2}^{(1)}$ in a similar way, and thus recover the whole file $\boldsymbol{X}^{(1)}$.

What we see in the above example is that in the i-th subquery, $i \in \mathbb{N}_d$, if $\boldsymbol{q}_i^{(j)}, j \in \mathbb{N}_n$, is a sum of a random and a deterministic vector then this corresponds to having the j-th coordinate in some imaginary codeword of $\mathcal{C}$ erased. Moreover, for complete recovery, we would like to obtain all code symbols of this codeword. In other words, we should be able to perform successful decoding. To improve the efficiency of the protocol, one should have a large number (say Γ) of i-th subqueries to the n nodes to be a sum of a random and a deterministic vector. The question is: how large can Γ be before breaking the protocol? A trivial answer is that we can set $\Gamma = d_{\min}^{\mathcal{C}} - 1$ as this is the number of erasures a code $\mathcal{C}$ can correct irrespective of the positions in which they occur.

Yet, nontrivially one can have Γ in the range $d_{\min}^{\mathcal{C}} - 1 < \Gamma \leq n - k$ because it is possible to correct $\Gamma > d_{\min}^{\mathcal{C}} - 1$ erasures in a codeword, if the erasures occur at specific positions. This depends on the structure of the code $\mathcal{C}$. In Papers III and IV, we use this fact to construct optimal PIR protocols for a DSS that uses a linear code and show that it is possible to achieve the MDS-PIR capacity even when the DSS uses non-MDS codes. Furthermore, we give partial results on when this is possible, which is related to the generalized Hamming weights and code automorphisms of the underlying code $\mathcal{C}$ of the DSS.

4.5 PIR in Distributed Caching

In the distributed caching scenario, we assume that some of the SBSs are spy nodes that (potentially) collaborate with each other. On the other hand, we assume that the MBS can be trusted. The users wish to retrieve files from the wireless network, but do not want the spy nodes to learn any information about which file the user requests. The goal is to retrieve data from the network privately, while minimizing the use of the backhaul link, i.e., while minimizing $\mathtt{R}_{\mathsf{bh}}$. Thus, the goal is to optimize the content placement $\boldsymbol{\mu}$ to minimize $\mathtt{R}_{\mathsf{bh}}$. Furthermore, the F files in the cellular network are encoded using F codes $\mathcal{C}_i, i \in \mathbb{N}_F$, with a code rate $R^{\mathcal{C}_i}$ in accordance to their popularity distribution $\boldsymbol{p}$. As a consequence, the PIR problem in distributed caching can somewhat be seen as a generalization of the PIR problem in DSSs.

4.5.1 Content Placement for the PIR Scenario

For the PIR scenario, the content placement in the SBSs is slightly different from the content placement (see Section 2.5.1) when the PIR property is not the objective. We explain the difference in the following. The system model for distributed caching described in Section 2.5 says that the i-th file $\boldsymbol{X}^{(i)}, i \in \mathbb{N}_F$, is partitioned into βk_i packets of size L/k_i bits that are then encoded using an $[N_{\mathsf{SBS}}, k_i]$ code to obtain βN_{SBS} coded packets. These are then stored on N_{SBS} SBSs. As mentioned in Section 4.3, to achieve PIR, it is necessary to have redundancy in the system. Therefore, unlike in Section 2.5, for the PIR scenario $k_i < N_{\mathsf{SBS}}$. In particular, the content placement $\boldsymbol{\mu}$ in the SBSs must satisfy

$$\mu_i \in \mathcal{M} \triangleq \{0, 1/(N_{\mathsf{SBS}} - 1), \ldots, 1/2, 1\}.$$

Note that $\mu_i = 1/N_{\mathsf{SBS}}$ is not allowed. This is in contrast to the case of no PIR, where $k_i = N_{\mathsf{SBS}}$ (and hence $\mu_i = 1/N_{\mathsf{SBS}}$) is possible (see (2.2)).

Chapter 5

Conclusions and Future Work

In this chapter, we summarize the contributions from the attached papers and provide potentially interesting ideas to extend the work presented in this thesis. This thesis covers three main aspects of distributed storage, which are individually covered in Papers I-IV. They are efficient storage in DSSs, security in DSSs, and privacy in DSSs. In this regard, Paper V is a bit different as it concerns privacy in wireless networks.

Repair Efficient Codes for Distributed Storage (Paper I)

In Paper I, we propose a new coding scheme for DSSs that performs efficient repair of a systematic node failure with low repair complexity. These codes are constructed using two smaller codes. The first code, named as the Class A code, is constructed using an MDS code, and then modified using a piggybacking scheme. The aim of the Class A code is to provide the fault tolerance as well as, through the piggybacking scheme, reduce the read cost for the repair of a number of symbols in the failed node. The second code, referred to as Class B code, is constructed in such a way that its parity symbols are a sum of certain systematic symbols, such that it can repair the remaining symbols in the failed node at low read cost and with low repair complexity. This ensures that the repair of the failed node occurs with low repair bandwidth and complexity.

In the paper, we provide two constructions of the Class B codes. The first is based on a very simple mathematical construct, and the second is based on a more complicated heuristic algorithm that further reduces the repair bandwidth compared to the first construction. Subsequently, for our proposed codes we characterize the fault tolerance, repair bandwidth, repair complexity, and encoding complexity. Finally, we numerically compare the performance of the code with various state-of-the-art codes. Our codes have a better repair complexity compared to Zigzag codes [14], MDS codes, Piggyback codes [42], generalized Piggyback codes [43], exact-repairable MDS codes [44], binary addition and shift implementable cyclic convolutional product matrix MBR codes [71], and in some cases better than Pyramid codes [11]. They also achieve a better repair bandwidth compared to MDS codes, Piggyback codes, generalized Piggyback codes, exact-repairable MDS codes, and are in some cases better than LRCs [12] and Pyramid codes. Interestingly, our proposed codes have a subpacketization that grows

linearly with the code dimension. Thus, they are good for memory constrained DSSs.

Security in Distributed Storage (Paper II)

In Paper II, we provide a coding scheme that ensures information-theoretic security against an (ℓ_1, ℓ_2) eavesdropper. The scheme involves serially concatenating a Gabidulin code with an RFC. The Gabidulin code ensures that the concatenated code provides security, while the RFC provides efficient repair of failed nodes and allows multiple parallel reads of data symbols stored on the DSS. To prove the information-theoretic security, we introduce a necessary and a sufficient condition for no information leakage to the eavesdropper. In fact, this condition is a further generalization of the sufficient condition in [21, 22]. The scheme of using a Gabidulin code as an outer code with an inner code to provide security, as well as, efficient repair of failed nodes was first proposed by [21]. In particular, their construction had either an MSR code or an LRC as the inner code. Unlike LRCs, RFCs have overlapping local groups. Thus, we generalize the proofs in [21].

Privacy in Distributed Systems (Papers III, IV, and V)

Papers III and IV deal with the PIR problem in DSSs. Paper III considers a DSS where data is stored using an arbitrary linear code and gives PIR protocols for such systems. The fundamental work in Paper III shows that it is possible to achieve the MDS-PIR capacity with a class of linear codes, and gives the structural properties of this class of codes. Paper IV shows that this rate is indeed the fundamental limit on the maximum PIR rate for any PIR protocol given the same underlying class of storage codes. For DSSs that do not use codes in this class, Paper IV also presents PIR protocols with improved PIR rates.

In particular, in Paper III we provide three PIR protocols, referred to as Protocols 1, 2, and 3. Protocols 1 and 2 achieve the PIR property under the assumption that the nodes in the DSS do not collude, whereas Protocol 3 achieves the PIR property even when the above condition is relaxed, i.e., the nodes are allowed to collude. Protocol 1 achieves a better PIR rate (when the DSS has a finite number of files) than Protocol 2, it requires the files stored to be exponentially large in the number of files, and is more complex than Protocol 2. Highlights of the two protocols are that Protocol 1 achieves the finite MDS-PIR capacity and Protocol 2 achieves the asymptotic MDS-PIR capacity even when the underlying storage code is non-MDS. Surprisingly enough, the two protocols achieve their respective MDS-PIR capacity for the same class of codes, which we refer to as the MDS-PIR capacity-achieving codes. Thus, we show that the MDS property required to achieve finite/asymptotic MDS-PIR capacity [27, 63, 64] is strictly unnecessary and overly restrictive. We then delve into the fundamentals of such codes and go on to provide a necessary condition based on generalized Hamming weights and a sufficient condition based on the automorphisms of the underlying storage codes. Subsequently, we prove that cyclic codes, RM codes, and distance-optimal information locality codes are MDS-PIR capacity-achieving codes. In the case when the DSS stores data using other codes, we provide an optimization algorithm that

maximizes the PIR rate of these protocols. After that, we consider the scenario where the nodes are allowed to collude. For such a scenario, we provide Protocol 3 that is a generalization of the protocol in [67] and improves its PIR rate. The protocol is based on three codes: the storage code, the query code, and the retrieval code. The query code characterizes the query of the protocol while the retrieval code determines the retrieval process. Unlike the protocol in [67], Protocol 3 achieves a PIR rate that is not limited to the minimum distance of the retrieval code for an arbitrary underlying storage code. Besides generalized RS codes [67] and RM codes [72], we show that codes based on the $(\mathcal{U}|\mathcal{U}+\mathcal{V})$ construction, where $\mathcal{U}$ can be an arbitrary binary linear code and $\mathcal{V}$ is a repetition code, can be used with this protocol. Furthermore, we prove that RM codes achieve the maximum PIR rate achievable by this protocol. As in the noncolluding case, we provide a necessary and a sufficient condition to achieve the maximum PIR rate for Protocol 3.

In Paper IV, we consider the noncolluding case. It starts from where Paper III ends by proving that PIR rates achieved by Protocol 1, for a given number of files in the system, are indeed equal to the maximum PIR capacity achieved by any protocol when the DSS uses MDS-PIR capacity-achieving codes. Furthermore, this implies that the rates achieved by Protocol 2, when the DSS uses the codes mentioned above, are equal to the asymptotic PIR capacity achieved by any protocol. When the DSS uses codes that are not in the class of MDS-PIR capacity-achieving codes, we present protocols that improve upon the PIR rate achieved by Protocols 1 and 2. In particular, we present a file-dependent and a file-independent PIR protocol that improve on the PIR rates of Protocols 1 and 2, respectively. Such protocols are based on the subcode property of the underlying storage code in the DSS.

In Paper V, we shift our focus from DSSs to DC where we look at privacy in wireless networks that cache content on several SBSs in order to reduce the backhaul usage. In the paper, we add the PIR problem to the traditional caching problem and ask the question what is the best content placement, such that the PIR property is achieved. In particular, we consider that multiple SBSs can collude. To this extent we present a PIR protocol that is an extension of Protocol 3 to the case where data is stored using codes with different code rates, depending upon the popularity of the data. We go on to derive the backhaul rate of the system and formulate the optimal content placement optimization. Interestingly, contrary to the case of the traditional caching problem, we show that uniform content allocation is optimal [30]. In other words, all content should be encoded with the same code rate (uniform rate) and then stored on SBSs. Additionally, we see numerically that for specific scenarios, the best coding scheme is a repetition code. In other words, popular content caching is optimal for specific scenarios.

5.1 Future Work

This thesis is concerned with wide-ranging topics of efficient storage, security, and privacy in DSSs and as such has much potential when it comes to future work. Some of the important extensions of the work done in the appended papers are

listed below.

1. One important extension would be to present new code constructions that have low repair bandwidth and low repair complexity for the repair of arbitrary failed nodes. Additionally, it would be interesting if this can be further extended to the scenario of simultaneous node repair.

2. It would be interesting to design coding schemes that achieve information-theoretic security in conjunction with PIR protocols. Trivially, one can use a secure coding scheme with the PIR protocols suggested in this thesis but is this the best way? Can the common randomness in the coding scheme and queries be leveraged?

3. Because Protocol 3 is quite restrictive, it cannot be used with any storage code and any given number of nodes that collude. Alternative constructions to Protocol 3 is one of the obvious extensions.

4. Another interesting future work is to determine the PIR capacity (for any storage code) when nodes are not allowed to collude, and when they are allowed to collude.

5. In Paper V, the system model assumed is simple as we assume that the MBS is not an adversary. It would be quite interesting to look at PIR protocols that achieve privacy not just from SBSs but also from the MBS. Subsequently, the question to ask is what is the best content placement in this scenario.

Bibliography

[1] R. Kluver, "Globalization, informatization, and intercultural communication," *American Commun. Journal*, vol. 3, no. 3, May 2000.

[2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute, Tech. Rep., May 2011.

[3] J. Gantz and D. Reinsel, "THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," EMC Corporation, Tech. Rep., 2012.

[4] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," in *Proc. 39th Very Large Data Bases Endowment (VLDB)*, Trento, Italy, Aug. 2013.

[5] S. Abiteboul, I. Manolescu, P. Rigaux, M. Rousset, and P. Senellart, *Web Data Managment*. Cambridge University Press, 2011.

[6] S. Sankar, M. Shaw, K. Vaid, and S. Gurumurthi, "Datacenter scale evaluation of the impact of temperature on hard disk drive failures," *ACM Trans. Storage*, vol. 9, no. 2, pp. 6:1–6:24, Jul. 2013.

[7] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Oct. 2003.

[8] J. J. Hanson, "An Introduction to the Hadoop Distributed File System," IBM, Tech. Rep., Feb. 2011.

[9] A. Datta and F. Oggier, "An overview of codes tailor-made for better repairability in networked distributed storage systems," *ACM Special Interest Group on Algo. and Computation Theory*, vol. 44, no. 1, pp. 89–105, Mar. 2013.

[10] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, "Scale-RS: An efficient scaling scheme for RS-coded storage clusters," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1704–1717, Jun. 2015.

[11] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. IEEE Int. Symp. Network Comput. and Appl. (NCA)*, Cambridge, MA, Jul. 2007.

[12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in Windows Azure storage," in *Proc. USENIX Annual Technical Conf.*, Boston, MA, Jun. 2012.

[13] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[14] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[15] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[16] V. Guruswami and M. Wootters, "Repairing Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5684–5698, Sep. 2017.

[17] I. Tamo, M. Ye, and A. Barg, "Optimal repair of Reed-Solomon codes: Achieving the cut-set bound," May 2017, arXiv:cs/1706.00112v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1706.00112

[18] F. Knop, S. R. Patil, and L. Coyne, "IBM spectrum scale security," IBM, Tech. Rep., Jan. 2017.

[19] J. Garay, R. Gennaro, C. Jutla, and T. Rabin, "Secure distributed storage and retrieval," in *Proc. Int. Workshop Dist. Algo. (WDAG)*, Saarbrüken, Germany, Sep. 1997.

[20] R. Pletka and C. Cachin, "Cryptographic security for a high-performance distributed file system," in *Proc. IEEE Conf. on Mass Storage Sys. and Tech. (MSST)*, San Diego, CA, Sep. 2007.

[21] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 60, no. 1, pp. 212–236, Jan. 2014.

[22] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "Information-theoretically secure regenerating codes for distributed storage," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Houston, TX, Dec. 2011.

[23] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6734–6753, Oct. 2011.

[24] M. Asteris and A. G. Dimakis, "Repairable fountain codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 1037–1047, May 2014.

[25] "Informational privacy in the digital age," American Civil Liberties Union (ACLU), Tech. Rep., Feb. 2015.

[26] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. Annual IEEE Symp. Foundations Comp. Sci. (FOCS)*, Milwaukee, WI, Oct. 1995, pp. 41–50.

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

[28] "Visual netorking index," Cisco, Tech. Rep., 2014.

[29] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[30] V. Bioglio, F. Gabry, and I. Land, "Optimizing MDS codes for caching at the edge," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, Dec. 2015, pp. 1–6.

[31] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[32] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Sov. Phys. Doklady*, vol. 7, no. 7, pp. 595–596, Jan. 1963.

[33] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba algorithm for efficient implementations," Jul. 2006. [Online]. Available: https://eprint.iacr.org/2006/224.pdf

[34] J. M. Pollard, "The fast Fourier transform in a finite field," *Math. Comput.*, vol. 25, no. 114, pp. 365–374, 1971.

[35] R. E. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*. Springer, 2001.

[36] S. Gao and T. Mateer, "Additive fast Fourier transforms over finite fields," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6265–6272, Dec. 2010.

[37] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[38] D. Papailiopoulos and A. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[39] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, Aug. 2014.

[40] Y. Wang, X. Yin, and X. Wang, "MDR codes: A new class of RAID-6 codes with optimal rebuilding and encoding," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 1008–1018, May 2014.

[41] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annual Allerton Conf. Commun., Control, and Comput.*, Monticello, IL, Sep./Oct. 2010.

[42] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5802–5820, Sep. 2017.

[43] S. Yuan and Q. Huang, "Generalized piggybacking codes for distributed storage systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, Dec. 2016.

[44] A. S. Rawat, I. Tamo, V. Guruswami, and K. Efremenko, "MDS code constructions with small sub-packetization and near-optimal repair bandwidth," *IEEE Trans. Inf. Theory*, to appear.

[45] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[46] A. Piemontese and A. Graell i Amat, "MDS-coded distributed storage for low delay wireless content delivery," in *Proc. IEEE Int. Symp. on Turbo Codes and Itr. Inf. Processing (ISTC)*, Brest, France, Sep. 2016.

[47] J. Pedersen, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Distributed storage in mobile wireless networks with device-to-device communication," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4862–4878, Nov. 2016.

[48] ——, "Optimizing MDS coded caching in wireless networks with device-to-device communication," 2018, arXiv:1701.06289v2 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1701.06289

[49] U. Niesen, D. Shah, and G. W. Wornell, "Caching in wireless networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6524–6540, Oct. 2012.

[50] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[51] O. Goldreich, *Foundations of Cryptography: Volume II, Basic Applications*. Cambridge, U. K.: Cambridge University Press, 2004.

[52] H. Delfs and H. Knebl, *Introduction to Cryptography: Principles and Applications*, 2nd ed. New York, NY, USA: Springer-Verlag, 2007.

[53] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[54] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems Inf. Transmiss.*, vol. 21, pp. 1–12, Jul. 1985.

[55] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, Nov. 1998.

[56] A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond, "Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval," in *Proc. Annual IEEE Symp. Foundations Comp. Sci. (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 261–270.

[57] S. Yekhanin, "Towards 3-query locally decodable codes of subexponential length," *Journal of the ACM*, vol. 55, no. 1, pp. 1–16, Feb. 2008.

[58] K. Efremenko, "3-query locally decodable codes of subexponential length," in *Proc. 41th Annual ACM Symp. Theory Comput. (STOC)*, Bethesda, MD, Jun. 2009, pp. 39–44.

[59] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 36th Annual ACM Symp. Theory Comput. (STOC)*, Chicago, IL, Jun. 2004, pp. 262–271.

[60] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, Jun./Jul. 2014, pp. 856–860.

[61] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, China, Jun. 2015, pp. 2842–2846.

[62] A. Fazeli, A. Vardy, and E. Yaakobi, "Codes for distributed PIR with low storage overhead," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, China, Jun. 2015, pp. 2852–2856.

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

[64] R. Tajeddine, O. W. Gnilke, and S. El Rouayheb, "Private information retrieval from MDS coded data in distributed storage systems," *to app. in IEEE Trans. Inf. Theory*, 2018.

[65] 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 Trans. Inf. Theory*, vol. 64, no. 2, pp. 1000–1022, Feb. 2017.

[66] ——, "The capacity of robust private information retrieval with colluding databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2361–2370, Apr. 2018.

[67] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, and D. A. Karpuk, "Private information retrieval from coded databases with colluding servers," *SIAM J. Appl. Algebra Geom.*, vol. 1, no. 1, pp. 647–664, Nov. 2017.

[68] Q. Wang and M. Skoglund, "Linear symmetric private information retrieval for MDS coded distributed storage with colluding servers," Aug. 2017, arXiv:1708.05673v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1708.05673

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

[70] S. Kadhe, B. Garcia, A. Heidarzadeh, S. El Rouayheb, and A. Sprintson, "Private information retrieval with side information: The single server case," in *Proc. 55th Annual Allerton Conf. Commun, Control, and Comput.*, Monticello, IL, Oct. 2017.

[71] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC codes: Low-complexity regenerating codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3053–3069, Jun. 2016.

[72] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, A.-L. Horlemann-Trautmann, D. Karpuk, and I. Kubjas, "Reed-Muller codes for private information reterival," in *Proc. 10th Int. Workshop Coding Cryptography (WCC)*, Saint-Petersburg, Russia, Sep. 2017.

Part B

Papers

Paper I

Code Constructions for Distributed Storage With Low Repair Bandwidth and Low Repair Complexity

Siddhartha Kumar, Alexandre Graell i Amat, Iryna Andriyanova, Fredrik Brännström, and Eirik Rosnes

IEEE Transactions on Communications, to appear, 2018.

Parts of this paper were presented at the IEEE Global Communications Conference (GLOBECOM), San Diego, CA, December 2015.

The layout has been revised.

Abstract

We present the construction of a family of erasure correcting codes for distributed storage that achieve low repair bandwidth and complexity at the expense of a lower fault tolerance. The construction is based on two classes of codes, where the primary goal of the first class of codes is to provide fault tolerance, while the second class aims at reducing the repair bandwidth and repair complexity. The repair procedure is a two-step procedure where parts of the failed node are repaired in the first step using the first code. The downloaded symbols during the first step are cached in the memory and used to repair the remaining erased data symbols at minimal additional read cost during the second step. The first class of codes is based on MDS codes modified using piggybacks, while the second class is designed to reduce the number of additional symbols that need to be downloaded to repair the remaining erased symbols. We numerically show that the proposed codes achieve better repair bandwidth compared to MDS codes, codes constructed using piggybacks, and local reconstruction/Pyramid codes, while a better repair complexity is achieved when compared to MDS, Zigzag, Pyramid codes, and codes constructed using piggybacks.

1 Introduction

In recent years, there has been a widespread adoption of distributed storage systems (DSSs) as a viable storage technology for Big Data. Distributed storage provides an inexpensive storage solution for storing large amounts of data. Formally, a DSS is a network of numerous inexpensive disks (or nodes) where data is stored in a distributed fashion. Storage nodes are prone to failures, and thus to losing the stored data. Reliability against node failures (commonly referred to as fault tolerance) is achieved by means of erasure correcting codes (ECCs). ECCs are a way of introducing structured redundancy, and for a DSS, it means addition of redundant nodes. In case of a node failure, these redundant nodes allow complete recovery of the data stored. Since ECCs have a limited fault tolerance, to maintain the initial state of reliability, when a node fails a new node needs to be added to the DSS network and populated with data. The problem of repairing a failed node is known as the repair problem.

Current DSSs like Google File System II and Quick File System use a family of Reed-Solomon (RS) ECCs [1]. Such codes come under a broader family of maximum distance separable (MDS) codes. MDS codes are optimal in terms of the fault tolerance/storage overhead tradeoff. However, the repair of a failed node requires the retrieval of large amounts of data from a large subset of nodes. Therefore, in the recent years, the design of ECCs that reduce the cost of repair has attracted significant attention. Pyramid codes [2] were one of the first code con-

structions that addressed this problem. In particular, Pyramid codes are a class of non-MDS codes that aim at reducing the number of nodes that need to be contacted to repair a single failed node, known as the repair locality. Other non-MDS codes that reduce the repair locality are local reconstruction codes (LRCs) [3] and locally repairable codes [4, 5]. Such codes achieve a low repair locality by ensuring that the parity symbols are a function of a small number of data symbols, which also entails a low repair complexity, defined as the number of elementary additions required to repair a failed node. Furthermore, for a fixed locality LRCs and Pyramid codes achieve the optimal fault tolerance.

Another important parameter related to the repair is the repair bandwidth, defined as the number of symbols downloaded to repair a single failed node. Dimakis *et al.* [6] derived an optimal repair bandwidth-storage per node tradeoff curve and defined two new classes of codes for DSSs known as minimum storage regenerating (MSR) codes and minimum bandwidth regenerating (MBR) codes that are at the two extremal points of the tradeoff curve. MSR codes are MDS codes with the best storage efficiency, i.e., they require a minimum storage of data per node (referred to as the sub-packetization level). On the other hand, MBR codes achieve the minimum repair bandwidth. Product-Matrix MBR (PM-MBR) codes and Fractional Repetition (FR) codes in [7] and [8], respectively, are examples of MBR codes. In particular, FR codes achieve low repair complexity at the cost of high storage overheads. Codes such as minimum disk input/output repairable (MDR) codes [9] and Zigzag codes [10] strictly fall under the class of MSR codes. These codes have a high sub-packetization level. Alternatively, the MSR codes presented in [11–18] achieve the minimum possible sub-packetization level.

Piggyback codes presented in [19] are another class of codes that achieve a sub-optimal reduction in repair bandwidth with a much lower sub-packetization level in comparison to MSR codes, using the concept of *piggybacking*. Piggybacking consists of adding carefully chosen linear combinations of data symbols (called piggybacks) to the parity symbols of a given ECC. This results in a lower repair bandwidth at the expense of a higher complexity in encoding and repair operations. More recently, the authors in [20] presented a family of codes that reduce the encoding and repair complexity of PM-MBR codes while maintaining the same level of fault tolerance and repair bandwidth. However, this comes at the cost of large alphabet size. In [21], binary MDS array codes that achieve optimal repair bandwidth and low repair complexity were introduced, with the caveat that the file size is asymptotic and that the fault tolerance is limited to 3.

In this paper, we propose a family of non-MDS ECCs that achieve low repair bandwidth and low repair complexity while keeping the field size relatively small and having variable fault tolerance. In particular, we propose a systematic code construction based on two classes of parity symbols. Correspondingly, there are two classes of parity nodes. The first class of parity nodes, whose primary goal is to provide erasure correcting capability, is constructed using an MDS code modified by applying specially designed piggybacks to some of its code symbols. As a secondary goal, the first class of parity nodes enable to repair a number of data symbols at a low repair cost by downloading piggybacked symbols. The second class of parity nodes is constructed using a block code whose parity symbols are

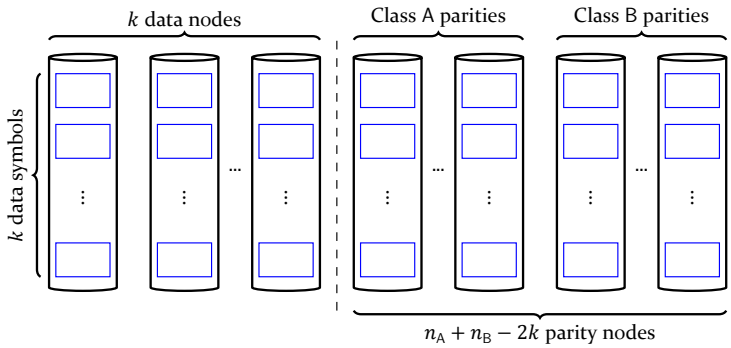

Figure I.1: System model of the DSS.

obtained through simple additions. The purpose of this class of parity nodes is not to enhance the erasure correcting capability, but rather to facilitate node repair at low repair bandwidth and low repair complexity by repairing the remaining failed symbols in the node. Compared to [22], we provide two constructions for the second class of parity nodes. The first one is given by a simple equation that represents the algorithmic construction in [22]. The second one is a heuristic construction that is more involved, but further reduces the repair bandwidth in some cases. Furthermore, we provide explicit formulas for the fault tolerance, repair bandwidth, and repair complexity of the proposed codes and numerically compare with other codes in the literature. The proposed codes achieve better repair bandwidth compared to MDS codes, Piggyback codes, generalized Piggyback codes [23], and exact-repairable MDS codes [24]. For certain code parameters, we also see that the proposed codes have better repair bandwidth compared to LRCs and Pyramid codes. Furthermore, they achieve better repair complexity than Zigzag codes, MDS codes, Piggyback codes, generalized Piggyback codes, exact-repairable MDS codes, and binary addition and shift implementable cyclic-convolutional (BASIC) PM-MBR codes [20]. Also, for certain code parameters, the codes have better repair complexity than Pyramid codes. The improvements over MDS codes, MSR codes, and the classes of Piggyback codes come at the expense of a lower fault tolerance in general.

2 System Model and Code Construction

We consider the DSS depicted in Fig. I.1, consisting of storage nodes, of which k are data nodes and $n - k$ are parity nodes. Consider a file that needs to be stored on the DSS. We represent a file as a $k \times k$ matrix $\boldsymbol{D} = [d_{i,j}]$, called the data array, over GF(q), where GF(q) denotes the Galois field of size q, with q being a prime number or a power of a prime number. In order to achieve reliability against node failures, the matrix $\boldsymbol{D}$ is encoded using an (n, k) vector code [25] to obtain a code matrix $\boldsymbol{C} = [c_{i,j}]$, referred to as the code array, of size $k \times n$, $c_{i,j} \in$ GF(q). The symbol $c_{i,j}$ in $\boldsymbol{C}$ is then stored at the i-th row of the j-th node in the DSS. Thus, each node stores k symbols. Each row in $\boldsymbol{C}$ is referred to as

a stripe so that each file in the DSS is stored over k stripes in n storage nodes. We consider the (n, k) code to be systematic, which means that $c_{i,j} = d_{i,j}$ for $i, j = 0, \ldots, k-1$. Correspondingly, we refer to the k nodes storing systematic symbols as data nodes and the remaining $n-k$ nodes containing parity symbols only as parity nodes. The efficiency of the code is determined by the code rate, given by $R = k^2/kn = k/n$. Alternatively, the inverse of the code rate is referred to as the storage overhead.

For later use, we denote the set of message symbols in the k data nodes as $\mathcal{D} = \{d_{i,j}\}$ and by $\mathcal{P}_t$, $t = k, \ldots, n-1$, the set of parity symbols in the t-th node. Subsequently, we define the set $\mathcal{D}_\mathcal{I} \subseteq \mathcal{D}$ as

$$\mathcal{D}_\mathcal{I} = \{d_{i,j} \in \mathcal{D} \mid (i,j) \in \mathcal{I}\},$$

where $\mathcal{I}$ is an arbitrary index set. We also define the operator $(a+b)_k \triangleq (a+b) \bmod k$ for integers a and b.

Our main goal is to construct codes that yield low repair bandwidth and low repair complexity of a single failed data node. We focus on the repair of data nodes since the raw data is stored on these nodes and the users can readily access the data through these nodes. Thus, their survival is a crucial aspect of a DSS. To this purpose, we construct a family of systematic (n, k) codes consisting of two different classes of parity symbols. Correspondingly, there are two classes of parity nodes, referred to as Class A and Class B parity nodes, as shown in Fig. I.1. Class A and Class B parity nodes are built using an (n_A, k) code and an (n_B, k) code, respectively, such that $n = n_\mathsf{A} + n_\mathsf{B} - k$. In other words, the parity nodes from the (n, k) code correspond to the parity nodes of Class A and Class B codes. The primary goal of Class A parity nodes is to achieve a good erasure correcting capability, while the purpose of Class B nodes is to yield low repair bandwidth and low repair complexity. In particular, we focus on the repair of data nodes. The repair bandwidth (in bits) per node, denoted by γ, is proportional to the average number of symbols (data and parity) that need to be downloaded to repair a data symbol, denoted by λ. More precisely, let β be the sub-packetization level of the DSS, which is the number of symbols per node.[1] Then,

$$\lambda = \frac{\gamma}{\nu\beta}, \tag{I.1}$$

where $\nu = m\lceil \log_2 p \rceil$ is the size (in bits) of a symbol in GF(q), where $q = p^m$ for some prime number p and positive integer m. λ can be interpreted as the repair bandwidth normalized by the size (in bits) of a node, and will be referred to as the *normalized* repair bandwidth.

The main principle behind our code construction is the following. The repair is performed one symbol at a time. After the repair of a data symbol is accomplished, the symbols read to repair that symbol are cached in the memory. Therefore, they can be used to repair the remaining data symbols at no additional read cost. The proposed codes are constructed in such a way that the repair of a new data symbol requires a low additional read cost (defined as the number of additional symbols that need to be read to repair the data symbol), so that λ (and hence γ) is kept low.

[1] For our code construction, $\beta = k$, but this is not the case in general.

Definition 6. *The* read cost *of a symbol is the number of symbols that need to be read to repair the symbol. For a symbol that is repaired after some others, the* additional read cost *is defined as the number of additional symbols that need to be read to repair the symbol. (Note that symbols previously read to repair other data symbols are already cached in the memory and to repair a new symbol only some extra symbols may need to be read.)*

3 Class A Parity Nodes

Class A parity nodes are constructed using a modified (n_{A}, k) MDS code, with $k+2 \leq n_{\mathsf{A}} < 2k$, over GF$(q)$. In particular, we start from an (n_{A}, k) MDS code and apply piggybacks [19] to some of the parity symbols. The construction of Class A parity nodes is performed in two steps as follows.

1) Encode each row of the data array using an (n_{A}, k) MDS code (same code for each row). The parity symbol $p_{i,j}^{\mathsf{A}}$ is obtained as[2]

$$p_{i,j}^{\mathsf{A}} = \sum_{l=0}^{k-1} \alpha_{l,j} d_{i,l},\ j = k, \ldots, n_{\mathsf{A}} - 1, \tag{I.2}$$

where $\alpha_{l,j}$ denotes a coefficient in GF(q) and $i = 0, \ldots, k-1$. Store the parity symbol in the corresponding row of the code array. Overall, $k(n_{\mathsf{A}} - k)$ parity symbols are generated.

2) Modify some of the parity symbols by adding piggybacks. Let τ, $1 \leq \tau \leq n_{\mathsf{A}} - k - 1$, be the number of piggybacks introduced per row. The parity symbol $p_{i,u}^{\mathsf{A}}$ is updated as

$$p_{i,u}^{\mathsf{A,p}} = p_{i,u}^{\mathsf{A}} + d_{(i+u-n_{\mathsf{A}}+\tau+1)_k, i}, \tag{I.3}$$

where $u = n_{\mathsf{A}} - \tau, \ldots, n_{\mathsf{A}} - 1$, the second term in the summation is the piggyback, and the superscript p in $p_{i,u}^{\mathsf{A,p}}$ indicates that the parity symbol contains piggybacks.

The fault tolerance (i.e., the number of node failures that can be tolerated) of Class A codes is given in the following theorem.

Theorem 1. *An (n_{A}, k) Class* A *code with τ piggybacks per row can tolerate*

$$f = \begin{cases} n_{\mathsf{A}} - k - \tau + \left\lfloor \frac{\sqrt{(n_{\mathsf{A}}-k-\tau)^2+4k}-(n_{\mathsf{A}}-k-\tau)}{2} \right\rfloor & \text{if } \tau \geq \xi \\ n_{\mathsf{A}} - k & \text{if } \tau < \xi \end{cases}$$

node failures, where $\xi = \frac{\sqrt{(n_{\mathsf{A}}-k-\tau)^2+4k}-(n_{\mathsf{A}}-k-\tau)}{2}$.

Proof. See Appendix A. □

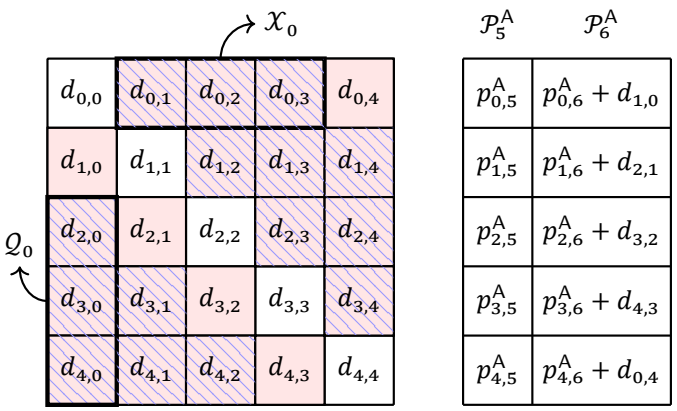

Figure I.2: A $(7, 5)$ Class A code with $\tau = 1$ constructed from a $(7, 5)$ MDS code. $\mathcal{P}_5^{\mathsf{A}}$ and $\mathcal{P}_6^{\mathsf{A}}$ are the parity nodes. For each row j, colored symbols belong to $\mathcal{D}_{\mathcal{R}_j}$.

We remark that for $\tau < \xi$, Class A codes are MDS codes.

When a failure of a data node occurs, Class A parity nodes are used to repair $\tau + 1$ of the k failed symbols. Class A parity symbols are constructed in such a way that, when node j is erased, $\tau + 1$ data symbols in this node can be repaired reading the (non-failed) $k - 1$ data symbols in the j-th row of the data array and $\tau + 1$ parity symbols in the j-th row of Class A parity nodes (see also Section 4.3). For later use, we define the set $\mathcal{R}_j$ as follows.

Definition 7. *For $j = 0, \ldots, k - 1$, the index set $\mathcal{R}_j$ is defined as*

$$\mathcal{R}_j = \{(j, (j+1)_k), (j, (j+2)_k), \ldots, (j, (j+k-1)_k)\}.$$

Then, the set $\mathcal{D}_{\mathcal{R}_j}$ is the set of $k - 1$ data symbols that are read from row j to recover $\tau + 1$ data symbols of node j using Class A parity nodes.

Example 1. *An example of a Class* A *code is shown in Fig. I.2. One can verify that the code can correct any* 2 *node failures. For each row j, the set $\mathcal{D}_{\mathcal{R}_j}$ is indicated in red color. For instance, $\mathcal{D}_{\mathcal{R}_0} = \{d_{0,1}, d_{0,2}, d_{0,3}, d_{0,4}\}$.*

The main purpose of Class A parity nodes is to provide good erasure correcting capability. However, the use of piggybacks helps also in reducing the number of symbols that need to be read to repair the $\tau + 1$ symbols of a failed node that are repaired using the Class A code, as compared to MDS codes. The remaining $k - \tau - 1$ data symbols of the failed node can also be recovered from Class A parity nodes, but at a high symbol read cost of k. Hence, the idea is to add another class of parity nodes, namely Class B parity nodes, in such a way that these symbols can be recovered with lower read cost.

[2] We use the superscript A to indicate that the parity symbol is stored in a Class A parity node.

4 Class B Parity Nodes

Class B parity nodes are obtained using an (n_{B}, k) linear block code with $n_{\mathsf{B}} < 2k-\tau$ over GF(q) to encode the $k \times k$ data symbols of the data array. This generates $k(n_{\mathsf{B}} - k)$ Class B parity symbols, $p^{\mathsf{B}}_{i,l}$, $i = 0, \ldots, k-1$, $l = n_{\mathsf{A}}, \ldots, n-1$. In [22], we presented an algorithm to construct Class B codes. In this section, we present a similar construction in a much more compact, mathematical manner.

4.1 Definitions and Preliminaries

Definition 8. *For $j = 0, \ldots, k-1$, the index set $\mathcal{Q}_j$ is defined as*

$$\mathcal{Q}_j = \{((j+\tau+1)_k, j), ((j+\tau+2)_k, j), \ldots, ((j+k-1)_k, j)\}.$$

Assume that data node j fails. It is easy to see that the set $\mathcal{D}_{\mathcal{Q}_j}$ is the set of $k-\tau-1$ data symbols that are not recovered using Class A parity nodes.

Example 2. *For the example in Fig. I.2, the set $\mathcal{D}_{\mathcal{Q}_j}$ is indicated by hatched symbols for each column j, $j = 0, \ldots, k-1$. For instance, $\mathcal{D}_{\mathcal{Q}_0} = \{d_{2,0}, d_{3,0}, d_{4,0}\}$.*

For later use, we also define the following set.

Definition 9. *For $j = 0, \ldots, k-1$, the index set $\mathcal{X}_j$ is defined as*

$$\mathcal{X}_j = \{(j, (j+1)_k), (j, (j+2)_k), \ldots, (j, (j+k-\tau-1)_k)\}.$$

Note that $\mathcal{X}_j = \mathcal{R}_j \cap \{\cup_{l=0}^{k-1} \mathcal{Q}_l\}$.

Example 3. *For the example in Fig. I.2, the set $\mathcal{D}_{\mathcal{X}_j}$ is indicated by hatched symbols for each row j. For instance, $\mathcal{X}_0 = \mathcal{R}_0 \cap \{\mathcal{Q}_0 \cup \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \mathcal{Q}_3 \cup \mathcal{Q}_4\} = \{(0,1), (0,2), (0,3)\}$, thus we have $\mathcal{D}_{\mathcal{X}_0} = \{d_{0,1}, d_{0,2}, d_{0,3}\}$.*

The purpose of Class B parity nodes is to allow the recovery of the data symbols in $\mathcal{D}_{\mathcal{Q}_j}$, $j = 0, \ldots, k-1$, at a low additional read cost. Note that after recovering $\tau+1$ symbols using Class A parity nodes, the data symbols in the sets $\mathcal{D}_{\mathcal{R}_j}$ are already stored in the decoder memory. Therefore, they are accessible for the recovery of the remaining $k-\tau-1$ data symbols using Class B parity nodes without the need of reading them again. The main idea is based on the following proposition.

Proposition 1. *If a Class B parity symbol p^{B} is the sum of one data symbol $d \in \mathcal{D}_{\mathcal{Q}_j}$ and a number of data symbols in $\mathcal{D}_{\mathcal{X}_j}$, then the recovery of d comes at the cost of one additional read (one should read parity symbol p^{B}).*

This observation is used in the construction of Class B parity nodes in Section 4.2 below to reduce the normalized repair bandwidth λ. In particular, we add up to $k-\tau-1$ Class B parity nodes which allow to reduce the additional read cost of all $k(k-\tau-1)$ data symbols in all $\mathcal{D}_{\mathcal{Q}_j}$'s to 1. (The addition of a single Class B parity node allows to recover one new data symbol in each $\mathcal{D}_{\mathcal{Q}_j}$, $j = 0, \ldots, k-1$, at the cost of one additional read.)

$\mathcal{P}_7^{\mathsf{B}}$	$\mathcal{P}_8^{\mathsf{B}}$	$\mathcal{P}_9^{\mathsf{B}}$
$d_{2,0}+d_{0,1}+d_{0,2}$	$d_{3,0}+d_{0,1}$	$d_{4,0}$
$d_{3,1}+d_{1,2}+d_{1,3}$	$d_{4,1}+d_{1,2}$	$d_{0,1}$
$d_{4,2}+d_{2,3}+d_{2,4}$	$d_{0,2}+d_{2,3}$	$d_{1,2}$
$d_{0,3}+d_{3,4}+d_{3,0}$	$d_{1,3}+d_{3,4}$	$d_{2,3}$
$d_{1,4}+d_{4,0}+d_{4,1}$	$d_{2,4}+d_{4,0}$	$d_{3,4}$

Figure I.3: Class B parity nodes for the data nodes in Fig. I.2.

4.2 Construction of Class B Nodes

For $t = 0, \ldots, k-1$, each parity symbol in the l-th Class B parity node, $l = n_{\mathsf{A}}, \ldots, n-1$, is sequentially constructed as

$$p_{t,l}^{\mathsf{B}} = d_{(\tau+1-n_{\mathsf{A}}+l+t)_k,t} + \sum_{j=0}^{k-\tau-3+n_{\mathsf{A}}-l} d_{t,(1+j+t)_k}. \tag{I.4}$$

The construction above follows Proposition 1 wherein $d_{(\tau+1-n_{\mathsf{A}}+l+t)_k,t} \in \mathcal{D}_{\mathcal{Q}_t}$ and $\{d_{t,(1+j+t)_k}\}_{j=0}^{k-\tau-3+n_{\mathsf{A}}-l} \subset \mathcal{D}_{\mathcal{X}_t}$. This ensures that the read cost of each of the k symbols $d_{(\tau+1-n_{\mathsf{A}}+l+t)_k,t}$ is 1. Thus, the addition of each parity node leads to k data symbols to have a read cost of 1. Note that adding the second term in (I.4) ensures that $k(k-\tau-1)$ data symbols are repaired by the Class B parity nodes. The same principle was used in [22]. It should be noted that the set of data symbols used in the construction of the parity symbols in (I.4) may be different compared to the construction in [22]. However, the overall average repair bandwidth remains the same.

Remark 1. *For the particular case $n_{\mathsf{B}} - k = k - \tau - 1$ one may neglect the second term in* (I.4)*. The resulting codes would still have the same repair bandwidth and lower repair complexity than the codes built from* (I.4)*. However, this construction would not allow rate-compatible Class* B *codes.*

In the sequel, we will refer to the construction of Class B parity nodes according to (I.4) as Construction 1.

Example 4. *With the aim to construct a* $(10, 5)$ *code, consider the construction of an* $(8, 5)$ *Class* B *code where the* $(7, 5)$ *Class* A *code, with* $\tau = 1$*, is as shown in Fig. I.2. For* $t = 0, \ldots, k-1$*, the parity symbols in the first Class* B *parity node (the* 7*-th node) are*

$$p_{t,7}^{\mathsf{B}} = d_{(2+t)_5,t} + \sum_{j=0}^{1} d_{t,(1+j+t)_5} = d_{(2+t)_5,t} + d_{t,(1+t)_5} + d_{t,(2+t)_5}.$$

The constructed parity symbols are as seen in Fig. I.3, where the t-th row in node $\mathcal{P}_7^{\mathsf{B}}$ contains the parity symbol $p_{t,7}^{\mathsf{B}}$. Notice that $d_{(2+t)_5,t} \in \mathcal{D}_{\mathcal{Q}_t}$ and $\{d_{t,(1+t)_5}, d_{t,(2+t)_5}\} \subset \mathcal{D}_{\mathcal{X}_t}$. In a similar way, the parity symbols in nodes $\mathcal{P}_8^{\mathsf{B}}$ and $\mathcal{P}_9^{\mathsf{B}}$ are

$$p_{t,8}^{\mathsf{B}} = d_{(3+t)_5,t} + \sum_{j=0}^{0} d_{t,(1+j+t)_5} = d_{(3+t)_5,t} + d_{t,(1+t)_5}$$

and

$$p_{t,9}^{\mathsf{B}} = d_{(4+t)_5,t} + \sum_{j=0}^{-1} d_{t,(1+j+t)_5} = d_{(4+t)_5,t},$$

respectively.

Consider the repair of the first data node in Fig. I.2. The symbol $d_{0,0}$ is reconstructed using $p_{0,5}^{\mathsf{A}}$. This requires reading the symbols $d_{0,1}$, $d_{0,2}$, $d_{0,3}$, and $d_{0,4}$. Since $p_{0,6}^{\mathsf{A}}$ is a function of all data symbols in the first row, reading $p_{0,6}^{\mathsf{A}} + d_{1,0}$ is sufficient for the recovery of $d_{1,0}$. From Fig. I.3, the symbols $d_{2,0}$, $d_{3,0}$, and $d_{4,0}$ can be recovered by reading just the parities $d_{2,0} + d_{0,1} + d_{0,2}$, $d_{3,0} + d_{0,1}$, and $d_{4,0}$, respectively. Thus, reading $5 + 4 = 9$ symbols is sufficient to recover all the symbols in the node, and the normalized repair bandwidth is $9/5 = 1.8$ per failed symbol. A more formal repair procedure is presented in Section 4.3.

Adding $n_{\mathsf{B}} - k$ Class B parity nodes allows to reduce the additional read cost of $n_{\mathsf{B}} - k$ data symbols from each $\mathcal{D}_{\mathcal{Q}_j}$, $j = 0, \ldots, k-1$, to 1. However, this comes at the cost of a reduction in the code rate, i.e., the storage overhead is increased. In the above example, adding $n_{\mathsf{B}} - k = 3$ Class B parity nodes leads to the reduction in code rate from $R = 5/7$ to $R = 5/10 = 1/2$. If a lower storage overhead is required, Class B parity nodes can be *punctured*, starting from the last parity node (for the code in Example 4, nodes $\mathcal{P}_9^{\mathsf{B}}$, $\mathcal{P}_8^{\mathsf{B}}$, and $\mathcal{P}_7^{\mathsf{B}}$ can be punctured in this order), at the expense of an increased repair bandwidth. If all Class B parity nodes are punctured, only Class A parity nodes would remain, and the repair bandwidth is equal to the one of the Class A code. Thus, our code construction gives a family of rate-compatible codes which provides a tradeoff between repair bandwidth and storage overhead: adding more Class B parity nodes reduces the repair bandwidth, but also increases the storage overhead.

4.3 Repair of a Single Data Node Failure: Decoding Schedule

The repair of a failed data node proceeds as follows. First, $\tau + 1$ symbols are repaired using Class A parity nodes. Then, the remaining symbols are repaired using Class B parity nodes. With a slight abuse of language, we will refer to the repair of symbols using Class A and Class B parity nodes as the decoding of Class A and Class B codes, respectively.

We will need the following definition.

Definition 10. *Consider a Class* B *parity node and let* $\mathcal{P}^{\mathsf{B}}$ *denote the set of parity symbols in this node. Also, let* $d \in \mathcal{D}_{\mathcal{Q}_j}$ *for some* j *and* $p^{\mathsf{B}} \in \mathcal{P}^{\mathsf{B}}$ *be the parity symbol* $p^{\mathsf{B}} = d + \sum_{d' \in \mathcal{D}'} d'$*, where* $\mathcal{D}' \subset \mathcal{D}$*, i.e., the parity symbol* p^{B} *is the sum of* d *and a subset of other data symbols. We define* $\breve{\mathcal{D}} = \mathcal{D}' \cup \{d\}$.

Suppose that node j fails. Decoding is as follows.

- **Decoding the Class** A **code**. To reconstruct the failed data symbol in the j-th row of the code array, k symbols ($k-1$ data symbols and $p^{\mathsf{A}}_{j,k}$) in the j-th row are read. These symbols are now cached in the memory. We then read the τ piggybacked symbols in the j-th row. By construction (see (I.3)), this allows to repair τ failed symbols, at the cost of an additional read each.
- **Decoding the Class** B **code**. Each remaining failed data symbol $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_j}$ is obtained by reading a Class B parity symbol whose corresponding set $\breve{\mathcal{D}}$ (see Definition 10) contains $d_{i,j}$. In particular, if several Class B parity symbols $p^{\mathsf{B}}_{i',j'}$ contain $d_{i,j}$, we read the parity symbol with largest index j'. This yields the lowest additional read cost.

5 A Heuristic Construction of Class B Nodes With Improved Repair Bandwidth

In this section, we provide a way to improve the repair bandwidth of the family of codes constructed so far. More specifically, we achieve this by providing a heuristic algorithm for the construction of the Class B code, which improves Construction 1 in Section 4 for some values of n and even values of k.

The algorithm is based on a simple observation. Let p^{B}_1 and p^{B}_2 be two parity symbols constructed from ρ data symbols in $\mathcal{D}$ in two different ways as follows:

$$p^{\mathsf{B}}_1 = d_{i,j} + d_{j,i} + d_{j,i_2} + \cdots + d_{j,i_{\rho-1}}, \tag{I.5}$$

$$p^{\mathsf{B}}_2 = d_{i,j} + d_{j,i_1} + d_{j,i_2} + \cdots + d_{j,i_{\rho-1}}, \tag{I.6}$$

where $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_j}$ (see Definition 8), $i_1, \ldots, i_{\rho-1} \neq i$, and $d_{j,i_1}, d_{j,i_2}, \ldots, d_{j,i_{\rho-1}} \in \mathcal{D}_{\mathcal{X}_j}$ (see Definition 9). Note that the only difference between the two parity symbols above is that p^{B}_2 does not involve $d_{j,i}$ (and that p^{B}_1 does not involve d_{j,i_1}). This has a major consequence in the repair of the data symbols $d_{i,j}, d_{j,i}, \ldots, d_{j,i_\rho-1}$ and $d_{i,j}, d_{j,i_1}, \ldots, d_{j,i_\rho-1}$ using p^{B}_1 and p^{B}_2, respectively. Consider the repair using parity symbol p^{B}_1. From Proposition 1, it is clear that the repair of symbol $d_{i,j}$ will have an additional read cost of 1, since the remaining $\rho-1$ data symbols are in $\mathcal{D}_{\mathcal{X}_j}$. As the symbol $d_{j,i} \in \mathcal{D}_{\mathcal{Q}_i}$ and $d_{i,j} \in \mathcal{D}_{\mathcal{X}_i}$, from Proposition 1 and the fact that $d_{j,i_2}, \ldots, d_{j,i_{\rho-1}} \notin \mathcal{D}_{\mathcal{X}_i}$, we can repair $d_{j,i}$ with an additional read cost of $\rho-1$. The remaining $\rho-2$ symbols each have an additional read cost of ρ, whereas the symbols repaired using p^{B}_2 incur an additional read cost of 1 for the symbol $d_{i,j}$ and ρ for the remaining symbols. Clearly, we see that the combined additional read cost, i.e., the sum of the individual additional read costs for each data symbol using p^{B}_1 is lower (by 1) than that using p^{B}_2.

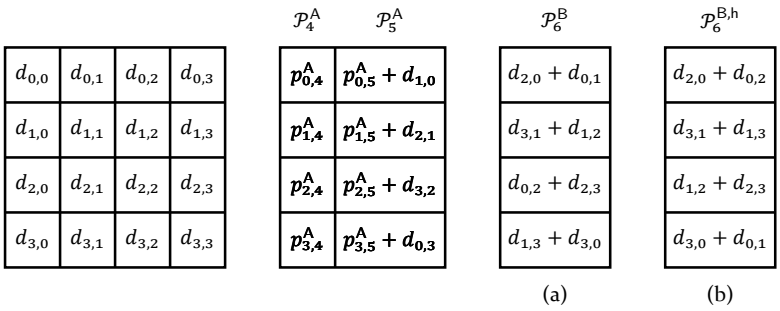

Figure I.4: A $(7, 4)$ code constructed from a $(6, 4)$ Class A code with $\tau = 1$ and a $(5, 4)$ Class B code. (a) Class B node constructed according to Construction 1 in Section 4. (b) A different configuration of the Class B node that reduces the repair bandwidth.

In the way Class A parity nodes are constructed and due to the structure of the sets $\mathcal{D}_{\mathcal{Q}_j}$ and $\mathcal{D}_{\mathcal{X}_j}$, it can be seen that $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_j}$ and $d_{j,i} \in \mathcal{D}_{\mathcal{X}_j}$ when $k \geq 2(\tau + 1)$. From Construction 1 of the Class B code in Section 4 we observe that for odd k and $k > 2(\tau + 1)$, the parity symbols in node $\mathcal{P}_l^{\mathsf{B}}$ are as in (I.5) for $n_{\mathsf{A}} \leq l \leq n_{\mathsf{A}} + \lfloor k/2 \rfloor - \tau - 1$. Furthermore, for $n_{\mathsf{A}} + \lfloor k/2 \rfloor - \tau \leq l \leq n - 1$, the parity symbols in node $\mathcal{P}_l^{\mathsf{B}}$ have the structure in (I.6). On the other hand, for k even and $k \geq 2(\tau + 1)$, the parity symbols in the node $\mathcal{P}_l^{\mathsf{B}}$ are as in (I.5) for $n_{\mathsf{A}} \leq l \leq n_{\mathsf{A}} + k/2 - \tau - 2$. However, contrary to case of k odd, the parity symbols in the node $\mathcal{P}_{n_{\mathsf{A}}+k/2-\tau-1}^{\mathsf{B}}$ follow (I.6). But since $k \geq 2(\tau+1)$, we know that $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_j}$ and $d_{j,i} \in \mathcal{D}_{\mathcal{X}_j}$. Thus, it is possible to construct some parity symbols in this node as in (I.5), and Construction 1 of Class B nodes in the previous section can be improved. However, the improvement comes at the expense of the loss of the mathematical structure of Class B nodes given in (I.4).

Example 5. *Consider the* $(7, 4)$ *code as shown in Fig. I.4. Fig. I.4(a) shows the node* $\mathcal{P}_6^{\mathsf{B}}$ *using Construction* 1 *in Section 4, while Fig. I.4(b) shows a different configuration of the node* $\mathcal{P}_6^{\mathsf{B}}$*. Note that* $k = 2(\tau + 1) = 4$*. Thus, each pair* $(\mathcal{D}_{\mathcal{Q}_j}, \mathcal{D}_{\mathcal{X}_j})$ *contains one symbol* $d_{i,j}$ *and* $d_{j,i}$*. The node* $\mathcal{P}_6^{\mathsf{B}}$ *has parity symbols according to* (I.6)*, while* $\mathcal{P}_6^{\mathsf{B,h}}$ *has two parity symbols as in* (I.5) *and two parity symbols according to* (I.6)*. The configuration of the* $(7, 4)$ *code arising from Construction* 1 *has a normalized repair bandwidth of* 2*, while the* $(7, 4)$ *code with node* $\mathcal{P}_6^{\mathsf{B,h}}$ *in Fig. I.4(b) has a repair bandwidth of* 1.875*, i.e., an improvement is achieved.*

In order to describe the modified code construction, we define the function $\text{read}(d, p^{\mathsf{B}})$ as follows.

Definition 11. *Consider the construction of the parity symbol* p^{B} *as* $p^{\mathsf{B}} = d + \sum_{d' \in \mathcal{D}'} d'$ *(see Definition 10). Then,*

$$\text{read}(d, p^{\mathsf{B}}) = |\check{\mathcal{D}} \backslash \mathcal{D}_{\mathcal{X}_j}|.$$

For a given data symbol d, the function $\text{read}(d, p^{\mathsf{B}})$ gives the additional number of symbols that need to be read to recover d (considering the fact that some

symbols are already cached in the memory). The set $\check{\mathcal{D}}$ represents the set of data symbols that the parity symbol p^{B} is a function of. We use the index set $\mathcal{U}$ to represent the indices of such data symbols. We denote by $\mathcal{U}_t, t = 0, \ldots, k-1$, the index set corresponding to the t-th parity symbol in the node (there are k parity symbols in a parity node).

In the following, denote by $\boldsymbol{A} = [a_{i,j}]$ a temporary matrix of read costs for the respective data symbols in $\boldsymbol{D} = [d_{i,j}]$. After Class A decoding,

$$a_{i,j} = \begin{cases} \infty & \text{if } d_{i,j} \in \cup_{t=0}^{k-1} \mathcal{D}_{\mathcal{Q}_t} \\ k & \text{if } i = j \\ 1 & \text{otherwise} \end{cases}. \tag{I.7}$$

In Section 5.1 below, we will show that the construction of parities depends upon the entries of $\boldsymbol{A}$. To this extent, for some real matrix $\boldsymbol{M} = [m_{i,j}]$ and index set $\mathcal{I}$, we define $\Psi(\boldsymbol{M}_{\mathcal{I}})$ as the set of indices of matrix elements of $\boldsymbol{M}$ from $\mathcal{I}$ whose values are equal to the maximum of all entries in $\boldsymbol{M}$ indexed by $\mathcal{I}$. More formally, $\Psi(\boldsymbol{M}_{\mathcal{I}})$ is defined as

$$\Psi(\boldsymbol{M}_{\mathcal{I}}) = \left\{ (i,j) \in \mathcal{I} \mid m_{i,j} = \max_{(i',j') \in \mathcal{I}} m_{i',j'} \right\}.$$

The heuristic algorithm to construct the Class B code is given in Appendix B and we will refer to the construction of the Class B code according to this algorithm as Construction 2. In the following subsection, we clarify the heuristic algorithm to construct the Class B code with the help of a simple example.

5.1 Construction Example

Let us consider the construction of a $(7, 4)$ code using a $(6, 4)$ Class A code and a $(5, 4)$ Class B code. In total, there are three parity nodes; two Class A parity nodes, denoted by $\mathcal{P}_4^{\mathsf{A}}$ and $\mathcal{P}_5^{\mathsf{A}}$, respectively, and one Class B parity node, denoted by $\mathcal{P}_6^{\mathsf{B,h}}$, where the upper index h is used to denote that the parity node is constructed using the heuristic algorithm. The parity symbols of the nodes are depicted in Fig. I.4. Each parity symbol of the Class B parity node is the sum of $k - \tau - 1$ data symbols $d_{i,j} \in \cup_{j'} \mathcal{D}_{\mathcal{Q}_{j'}}$, constructed such that the read cost of each symbol $d_{i,j}$ is lower than $a_{i,j}$ as shown below.

1. **Construction of $\mathcal{P}_6^{\mathsf{B,h}}$**
   Each parity symbol in this node is constructed using $\rho_6 = k - \tau - 1 = 2$ unique symbols as follows.

   1.a Since no symbols have been constructed yet, we have $\mathcal{U}_{t_1} = \emptyset$, $t_1 = 0, \ldots, 3$. (This corresponds to the execution of Line 1 to Line 19 of Algorithm 2 in Appendix B.)

   1.b Select $d_{i,0} \in \mathcal{D}_{\mathcal{Q}_0}$ such that its read cost is maximum, i.e., $d_{i,0} \in \mathcal{D}_{\Psi(\boldsymbol{A}_{\mathcal{Q}_0})}$. Choose $d_{i,0} = d_{2,0}$, as $a_{2,0} = \infty$. Note that we choose $d_{2,0}$ since $d_{0,2} \in \mathcal{D}_{\mathcal{X}_0}$.

1.c Construct $p_{0,6} = d_{2,0} + d_{0,2}$ (see Line 4 of Algorithm 2). Correspondingly, we have $\mathcal{U}_0 = \{(2,0),(0,2)\}$.

1.d Recursively construct the next parity symbol in the node as follows. Similar to Item 1.b, choose $d_{3,1} \in \mathcal{D}_{\Psi(\boldsymbol{A}_{\mathcal{Q}_1})}$. Construct $p_{1,6} = d_{3,1} + d_{1,3}$. Likewise, we have $\mathcal{U}_1 = \{(3,1),(1,3)\}$

1.e For the next parity symbol, note that $d_{0,2}$ is already used in the construction of $p_{0,6}$. The only possible choice of symbol in $\mathcal{D}_{\mathcal{Q}_2}$ is $d_{1,2}$, but $d_{2,1} \notin \mathcal{D}_{\mathcal{X}_2}$. Therefore, we choose $d_{i_1,2} \in \mathcal{D}_{\Psi(\boldsymbol{A}_{\mathcal{Q}_2 \setminus \cup_{j'} \mathcal{U}_{j'}})}$ (see Line 7 of Algorithm 2). In particular, since $a_{1,2} = \infty$, we choose $d_{i_1,2} = d_{1,2}$. Then, Lines 8 to 11 of Algorithm 2 are executed.

1.f Choose an element $d_{2,i_2} \in \mathcal{D}_{\mathcal{X}_2 \setminus \cup_{j'} \mathcal{U}_{j'}}$. In other words choose a symbol in $\mathcal{D}_{\mathcal{X}_2}$ which has not been used in $p_{0,6}$ and $p_{1,6}$. We have $d_{2,i_2} = d_{2,3}$. Construct $p_{2,6} = d_{1,2} + d_{2,3}$. Thus, $\mathcal{U}_2 = \{(1,2),(2,3)\}$.

1.g To construct the last parity symbol, we look for data symbols from the sets $\mathcal{D}_{\mathcal{Q}_3}$ and $\mathcal{D}_{\mathcal{X}_3}$. However, all symbols in $\mathcal{D}_{\mathcal{Q}_3}$ have been used in the construction of previous parity symbols. Therefore, we cyclically shift to the next pair of sets $(\mathcal{D}_{\mathcal{Q}_0}, \mathcal{D}_{\mathcal{X}_0})$. Following Items 1.e and 1.f, we have $p_{3,6} = d_{3,0} + d_{0,1}$ and $\mathcal{U}_3 = \{(3,0),(0,1)\}$.

Note that $|\mathcal{U}_t| = 2$ for all t, thus this completes the construction of the $(7,4)$ code. The Class B parity node constructed above is depicted in Fig. I.4(b).

5.2 Discussion

In general, the algorithm constructs $n_{\mathsf{B}} - k$ parity nodes, $\mathcal{P}^{\mathsf{B}}_{n_{\mathsf{A}}}, \ldots, \mathcal{P}^{\mathsf{B}}_{n-1}$, recursively. In the l-th Class B node, $l = n_{\mathsf{A}}, \ldots, n-1$, each parity symbol is a sum of at most $\rho_l = k - \tau - 1 - l + n_{\mathsf{A}}$ symbols $d_{i,j} \in \cup_{j'} \mathcal{D}_{\mathcal{Q}_{j'}}$. Each parity symbol $p_{t,l}$, $t = 0, \ldots, k-1$, in the l-th Class B parity node with $\rho_l > 1$ is constructed recursively with $\rho_l - 1$ recursion steps. In the first recursion step, each parity symbol $p_{t,l}$ is either equal to a single data symbol or a sum of 2 data symbols. In the latter case, the first symbol $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_t}$ is chosen as the symbol with the largest read cost $a_{i,j}$. The second symbol is $d_{j,i} \in \mathcal{D}_{\mathcal{X}_t}$ if such a symbol exists. Otherwise (i.e., if $d_{j,i} \notin \mathcal{D}_{\mathcal{X}_t}$), symbol $d_{j,i''} \in \mathcal{D}_{\mathcal{X}_t}$ is chosen. In the remaining $\rho_l - 2$ recursion steps a subsequent data symbol $d_{j,i'} \in \mathcal{D}_{\mathcal{X}_t}$ (if it exists) is added to $p_{t,l}$. Doing so ensures that k symbols have a new read cost that is reduced to 1 when parity symbols $p_{t,l}$ are used to recover them. Having obtained these parity symbols, the read costs of all data symbols in $\cup_{j'} \mathcal{D}_{\mathcal{Q}_{j'}}$ are updated and stored in $\boldsymbol{A}$. This process is repeated for successive parity nodes. If $\rho_l = 1$ for the l-th parity node, its parity symbols $p_{t,l}$ are equal to the data symbols $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_t}$ whose read costs $a_{i,j}$ are the maximum possible.

In the above example, only a single recursion for the construction of $\mathcal{P}^{\mathsf{B,h}}_6$ is needed, where each parity symbol is a sum of two data symbols.

6 Code Characteristics and Comparison

In this section, we characterize different properties of the codes presented in Sections 3-5. In particular, we focus on the fault tolerance, repair bandwidth, repair complexity, and encoding complexity. We consider the complexity of elementary arithmetic operations on field elements of size $\nu = m\lceil \log_2 p \rceil$ in GF(q), where $q = p^m$ for some prime number p and positive integer m. The complexity of addition is $O(\nu)$, while that of multiplication is $O(\nu^2)$, where the argument of $O(\cdot)$ denotes the number of elementary binary additions.[3]

6.1 Code Rate

The code rate for the proposed codes is given by $R = \frac{k}{n_\mathsf{A}+n_\mathsf{B}-k}$. It can be seen that the code rate is bounded as

$$\frac{k}{3k-\tau-2} \leq R \leq \frac{k}{k+3}.$$

The upper bound is achieved when $n_\mathsf{A} = k+2$, $\tau = 1$, and $n_\mathsf{B} = k+1$, while the lower bound is obtained from the upper bounds on n_A and n_B given in Sections 3 and 4.

6.2 Fault Tolerance

The proposed codes have fault tolerance equal to that of the corresponding Class A codes, which depends on the MDS code used in their construction and τ (see Theorem 1). Class B nodes do not help in improving the fault tolerance. The reason is that improving the fault tolerance of the Class A code requires the knowledge of the piggybacks that are strictly not in the set $\cup_j \mathcal{D}_{Q_j}$, while Class B nodes can only be used to repair symbols in $\cup_j \mathcal{D}_{Q_j}$.

In the case where the Class B code has parameters $(n_\mathsf{B} = k+1, k)$, the resulting $(n = n_\mathsf{A}+1, k)$ code has fault tolerance $n_\mathsf{A} - k$ for $\tau < \xi$, i.e., one less than that of an $(n = n_\mathsf{A}+1, k)$ MDS code.

6.3 Repair Bandwidth of Data Nodes

According to Section 4.3, to repair the first $\tau+1$ symbols in a failed node, $k-1$ data symbols and $\tau+1$ Class A parity symbols are read. The remaining $k-\tau-1$ data symbols in the failed node are repaired by reading Class B parity symbols.

Let f_l, $l = n_\mathsf{A}, \ldots, n-1$, denote the number of parity symbols that are used from the l-th Class B node according to the decoding schedule in Section 4.3. Due to

[3]It should be noted that the complexity of multiplication is quite pessimistic. However, for the sake of simplicity we assume it to be $O(\nu^2)$. When the field is GF(2^ν) there exist algorithms such as the Karatsuba-Ofman algorithm [26, 27] and the Fast Fourier Transform [28–30] that lower the complexity to $O(\nu^{\log_2 3})$ and $O(\nu \log_2 \nu)$, respectively.

Construction 1 in Section 4, we have

$$\begin{aligned} f_{n_{\mathsf{A}}} &= f_{n_{\mathsf{A}}+1} = \cdots = f_{n-2} = 1, \\ f_{n-1} &= k - \tau - 1 - \sum_{l=n_{\mathsf{A}}}^{n-2} f_l = k - \tau - n + n_{\mathsf{A}}. \end{aligned} \tag{I.8}$$

The Class B nodes $n_{\mathsf{A}}, \ldots, n-2$ are used to repair $n-1-n_{\mathsf{A}}$ symbols with an additional read cost of $n-1-n_{\mathsf{A}}$ (1 per symbol). The remaining $k-\tau-n+n_{\mathsf{A}} = f_{n-1}$ erased symbols are corrected using the $(n-1)$-th Class B node. The repair of one of the f_{n-1} symbols entails an additional read cost of 1. On the other hand, since the parity symbols in the $(n-1)$-th Class B node are a function of f_{n-1} symbols, the repair of the remaining $f_{n-1}-1$ symbols entails an additional read cost of at most f_{n-1} each. In all, the $k-\tau-1$ erased symbols in the failed node have a total additional read cost of at most $n-n_{\mathsf{A}}+(f_{n-1}-1)f_{n-1}$. The normalized repair bandwidth for the failed systematic node is therefore given as

$$\lambda^{\mathsf{s}} \leq \frac{k+\tau+n-n_{\mathsf{A}}+(f_{n-1}-1)f_{n-1}}{k} = \frac{2k-2f_{n-1}+f_{n-1}^2}{k}.$$

Note that f_{n-1} is function of τ, and it follows from Section 6.2 and (I.8) that when τ increases, the fault tolerance reduces while λ^{s} improves. Furthermore, as n_{B} increases (thereby as n increases), f_{n-1} decreases. This leads to a further reduction of the normalized repair bandwidth.

6.4 Repair Complexity of a Failed Data Node

To repair the first symbol requires k multiplications and $k-1$ additions. To repair the following τ symbols require an additional τk multiplications and additions. Thus, the repair complexity of repairing $\tau+1$ failed symbols is

$$C_{\mathsf{r}}^{\mathsf{A}} = O((k-1)v + kv^2) + O(\tau k(v+v^2)).$$

For Construction 1, the remaining $k-\tau-1$ failed data symbols in the failed node are corrected using $k-\tau-1$ parity symbols from $n_{\mathsf{B}}-k$ Class B nodes. To this extent, note that $\sum_{l=n_{\mathsf{A}}}^{n-1} f_l = k-\tau-1$. The repair complexity for repairing the remaining $k-\tau-1$ symbols is

$$C_{\mathsf{r}}^{\mathsf{B}} = \sum_{l=n_{\mathsf{A}}}^{n-1} O(f_l(k-\tau-2-l+n_{\mathsf{A}})v). \tag{I.9}$$

From (I.8), (I.9) simplifies to

$$C_{\mathsf{r}}^{\mathsf{B}} = \sum_{l=n_{\mathsf{A}}}^{n-2} O((k-\tau-2-l+n_{\mathsf{A}})v) + O((k-\tau-n+n_{\mathsf{A}})(k-\tau-1-n+n_{\mathsf{A}})v).$$

For Construction 2, the final $k-\tau-1$ failed data symbols require at most $k-\tau-2$ additions, since Class B parity symbols are constructed as sums of at most $k-\tau-1$ data symbols. The corresponding repair complexity is therefore

$$C_{\mathrm{r}}^{\mathsf{B}} \leq O((k-\tau-2)(k-\tau-1)v).$$

Finally, the total repair complexity is $C_{\mathrm{r}}^{\mathrm{s}} = C_{\mathrm{r}}^{\mathsf{A}} + C_{\mathrm{r}}^{\mathsf{B}}$.

6.5 Repair Bandwidth and Complexity of Parity Nodes

We characterize the normalized repair bandwidth and repair complexity of Class A and B parity nodes.

Class A nodes consist of $n_{\mathsf{A}}-k$ MDS parity nodes of which τ nodes are modified with a single piggyback. Thus, the repair of each parity symbol in the $n_{\mathsf{A}}-k-\tau$ non-modified nodes requires downloading k data symbols. To obtain the parity symbol, one needs to perform $k-1$ additions and k multiplications. Thus, each parity symbol in these nodes has a repair bandwidth of k and a repair complexity of $O((k-1)v+kv^2)$, while each erased parity symbol in the τ piggybacked nodes requires reading $k+1$ data symbols. Such parity symbols are obtained by performing $k-1$ additions and k multiplications to get the original MDS parity symbol and then finally a single addition of the piggyback to the MDS parity symbol is required. Overall, the normalized repair bandwidth is $k+1$ and the normalized repair complexity is $O(k(v+v^2))$. In average, the normalized repair bandwidth and the normalized repair complexity of Class A parity nodes are

$$\lambda^{\mathrm{p,A}} = k + \frac{\tau}{n_{\mathsf{A}}-k}, \quad C_{\mathrm{R}}^{\mathrm{p,A}} = O\left((k-1)v + kv^2 + \frac{\tau v}{n_{\mathsf{A}}-k}\right),$$

respectively.

Considering the i-th Class B node, the repair of an erased parity symbol requires downloading $k-\tau-1-i$, $i=0,\ldots,n_{\mathsf{B}}-k-1$, data symbols. The repair entails $k-\tau-2-i$ additions, and the average normalized repair bandwidth $\lambda^{\mathrm{p,B}}$ and repair complexity $C_{\mathrm{r}}^{\mathrm{p,B}}$ are given as

$$\lambda^{\mathrm{p,B}} = \frac{\sum_{i=0}^{n_{\mathsf{B}}-k-1} k-\tau-1-i}{n_{\mathsf{B}}-k} = \frac{1}{2}\left(3k-2\tau-n_{\mathsf{B}}-1\right),$$
$$C_{\mathrm{r}}^{\mathrm{p,B}} = O\left(\frac{\sum_{i=0}^{n_{\mathsf{B}}-k-1} k-\tau-2-i}{n_{\mathsf{B}}-k}v\right) = O\left(\frac{1}{2}(3k-2\tau-n_{\mathsf{B}}-3)v\right).$$

6.6 Encoding Complexity

The encoding complexity, denoted by C_{e}, is the sum of the encoding complexities of Class A and Class B codes. The generation of each of the $n_{\mathsf{A}}-k$ Class A parity symbols in one row of the code array, $p_{i,j}^{\mathsf{A}}$ in (I.2), requires k multiplications and $k-1$ additions. Adding data symbols to τ of these parity symbols according to

(I.3) requires an additional τ additions. The encoding complexity of the Class A code is therefore

$$C_{\mathrm{e}}^{\mathrm{A}} = O((n_{\mathrm{A}} - k)(kv^2 + (k-1)v)) + O(\tau v).$$

According to Sections 4 and 5, the parity symbols in the first Class B parity node are constructed as sums of at most $k - \tau - 1$ data symbols, and each parity symbol in the subsequent parity nodes is constructed as a sum of data symbols from a set of size one less. Therefore, the encoding complexity of the Class B code is

$$\begin{aligned} C_{\mathrm{e}}^{\mathrm{B}} &\leq \sum_{i=1}^{n-n_{\mathrm{A}}} O((k-\tau-1-i)v) \\ &= O\left(\frac{1}{2}(n-n_{\mathrm{A}})(2k-2\tau-3-n+n_{\mathrm{A}})v\right). \end{aligned} \tag{I.10}$$

Note that for Construction 1 the upper bound on $C_{\mathrm{e}}^{\mathrm{B}}$ in (I.10) is tight. Finally, $C_{\mathrm{e}} = C_{\mathrm{e}}^{\mathrm{A}} + C_{\mathrm{e}}^{\mathrm{B}}$.

6.7 Code Comparison

In this section, we compare the performance of the proposed codes with that of several codes in the literature, namely MDS codes, exact-repairable MDS codes [24], MDR codes [9], Zigzag codes [10], Piggyback codes [19], generalized Piggyback codes [23], EVENODD codes [25], Pyramid codes [2], and LRCs [3]. Throughout this section, we compare the repair bandwidth and the repair complexity of the systematic nodes with respect to other codes, except for exact-repairable MDS and BASIC PM-MBR codes. The reported repair bandwidth and complexity for these codes are for all nodes (both systematic and parity nodes).

Section 6.7 provides a summary of the characteristics of the proposed codes as well as different codes proposed in the literature.[4] In the table, column 2 reports the value of β (see (I.1)) for each code construction. For our code, $\beta = k$, unlike for MDR and Zigzag codes, for which β grows exponentially with k. This implies that our codes require less memory to cache data symbols during repair. On the contrary, EVENODD codes have a lower sub-packetization and repair complexity, but this comes at the cost of having the same repair bandwidth as MDS codes. The Piggyback codes presented in the table and throughout this section are from the piggybacking design 1 in [19], which provides efficient repair for only data nodes. The fault tolerance f, the normalized repair bandwidth λ, the normalized repair complexity, and the encoding complexity, discussed in the previous subsections, are reported in columns 3, 4, 5, and 6, respectively.

In Figs. I.5 and I.6, we compare our codes with Construction 1 for the Class B codes (i.e., the codes are constructed as shown in Sections 3 and 4) with other codes in the literature. We remark that the Pyramid codes in Fig. I.6 refer to

[4] The variables (t, t_r) and r in Section 6.7 are defined in [19] and [3], respectively. The definition of ℓ comes directly from r that is defined in [19].

	β	Fault Tolerance	Norm. Repair Band.	Norm. Repair Compl.	Enc. Complexity
MDS	1	$n-k$	k	$O((k-1)v+kv^2)$	$O((n-k)(k-1)v+kv^2)$
LRC [3]	1	$r+1$	$\frac{k}{n-k-r}$	$O((\lceil\frac{k}{n-k-r}\rceil-1)v)$	$rO((k-1)v+kv^2)+(n-k-r)O((\lceil\frac{k}{n-k-r}\rceil-1)v)$
MDR [9]	2^k	2	$\frac{k+1}{2}$	$O(k-1)$	$O(2(k-1))$
Zigzag [10]	$(n-k)^{k-1}$	$n-k$	$\frac{n-1}{n-k}$	$O((k-1)v+kv^2)$	$O((n-k)(k-1)v+kv^2)$
Piggyback [19]	2	$n-k$	$\frac{(k-t_r)(k+t)+t_r(k+t_r+\ell-2)}{2k}$	–	–
EVENODD[25]	$k-1$	2	k	$O((k-1)v)$	$O(\frac{2k^2-2k-1}{k-1}v)$
Proposed codes	k	f	λ^{s}	C_r^{s}/k	C_{e}

Table I.1: Comparison of (n,k) codes that aim at reducing the repair bandwidth. The repair bandwidth and the repair complexity are normalized per symbol, while the encoding complexity is given per row of the code array. Note that for MDR and EVENODD codes, $n=k+2$ where k is prime for EVENODD codes.

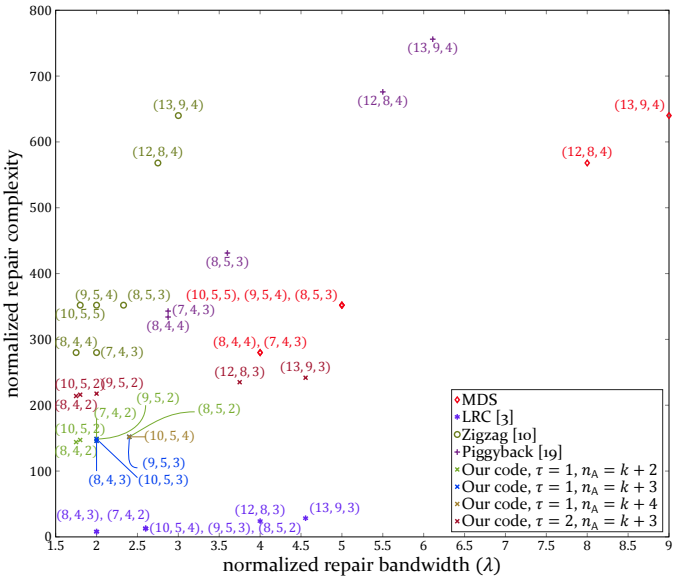

Figure I.5: Comparisons of different (n, k, f) codes with $\nu = 8$.

the basic Pyramid codes in [2], while the exact-repairable MDS codes refer to the $(2, n-k, n-1)$ exact-repairable MDS codes from [24, Sec. IV]. The aforementioned notation, unlike our notation in this paper, refers to an $(n, k, n-k)$ code that has $\lambda \leq 2\frac{n-1}{n-k}$, $\beta = n-k$, and repair locality of $n-1$. For generalized Piggyback codes [23], we choose $\beta = k$. Also note that the parameters s, p are chosen according to [23, Eq. 20], i.e., $s = \left\lfloor k\frac{\sqrt{n-k}-1}{\sqrt{n-k}} \right\rfloor$ or $s = \left\lceil k\frac{\sqrt{n-k}-1}{\sqrt{n-k}} \right\rceil$ and $p = k - s$, whichever pair of values gives the lowest repair bandwidth. In case of a tie, the pair that gives the lowest repair complexity was chosen. In particular, the figure plots the normalized repair complexity of (n, k, f) codes over GF(2^8) ($\nu = 8$) versus their normalized repair bandwidth λ. In the figure, we show the exact repair bandwidth for our proposed codes, while the reported repair complexities and the repair bandwidths of the other codes, except for Piggyback, generalized Piggyback, and exact-repairable MDS codes, are from Section 6.7.[5] For Piggyback, generalized Piggyback, and exact-repairable MDS codes exact values for the repair bandwidth and the repair complexity are calculated directly from the codes. Furthermore, for a fair comparison we assume the parity symbols in the first parity node of all storage codes to be weighted sums. The only exception is the LRCs and the exact-repairable MDS codes, as the code design enforces the parity-check equations to be non-weighted sums. Thus, changing it would alter the maximum erasure correcting capability of the LRC and the repair bandwidth of the exact-repairable MDS code. We also assume that the LRCs and the Pyramid codes have a repair locality of $k/2$. For the generalized Piggyback codes, we

[5] For LRCs the expressions for the repair bandwidth and the repair complexity tabulated in Section 6.7 are used when $n-k-r$ is a divisor of k. When $n-k-r$ is not a divisor of k, exact values for the repair bandwidth and the repair complexity are calculated directly from the codes.

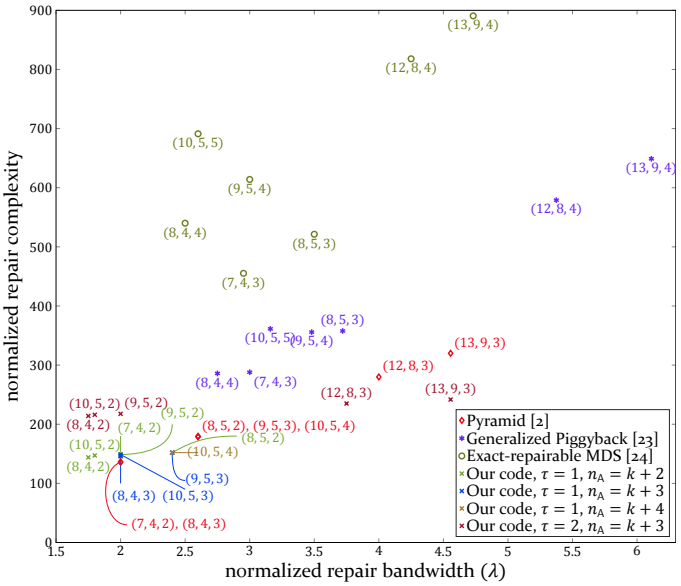

Figure I.6: Comparisons of different (n, k, f) codes with $\nu = 8$.

assume that the codes have sub-packetization $\beta = k$. For the Piggyback codes, we consider the construction that repairs just the data nodes. Therefore, they have a sub-packetization of 2.

The best codes for a DSS should be the ones that achieve the lowest repair bandwidth and have the lowest repair complexity. As seen in Fig. I.5, MDS codes have both high repair complexity and repair bandwidth, but they are optimal in terms of fault tolerance for a given n and k. Zigzag codes achieve the same fault tolerance and high repair complexity as MDS codes, but at the lowest repair bandwidth. At the other end, LRCs yield the lowest repair complexity, but a higher repair bandwidth and worse fault tolerance than Zigzag codes. Piggyback codes, generalized Piggyback codes, and exact-repairable MDS codes have a repair bandwidth between those of Zigzag and MDS codes, but with a higher repair complexity. Strictly speaking, they have a repair complexity higher than MDS codes. For a given storage overhead, our proposed codes have better repair bandwidth than MDS codes, Piggyback codes, generalized Piggyback codes, and exact-repairable MDS codes. In particular, the numerical results in Figs. I.5 and I.6 show that for different code parameters with $\tau = 1$ and $n_{\mathrm{A}} - k = 2$ our proposed codes yield a reduction of the repair bandwidth in the range of $64\% - 50\%$, $39.13\% - 30.43\%$, $43.04\% - 33.33\%$, and $33.33\% - 30\%$, respectively. Furthermore, our proposed codes yield lower repair complexity as compared to MDS, Piggyback, generalized Piggyback, exact-repairable MDS, and Zigzag codes. Again, the numerical analysis in Figs. I.5 and I.6 shows that for different code parameters with $\tau = 1$ and $n_{\mathrm{A}} - k = 2$ our proposed codes yield a reduction of the repair complexity in the range of $58.18\% - 47.86\%$, $64.75\% - 55.23\%$, $59.26\% - 49.30\%$, $78.70\% - 67.93\%$, and $58.18\% - 47.86\%$, respectively. However, the benefits in terms of repair bandwidth and/or repair complexity with respect to MDS codes, Zigzag codes, and codes constructed using the piggybacking framework come at

a price of a lower fault tolerance. For fixed (n, k, f) it can be seen that the proposed codes yield a reduction of the repair bandwidth in the range of $7.69\% - 0\%$ compared to LRCs and Pyramid codes, while in some cases, for the latter codes our proposed codes achieve a reduction of the repair complexity in the range of $24.44\% - 15.18\%$.

In Table I.2, we compare the normalized repair complexity of the proposed codes with Construction 1 for the Class B code and BASIC PM-MBR codes presented in [20]. BASIC PM-MBR codes are constructed from an algebraic ring $\mathcal{R}_m$, where each symbol in the ring is a binary vector of length m. In order to have a fair comparison with our codes, we take the smallest possible field size for our codes and the smallest possible ring size for the codes in [20]. Furthermore, to compare codes with similar storage overhead, we consider BASIC PM-MBR codes with repair locality, denoted by δ^{b}, that leads to the same file size as for our proposed codes (which is equal to k^2) and code length n such that the code rate (which is the inverse of the storage overhead) is as close as possible to that of our proposed codes. The Class A codes of the proposed codes in the table are $(n_{\mathsf{A}}, n_{\mathsf{A}} - f)$ RS codes. The codes are over $\mathrm{GF}(n_{\mathsf{A}} + 1)$ if $n_{\mathsf{A}} + 1$ is a prime (or a power of a prime). If $n_{\mathsf{A}} + 1$ is not a prime (or a power of prime), we construct an $(n', n' - f)$ RS code over $\mathrm{GF}(n' + 1)$, where $n' + 1$ is the smallest prime (or the smallest power of a prime) with $n' + 1 \geq n_{\mathsf{A}} + 1$, and then we shorten the RS code to obtain an $(n_{\mathsf{A}}, n_{\mathsf{A}} - f)$ Class A code. In the table, the parameters for the proposed codes are given in columns 1 to 5 and those of the codes in [20] in columns 6 to 9. The code rates R and R^{b} of the proposed codes and the BASIC PM-MBR codes[6] are given in columns 4 and 7, respectively. The smallest possible field size for our codes and the smallest possible ring size for the BASIC PM-MBR codes are given in columns 5 and 9, respectively. The normalized repair complexity[7] for BASIC PM-MBR codes, $C_{\mathsf{r}}^{\mathsf{b}}$, is given in column 10, while the normalized repair complexity of the proposed codes, C_{r}, is given in column 11. The normalized repair bandwidth for BASIC PM-MBR codes, λ^{b}, is given in column 12, while the normalized repair bandwidth of the proposed codes, λ, is given in the last column. It can be seen that the proposed codes achieve significantly better repair complexity. However, this comes at the cost of a lower fault tolerance for the codes in rows 2 and 3 (but our codes have significantly higher code rate) and higher repair bandwidth (since BASIC PM-MBR codes are MBR codes, their normalized repair bandwidth is equal to 1). For the $(9, 5, 3)$ code, the same fault tolerance of the $(8, 5, 3)$ code in [20] is achieved, despite the fact that the proposed code has a higher code rate. We remark that FR codes achieve a better (trivial) repair complexity compared to our proposed codes. However, this comes at a cost of $R < 0.5$ and they cannot be constructed for any n, k, and δ, where δ is the repair locality.

In Table I.3, we compare the normalized repair bandwidth of the proposed codes using Construction 1 and Construction 2 for Class B nodes. In the table,

[6] BASIC PM-MBR codes have code rate $R^{\mathsf{b}} = B/n\delta^{\mathsf{b}}$, where $B = \binom{k+1}{2} + k(\delta^{\mathsf{b}} - k)$ is the file size and $\delta^{\mathsf{b}} \in \{k, k+1, \ldots, n-1\}$.

[7] The normalized repair complexity of BASIC PM-MBR codes is $C_{\mathsf{r}}^{\mathsf{b}} = (3.5\delta^{\mathsf{b}} + 2.5)(m-1)/2$ [20]. Such codes have $\beta = \delta^{\mathsf{b}}$, $\lambda = 1$, and $f = n - k$. The value of m is conditioned on the code length n (see [20, Th. 14]).

Proposed	n_{A}	τ	R	GF(q)	**BASIC PM-MBR**	R^{b}	δ^{b}	$\mathcal{R}_m$	$C_{\mathsf{r}}^{\mathsf{b}}$	C_{r}	λ^{b}	λ
$(9,5,3)$	8	1	0.5556	GF(11)	$(8,5,3)$	0.4464	7	$\mathcal{R}_{11}$	135	44	1	2.4
$(11,7,3)$	10	2	0.6364	GF(11)	$(11,7,4)$	0.4454	10	$\mathcal{R}_{11}$	187.5	66.2857	1	3
$(14,9,3)$	12	2	0.6428	GF(13)	$(14,9,5)$	0.4450	13	$\mathcal{R}_{17}$	384	70.6667	1	3.5556

Table I.2: Comparison of normalized repair complexity and bandwidth of (n,k,f) BASIC PM-MBR codes

Code	n_{A}	τ	λ^{C1}	λ^{C2}	**Improvement**
$(7,4)$	6	1	2	1.875	6.25%
$(10,6)$	9	2	2.5	2.4167	3.33%
$(13,8)$	12	3	3	2.9375	2.08%
$(14,8)$	12	3	2.375	2.3125	2.63%
$(16,10)$	15	4	3.5	3.45	1.43%

Table I.3: Improvement in normalized repair bandwidth of the proposed (n,k) codes when the Class B nodes are heuristically constructed.

with λ^{C1} and λ^{C2}, we refer to the normalized repair bandwidth for Construction 1 and Construction 2, respectively. For the codes presented, it is seen that the heuristic construction yields an improvement in repair bandwidth in the range of $1\% - 7\%$ with respect to Construction 1.

7 Conclusion

In this paper, we constructed a new class of codes that achieve low repair bandwidth and low repair complexity for a single node failure. The codes are constructed from two smaller codes, Class A and B, where the former focuses on the fault tolerance of the code, and the latter focuses on reducing the repair bandwidth and complexity. It is numerically seen that our proposed codes achieve better repair complexity than Zigzag codes, MDS codes, Piggyback codes, generalized Piggyback codes, exact-repairable MDS codes, BASIC PM-MBR codes, and are in some cases better than Pyramid codes. They also achieve a better repair bandwidth compared to MDS codes, Piggyback codes, generalized Piggyback codes, exact-repairable MDS codes, and are in some cases better than LRCs and Pyramid codes. A side effect of such a construction is that the number of symbols per node that need to be encoded grows only linearly with the code dimension. This implies that our codes are suitable for memory constrained DSSs as compared to Zigzag and MDR codes, for which the number of symbols per node increases exponentially with the code dimension.

A Proof of Theorem 1

Consider an arbitrary set of $\tau' \leq \tau$ piggybacked nodes, denoted by $\mathcal{T} = \{j_1, j_2, \ldots, j_{\tau'}\}$, $j_i = j_i' - (n_{\mathsf{A}} - \tau) + 1$, $j_i' \in \{n_{\mathsf{A}} - \tau, \ldots, n_{\mathsf{A}} - 1\}$. Then, the

repair of the i-th row, $i = 0, \ldots, k-1$, using the piggybacked nodes in $\mathcal{T}$, would depend upon the knowledge of the data symbols (piggybacks) in the rows $(i+j_1)_k, (i+j_2)_k, \ldots, (i+j_{\tau'})_k$. This is because the knowledge of the piggybacks in these rows allows to obtain the original MDS parity symbols in the i-th row. In the following, we use this observation. We first proceed to prove that if $\tau'(\tau' + n_{\mathsf{A}} - k - \tau) < k$, then $\theta + \tau' \leq n_{\mathsf{A}} - k - \tau + \tau'$ data nodes can be corrected using θ non-modified parity nodes and the τ' piggybacked nodes in $\mathcal{T}$. Using this we will complete the proof of Theorem 1.

Lemma 1. *Consider an (n_{A}, k) Class A code with $k+2 \leq n_{\mathsf{A}} < 2k$. The code consists of τ piggybacked nodes and $n_{\mathsf{A}} - k - \tau$ non-modified MDS parity nodes. If $\tau'(\tau' + n_{\mathsf{A}} - k - \tau) < k$, then the code can correct $\theta + \zeta$ data node failures using $\theta \leq n_{\mathsf{A}} - k - \tau$ non-modified parity nodes and the τ' piggybacked parity nodes in the set $\mathcal{T} = \{1, \ldots, \tau'\}$ for $\zeta \leq \tau' \leq \tau$.*

Proof. We consider first the case when $\zeta = \tau'$. Then, assume that $\theta + \zeta$ data nodes fail and there exists a sequence $i, (i+1)_k, \ldots, (i-1+\tau')_k$ of τ' data nodes that are available. By construction, the parity symbol $p^{\mathsf{A,p}}_{(j-1)_k, n_{\mathsf{A}}-\tau-1+t}$, $t \in \mathcal{T}$, is (see (I.3)) given by

$$p^{\mathsf{A,p}}_{(j-1)_k, n_{\mathsf{A}}-\tau-1+t} = p^{\mathsf{A}}_{(j-1)_k, n_{\mathsf{A}}-\tau-1+t} + d_{(j-1+t)_k, (j-1)_k}, \tag{I.11}$$

where $j = 0, \ldots, k-1$. To recover all data symbols, set $i' = (i+\tau')_k$ and perform the following steps.

1. Obtain the $\zeta = \tau'$ MDS parity symbols $p^{\mathsf{A}}_{(i'-1)_k, n_{\mathsf{A}}-\tau-1+t}$, $t \in \mathcal{T}$ (see (I.11)) in the $(i'-1)_k$-th row. This is possible because the piggybacks in the $(i'-1)_k$-th node are available.

2. Using the $\theta + \zeta$ MDS parity symbols and $k - \theta - \zeta$ data symbols in the $(i'-1)_k$-th row, recover the missing $\theta + \zeta$ symbols in the $(i'-1)_k$-th row of the failed data nodes.

3. $i' \leftarrow (i'-1)_k$.

4. Repeat Items 1), 2), and 3) $\tau' - 1$ times. This ensures that the failed symbols in the τ' rows $i', (i'+1)_k, \ldots, (i'-1+\tau')_k$ are recovered. This implies that the piggyback symbols

$$\begin{aligned} &d_{i', (i'-1)_k}, \ldots, d_{(i'-1+\tau')_k, (i'-1)_k} \text{ in node } (i'-1)_k, \\ &d_{i', (i'-2)_k}, \ldots, d_{(i'-2+\tau')_k, (i'-2)_k} \text{ in node } (i'-2)_k, \\ &\qquad\qquad \vdots \\ &d_{i', (i'-\tau')_k} \qquad\qquad\qquad\quad \text{in node } (i'-\tau')_k, \end{aligned} \tag{I.12}$$

   are recovered. In other words, (I.12) says that in the $(i'-t)_k$-th node, $\tau' + 1 - t$ piggybacked symbols are recovered. More specifically, for $t = 1$, τ' piggybacked symbols are recovered. Thus,

5. repeat Items 1) and 2), and set

6. $i' \leftarrow (i'-1)_k$. We thus recover the i'-th row and obtain the piggyback symbols in $\mathcal{D}_{\mathcal{R}_{i'}} \backslash \mathcal{D}_{\mathcal{X}_{i'}}$. This increases the number of obtained piggyback symbols by 1 in the next τ' nodes $i', (i'-1)_k, \ldots, (i'+\tau'-1)_k$. In a similar fashion, we now have τ' piggybacked symbols in the i'-th node, and Items 5) and 6) are repeated until all k rows have been recovered. With this recursion, one recovers the $\theta + \tau' = \theta + \zeta$ failed data nodes.

For the case when $\zeta < \tau'$, the aforementioned decoding procedure is still able recover $\theta + \zeta$ data node failures. This is because, in order to repair failed symbols in the $(i'-1)_k$-th row, one needs just $\zeta < \tau'$ MDS parity symbols from $\mathcal{T}' \subseteq \mathcal{T}$. If the $(i'-1)_k$-th node is available, then ζ piggybacks allow to recover the MDS parity symbols (see Item 1)). On the contrary, if the $(i-1)_k$-th node has been erased, then ζ piggybacks are obtained from (I.12).

Note that the argument above assumes that τ' consecutive data nodes are available. Thus, in order to guarantee that any $\theta + \zeta$ data nodes can be corrected, we consider the worst case scenario for node failures, where we equally spread $\theta + \zeta$ data node failures across the k data nodes. Since

$$\frac{k}{\theta + \zeta} \geq \frac{k}{n_{\mathsf{A}} - k - \tau + \tau'} > \tau',$$

where the last inequality follows by the assumption on τ' stated in the lemma, it follows that the largest gap of non-failed data nodes in the worst case scenario is indeed greater than or equal to τ'. □

The above lemma shows that the Class A code can correct up to $n_{\mathsf{A}} - k - \tau + \tau'$ erasures using non-modified parity nodes and τ' modified parity nodes, provided the condition on τ' is satisfied.

To prove that the code can correct $n_{\mathsf{A}} - k - \tau + \tau'$ arbitrary node failures, let us assume that ρ Class A parity nodes and $n_{\mathsf{A}} - k - \tau + \tau' - \rho$ data nodes have failed. More precisely, let $\rho_1 \leq n_{\mathsf{A}} - k - \tau$ non-modified nodes, $\rho_2 \leq \tau'$ piggybacked nodes in $\mathcal{T}$, and $\rho_3 \geq 0$ remaining piggybacked nodes, where $\rho_1 + \rho_2 + \rho_3 = \rho$, fail. Clearly, it can be seen that there are $n_{\mathsf{A}} - k - \tau - \rho_1$ non-modified parity nodes and a set of $\zeta = \tau' - \rho_2$ modified nodes $\mathcal{T}' \subseteq \mathcal{T}$ available. Also, note that the number of data node failures is ρ_3 less than the number of combined available piggybacked nodes in $\mathcal{T}'$ and available non-modified parity nodes. Thus, using Lemma 1 with $\theta = n_{\mathsf{A}} - k - \tau - \rho_1 - \rho_3$ and $\zeta = \tau' - \rho_2$, it follows that $\theta + \zeta = (n_{\mathsf{A}} - k - \tau - \rho_1 - \rho_3) + (\tau' - \rho_2) = n_{\mathsf{A}} - k - \tau + \tau' - \rho$ data nodes can be repaired. The remaining ρ failed parity nodes can then be repaired using the k^2 data symbols in the k data nodes.

We remark that the decoding procedure in Lemma 1 in essence solves a system of linear equations by eliminating $\tau' k$ variables (piggybacks) in τ' parity nodes. Once the piggybacks are eliminated, the $k(\theta + \tau')$ data symbols are obtained by solving k systems of linear equations. Thus, the decoding procedure is optimal, i.e., it is maximum likelihood decoding.

Consider the quadratic function $\psi(\tau') = {\tau'}^2 + (n_{\mathsf{A}} - k - \tau)\tau' - k$. According to the proof of Lemma 1 when $\psi(\tau') \geq 0$, the decoding procedure fails as one can construct a failure pattern for the data nodes where the largest separation (in the

Algorithm 1: Class B node construction when k is even

Initialization:
$\boldsymbol{A} = [a_{i,j}]$ as defined in (I.7)
$n = n_{\mathsf{A}} + n_{\mathsf{B}} - k$ with $n_{\mathsf{B}} < 2k - \tau$ and $n_{\mathsf{A}} < 2k$
1 **for** $l \in \{n_{\mathsf{A}}, \ldots, n-1\}$ **do**
2 $\rho_l \leftarrow k - \tau - 1 - l + n_{\mathsf{A}}$
3 **if** $l \leq n_{\mathsf{A}} + k/2 - \tau - 2$ **then**
4 $\mathcal{P}_l^{\mathsf{B,h}} \leftarrow \mathcal{P}_l^{\mathsf{B}}$
5 **else if** $l > n_{\mathsf{A}} + k/2 - \tau - 2$ and $\rho_l > 1$ **then**
6 $\mathcal{P}_l^{\mathsf{B,h}} \leftarrow$ `ConstructNode(`$\boldsymbol{A}, \rho_l$`)`
7 **else if** $l > n_{\mathsf{A}} + k/2 - \tau - 2$ and $\rho_l = 1$ **then**
8 $\mathcal{P}_l^{\mathsf{B,h}} \leftarrow$ `ConstructLastNode()`
9 **end**
10 $\boldsymbol{A} \leftarrow$ `UpdateReadCost()`
11 **end**

number of available nodes) between the failed nodes would be strictly smaller than τ'. The largest τ' such that $\psi(\tau') < 0$ can be determined as follows. By simple arithmetic, one can prove that $\psi(\tau')$ is a convex function with a negative minima and with a positive and a negative root. Therefore,

$$0 \leq \tau' < \xi = \frac{\sqrt{(n_{\mathsf{A}} - k - \tau)^2 + 4k} - (n_{\mathsf{A}} - k - \tau)}{2},$$

where ξ is the positive root of $\psi(\tau')$. Furthermore, it may happen that $\tau < \xi$. Therefore, the maximum number of node failures that the code can tolerate is

$$f = \begin{cases} n_{\mathsf{A}} - k - \tau + \left\lfloor \frac{\sqrt{(n_{\mathsf{A}}-k-\tau)^2+4k}-(n_{\mathsf{A}}-k-\tau)}{2} \right\rfloor & \text{if } \tau \geq \xi \\ n_{\mathsf{A}} - k & \text{if } \tau < \xi \end{cases}.$$

B Class B Parity Node Construction

In this appendix, we give an algorithm that constructs $n_{\mathsf{B}} - k$ Class B parity nodes $\mathcal{P}_{n_{\mathsf{A}}}^{\mathsf{B,h}}, \ldots, \mathcal{P}_{n-1}^{\mathsf{B,h}}$. The algorithm is a heuristic for the construction of Class B parity nodes such that the repair bandwidth of failed nodes is further reduced in comparison with Construction 1 in Section 4.

The nodes $\mathcal{P}_l^{\mathsf{B,h}}$, $l = n_{\mathsf{A}}, \ldots, n-1$, are constructed recursively as shown in Algorithm 1. The k parity symbols in the l-th node are sums of at most ρ_l data symbols $d_{i,j} \in \cup_{j'} \mathcal{D}_{Q_{j'}}$. The construction of the l-th node for $l \leq n_{\mathsf{A}} + k/2 - \tau - 2$ (see Line 4) is identical to that resulting from Construction 1 in Section 4. The remaining parity nodes $\mathcal{P}_l^{\mathsf{B,h}}$, $l > n_{\mathsf{A}} + k/2 - \tau - 2$, are constructed using the sub-procedures `ConstructNode(` $\boldsymbol{A}, \rho_l$`)` and `ConstructLastNode()`. After the construction of each parity node, the read costs of the data symbols $d_{i,j}$ are updated by the sub-procedure `UpdateReadCost()`. In the following, we describe each of the above-mentioned sub-procedures.

B.1 ConstructNode()

This sub-procedure allows the construction of the l-th Class B parity node, where each parity symbol in the node is a sum of at most ρ_l data symbols. The algorithm for the sub-procedure is shown in Algorithm 2. Here, the algorithm is divided into two parts. The first part (Line 1 to Line 19) adds at most two data symbols to each of the k parity symbols, while the second part (Line 21 to Line 43) adds at most $\rho_l - 2$ data symbols.

In the first part, each parity symbol $p^{\mathsf{B}}_{t,l}$, $t = 0, \ldots, k-1$, is recursively constructed by adding a symbol $d_{i,j} \in \mathcal{D}_{\mathcal{Q}_j \setminus \cup_{j'} \mathcal{U}_{j'}}$ which has a corresponding read cost $a_{i,j}$ that is the largest among all symbols indexed by $\mathcal{Q}_j \setminus \cup_{j'} \mathcal{U}_{j'}$ (see Line 2 and Line 7). The next symbol added to $p^{\mathsf{B}}_{t,l}$ is $d_{j,i} \in \mathcal{D}_{\mathcal{X}_j}$ if such a symbol exists (Line 4). Otherwise, the symbol added is $d_{j,i_2} \in \mathcal{D}_{\mathcal{X}_j \setminus \cup_{j'} \mathcal{U}_{j'}}$ if such a symbol exists (Line 10). The set $\mathcal{U}_t$ denotes the index set of data symbols from $\mathcal{D}$ that are added to $p^{\mathsf{B}}_{t,l}$. Note that there exist multiple choices for the symbol $d_{i,j}$. A symbol $d_{i,j}$ such that there is a valid $d_{j,i} \in \mathcal{D}_{\mathcal{X}_j}$ is preferred, since it allows a larger reduction of the repair bandwidth.

The second part of the algorithm chooses recursively at most $\rho_l - 2$ data symbols that should participate in the construction of the k parity symbols. The algorithm chooses a symbol $d_{j,i_3} \in \mathcal{D}_{\mathcal{X}_j \cup_{j'} \mathcal{U}_{j'}} \neq \emptyset$ such that $\mathrm{read}(d_{j,i_3}, p^{\mathsf{B}}_{t,l} + d_{j,i_3}) < a_{j,i_3}$ (see Line 23). In other words, choose data symbols such that their read cost do not increase. It may happen that $\mathcal{D}_{\mathcal{X}_j \setminus \cup_{j'} \mathcal{U}_{j'}} = \emptyset$. If so, select $d_{j_2,i} \in \cup_{j_1} \mathcal{D}_{\mathcal{X}_{j_1} \setminus \cup_{j'_1} \mathcal{U}_{j'_1}}$ such that $d_{i',j_2} \in \mathcal{D}_{\mathcal{U}_t}$ and $a_{j_2,i} > 1$, for some $i' \neq i$, exists, and then add $d_{j_2,i}$ to the parity symbol $p^{\mathsf{B}}_{t,l}$ (see Line 33). If $d_{j_2,i}$ does not exist, then an arbitrary symbol $d_{j_3,i} \in \cup_{j_1} \mathcal{D}_{\mathcal{X}_{j_1} \setminus \cup_{j'_1} \mathcal{U}_{j'_1}}$ is added (see Line 37). This process is then repeated $\rho_l - 2$ times.

B.2 ConstructLastNode()

This procedure constructs the l-th Class B parity node that has $\rho_l = 1$. In other words, each parity symbol $p^{\mathsf{B}}_{t,l}$ is a data symbol $d_{i,j} \in \cup_{j'} \mathcal{D}_{\mathcal{Q}_{j'}}$. The procedure works as follows. First, initialize $\mathcal{U}_t$ to be the empty set for $t = 0, \ldots, k-1$. Then, for $t = 0, \ldots, k-1$, assign first the data symbol $d_{i,j}$ to the parity symbol $p^{\mathsf{B}}_{t,l}$, where $d_{i,j} \in \mathcal{D}_{\Psi(\boldsymbol{A}_{\cup_{j'} \mathcal{Q}_{j'} \setminus \cup_{t'} \mathcal{U}_{t'}})}$, and then subsequently add (i,j) to $\mathcal{U}_t$. Note that for each iteration there may exist several choices for $d_{i,j}$, in which case we can pick one of these randomly.

B.3 UpdateReadCost()

After the construction of the l-th node, we update the read costs of all data symbols $d_{i,j} \in \cup_{j'} \mathcal{D}_{\mathcal{Q}_{j'}}$. These updated values are used during the construction of the $(l+1)$-th node. The read cost updates for the parity symbol $p^{\mathsf{B}}_{t,l}$, $t = 0, \ldots, k-1$, are

$$a_{i,j} = \mathrm{read}(d_{i,j}, p^{\mathsf{B}}_{t,l}), \ \forall d_{i,j} \in \mathcal{D}_{\mathcal{U}_t}.$$

Algorithm 2: `ConstructNode` $(\boldsymbol{A}, \rho_l)$

Initialization:
$t, j \leftarrow 0$
$\mathcal{U}_{t_1} \leftarrow \emptyset, \quad t_1 = 0, \ldots, k-1$
while $\min_{\forall t_1} |\mathcal{U}_{t_1}| < 2$ **do**
  Select $d_{i,j} \in \mathcal{D}_{\Psi(\boldsymbol{A}_{\mathcal{Q}_j \setminus \cup_{j'} \mathcal{U}_{j'}})}$ s.t. $\exists\, d_{j,i} \in \mathcal{D}_{\mathcal{X}_j}, a_{j,i} > 1$
  if $d_{i,j}$ *exists* **then**
    $p^{\mathsf{B}}_{t,l} \leftarrow d_{i,j} + d_{j,i}$
    $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{(i,j),(j,i)\}; t \leftarrow t+1$
  else
    Select $d_{i_1,j} \in \mathcal{D}_{\Psi(\boldsymbol{A}_{\mathcal{Q}_j \setminus \cup_{j'} \mathcal{U}_{j'}})}$
    if $d_{i_1,j}$ *exists* **then**
      if $\exists\, d_{j,i_2} \in \mathcal{D}_{\mathcal{X}_j \setminus \cup_{j'} \mathcal{U}_{j'}}$ s.t. $\mathrm{read}(d_{j,i_2}, p^{\mathsf{B}}_{t,l} + d_{j,i_2}) < a_{j,i_2}$ and $a_{j,i_2} > 1$ **then**
        $p^{\mathsf{B}}_{t,l} \leftarrow d_{i_1,j} + d_{j,i_2}$
        $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{(i_1,j),(j,i_2)\}; t \leftarrow t+1$
      else
        $p^{\mathsf{B}}_{t,l} \leftarrow d_{i_1,j}$
        $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{(i_1,j),(-1,-1)\}; t \leftarrow t+1$
      end
    end
  end
  $j \leftarrow (j+1)_k$
end
$t, j \leftarrow 0$
while $\min_{\forall t_1} |\mathcal{U}_{t_1}| \le \rho_l$ **do**
  if $\mathcal{D}_{\mathcal{X}_j \setminus \cup_{j_1} \mathcal{U}_{j_1}} \neq \emptyset$ **then**
    if $\exists\, d_{j,i_3} \in \mathcal{D}_{\mathcal{X}_j \setminus \cup_{j'} \mathcal{U}_{j'}}$ s.t. $\mathrm{read}(d_{j,i_3}, p^{\mathsf{B}}_{t,l} + d_{j,i_3}) < a_{j,i_3}$ and $a_{j,i_3} > 1$ **then**
      $p^{\mathsf{B}}_{t,l} \leftarrow p^{\mathsf{B}}_{t,l} + d_{j,i_3}$
      $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{(j,i_3)\}; t \leftarrow t+1$
    else if $l > n_{\mathsf{A}}$ **then**
      $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup (-1,-1)\}; t \leftarrow t+1$
    end
  else
    if $|\mathcal{U}_t| \le \rho_l - 1$ **then**
      Select $d_{j_2,i} \in \cup_{j_1} \mathcal{D}_{\mathcal{X}_{j_1} \setminus \cup_{j_1'} \mathcal{U}_{j_1'}}$ s.t. $d_{i',j_2} \in \mathcal{D}_{\mathcal{U}_t}$ and $a_{j_2,i} > 1$ for some $i' \neq i$
      if $d_{j_2,i}$ *exists* **then**
        $p^{\mathsf{B}}_{t,l} \leftarrow p^{\mathsf{B}}_{t,l} + d_{j_2,i}$
        $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{(j_2,i)\}; t \leftarrow t+1$
      else
        Select $d_{j_3,i} \in \cup_{j_1} \mathcal{D}_{\mathcal{X}_{j_1} \setminus \cup_{j_1'} \mathcal{U}_{j_1'}}$
        $p^{\mathsf{B}}_{t,l} \leftarrow p^{\mathsf{B}}_{t,l} + d_{j_3,i}$
        $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{(j_3,i)\}; t \leftarrow t+1$
      end
    end
  end
  $j \leftarrow (j+1)_k$
end
return $\{p^{\mathsf{B}}_{0,l}, \ldots, p^{\mathsf{B}}_{k-1,l}\}$

References

[1] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, "Scale-RS: An efficient scaling scheme for RS-coded storage clusters," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1704–1717, Jun. 2015.

[2] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. IEEE Int. Symp. Network Comput. and Appl. (NCA)*, Cambridge, MA, Jul. 2007.

[3] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in Windows Azure storage," in *Proc. USENIX Annual Technical Conf.*, Boston, MA, Jun. 2012.

[4] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," in *Proc. 39th Very Large Data Bases Endowment (VLDB)*, Trento, Italy, Aug. 2013.

[5] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[6] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[7] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[8] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annual Allerton Conf. Commun., Control, and Comput.*, Monticello, IL, Sep./Oct. 2010.

[9] Y. Wang, X. Yin, and X. Wang, "MDR codes: A new class of RAID-6 codes with optimal rebuilding and encoding," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 1008–1018, May 2014.

[10] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[11] V. R. Cadambe, C. Huang, J. Li, and S. Mehrotra, "Polynomial length MDS codes with optimal repair in distributed storage," in *Proc. 45th Asilomar Conf. Signals, Syst. and Comput. (ASILOMAR)*, Pacific Grove, CA, Nov. 2011.

[12] G. K. Agarwal, B. Sasidharan, and P. V. Kumar, "An alternate construction of an access-optimal regenerating code with optimal sub-packetization level," in *Proc. 21st Nat. Conf. Commun. (NCC)*, Mumbai, India, Feb. 2015.

[13] J. Li, X. Tang, and U. Parampalli, "A framework of constructions of minimal storage regenerating codes with the optimal access/update property," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1920–1932, Apr. 2015.

[14] Z. Wang, I. Tamo, and J. Bruck, "Explicit minimum storage regenerating codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 8, pp. 4466–4480, Aug. 2016.

[15] B. Sasidharan, M. Vajha, and P. V. Kumar, "An explicit, coupled-layer construction of a high-rate MSR code with low sub-packetization level, small field size and all-node repair," Sep. 2016, arXiv: 1607.07335v3. [Online]. Available: https://arxiv.org/abs/1607.07335

[16] M. Ye and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6307–6317, Oct. 2017.

[17] J. Li, X. Tang, and C. Tian, "A generic transformation for optimal repair bandwidth and rebuilding access in MDS codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017.

[18] N. Raviv, N. Silberstein, and T. Etzion, "Constructions of high-rate minimum storage regenerating codes over small fields," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2015–2038, Apr. 2017.

[19] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5802–5820, Sep. 2017.

[20] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC codes: Low-complexity regenerating codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3053–3069, Jun. 2016.

[21] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-fault-tolerant binary MDS array codes with asymptotically optimal repair," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017.

[22] S. Kumar, A. Graell i Amat, I. Andriyanova, and F. Brännström, "A family of erasure correcting codes with low repair bandwidth and low repair complexity," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, Dec. 2015.

[23] S. Yuan and Q. Huang, "Generalized piggybacking codes for distributed storage systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, Dec. 2016.

[24] A. S. Rawat, I. Tamo, V. Guruswami, and K. Efremenko, "MDS code constructions with small sub-packetization and near-optimal repair bandwidth," *IEEE Trans. Inf. Theory*, to appear.

[25] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[26] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Sov. Phys. Doklady*, vol. 7, no. 7, pp. 595–596, Jan. 1963.

[27] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba algorithm for efficient implementations," Jul. 2006. [Online]. Available: https://eprint.iacr.org/2006/224.pdf

[28] J. M. Pollard, "The fast Fourier transform in a finite field," *Math. Comput.*, vol. 25, no. 114, pp. 365–374, Apr. 1971.

[29] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*. Springer, 2001.

[30] S. Gao and T. Mateer, "Additive fast Fourier transforms over finite fields," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6265–6272, Dec. 2010.

Paper II

Secure Repairable Fountain Codes

Siddhartha Kumar, Eirik Rosnes, and Alexandre Graell i Amat

IEEE Communications Letters, vol. 20, no. 8, August 2016.

The layout has been revised.

Abstract

In this letter, we provide the construction of repairable fountain codes (RFCs) for distributed storage systems that are information-theoretically secure against an eavesdropper that has access to the data stored in a subset of the storage nodes and the data downloaded to repair an additional subset of storage nodes. The security is achieved by adding random symbols to the message, which is then encoded by the concatenation of a Gabidulin code and an RFC. We compare the achievable code rates of the proposed codes with those of secure minimum storage regenerating codes and secure locally repairable codes.

1 Introduction

The design of information-theoretically secure distributed storage systems (DSSs) has attracted a significant interest in the last few years [1, 2]. DSSs use erasure correcting codes (ECCs) to yield fault tolerance against storage node failures. The resiliency of the DSS against passive attacks is a good measure of its security. Passive attacks are those where the attacker (referred to as the eavesdropper) gains access to a subset of storage nodes and thereby to partial information on the data stored on the DSS. Information-theoretic security against such attacks involves mixing of information symbols (called the *message*) with random symbols, prior to encoding by an ECC, in a manner such that the eavesdropper does not gain any information about the original message even if he has access to some code symbols.

Using this idea, [1, 2] provided explicit constructions of minimum storage regenerating (MSR) codes that achieve security for an (ℓ_1, ℓ_2) eavesdropper model where the eavesdropper has access to the content of ℓ_1 storage nodes and the data that needs to be downloaded to repair ℓ_2 additional storage nodes. The design of secure locally repairable codes (LRCs) was also addressed in [2]. In particular, to achieve security, random symbols are appended to the message and the resulting vector of symbols is precoded by a Gabidulin code [3] prior to encoding by an LRC (or MSR code) in [2]. Achieving security comes at the expense of a lower code rate with respect to the original LRC (or MSR code), due to appending random symbols to the message [2]. For the LRC- and MSR-based secure codes, the authors in [2] derived the maximum message size (equivalently, the maximum code rate) that allows to achieve security. Moreover, the code constructions in [2] achieve this maximum. A sufficient condition for the information leakage to the eavesdropper to be zero was also given in [1, 2].

LRCs [4] and MSR codes [5] are appealing code families because they are repair efficient. Repairable fountain codes (RFCs) are another class of repair-efficient ECCs [6]. Like LRCs, they yield a good locality, which implies that few storage nodes are involved in the repair of a failed node.

In this letter, we present the construction of RFCs that are information-theoretically secure for the (ℓ_1, ℓ_2) eavesdropper model. As in [2], we achieve

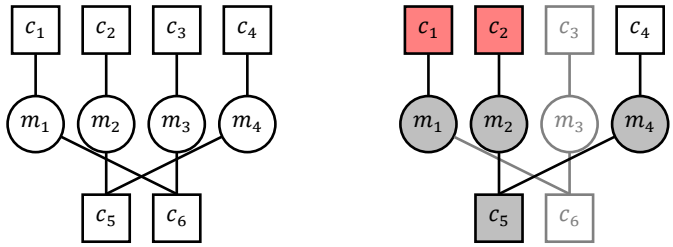

Figure II.1: A DSS with 6 storage nodes employing a $(6, 4)$ ECC. $\boldsymbol{m} = (m_1, m_2, m_3, m_4)$ is encoded into the codeword $\boldsymbol{c} = (c_1, c_2, c_3, c_4, c_5, c_6)$. Each code symbol is stored in a storage node. (a) Bipartite graph of the $(6, 4)$ ECC; squares and circles represent code symbols (storage nodes) and message symbols, respectively. (b) Example of a $(1, 1)$ eavesdropper, where $\mathcal{S}_1 = \{c_1\}$ and $\mathcal{S}_2 = \{c_2\}$ (in red). Gray symbols are the symbols that the eavesdropper obtains.

security by appending random symbols to the message and precoding by a Gabidulin code. We prove that the proposed code construction is completely secure for the (ℓ_1, ℓ_2) eavesdropper model. To prove security, we give a necessary condition for the information leakage to the eavesdropper to be zero, thus extending the sufficient condition in [1, 2]. Our proof differs from the one in [1, 2] and is based on simple information theory equalities. We compare the achievable code rates (the maximum code rate that allows to achieve security) of the proposed codes with those of secure MSR codes and LRCs in [2]. We show that, for a given rate of the underlying code (RFC, LRC, or MSR code), secure RFCs yield the same achievable code rates as those of secure LRCs and better than those of secure MSR codes when the rate of the underlying code is high enough.

2 System Model

We consider a DSS with n storage nodes, each storing one symbol. A message $\boldsymbol{m} = (m_1, m_2, \ldots, m_k)$, of length k symbols $m_i \in \mathrm{GF}(q^p)$, $i = 1, \ldots, k$, where q is a prime and p is a positive integer, is first encoded using an (n, k) ECC of rate $R = k/n$ into a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ of length n. Each of the n code symbols c_i, $i = 1, \ldots, n$, is then stored into a different storage node. We assume that code symbol c_i is stored in the ith storage node and, with a slight abuse of notation, we will refer to both code symbol and storage node i by c_i.

Example 1. *The bipartite graph shown in Fig. II.1(a) represents a message stored on a DSS with $n = 6$ storage nodes using a $(6, 4)$ ECC. Each code symbol c_i, $i = 1, \ldots, 6$, is a linear combination of its neighboring message symbols m_i, $i = 1, \ldots, 4$ (circles). Each code symbol (squares) is stored on a different storage node.*

2.1 Security Model

We consider an (ℓ_1, ℓ_2) eavesdropper model [2], where the eavesdropper can passively observe, but not modify, the content of $\ell = \ell_1 + \ell_2 < k$ storage nodes. Out of the ℓ nodes, the eavesdropper can observe the symbols stored in a subset of ℓ_1 storage nodes, which we denote by $\mathcal{S}_1$ ($|\mathcal{S}_1| = \ell_1$). Furthermore, it can observe the data downloaded during the repair of a subset of ℓ_2 storage nodes, denoted by $\mathcal{S}_2$ ($|\mathcal{S}_2| = \ell_2$), where $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. This model is relevant in the scenario where nodes are located at different geographical locations. Peer-to-peer storage systems are examples of such DSSs [1]. We denote the subset of storage nodes from which data is downloaded to repair storage nodes in $\mathcal{S}_2$ by $\mathcal{S}_\mathrm{d}$. We will refer to the symbols the eavesdropper obtains as the *eavesdropped symbols*. We also assume that the eavesdropper has perfect knowledge of the ECC used for encoding.

Definition 1 ([1, 2])**.** *Let $\boldsymbol{e}$ be the vector of eavesdropped symbols that the eavesdropper obtains from the storage nodes in $\mathcal{S}_1 \cup \mathcal{S}_\mathrm{d}$. A DSS storing a message $\boldsymbol{m}$ (possibly encoded by an ECC) is said to be* completely secure *against an (ℓ_1, ℓ_2) eavesdropper if the mutual information between the message and the eavesdropped symbols is zero, i.e., $I(\boldsymbol{m}; \boldsymbol{e}) = 0$.*

Example 2. *Fig. II.1(b) shows an example of a $(1, 1)$ eavesdropper where $\mathcal{S}_1 = \{c_1\}$ and $\mathcal{S}_2 = \{c_2\}$. Thus, the eavesdropper obtains $c_1 = m_1$ and the downloaded data $c_5 = m_2 + m_4$ and $c_4 = m_4$, and thereby m_2, during the repair of $\mathcal{S}_2 = \{c_2\}$. In all, the eavesdropper obtains the symbols m_1, m_2, m_4, and $c_5 = m_2 + m_4$, colored in gray in the figure.*

3 Gabidulin and Repairable Fountain Codes

We summarize Gabidulin codes and RFCs, which are the building blocks of the secure RFCs presented in Section 4.

3.1 Gabidulin Codes

Gabidulin codes are a class of rank codes [3]. An (N, K) Gabidulin code (over $\mathrm{GF}(q^p)$) of length N, dimension K, and minimum rank distance D_min, can correct up to $D_\mathrm{min} - 1$ rank erasures. Gabidulin codes are maximum rank distance codes, i.e., they achieve the Singleton bound, $D_\mathrm{min} \leq N - K + 1$, and are obtained by evaluations of polynomials. More specifically, Gabidulin codes use linearized polynomials.

Definition 2. *A linearized polynomial $f(y)$ of degree $t > 0$ over $GF(q^p)$ has the form*

$$f(y) = \sum_{i=0}^{t} a_i y^{q^i},$$

where $a_i \in GF(q^p)$ and $a_t \neq 0$.

A message $\boldsymbol{m} = (m_1, \ldots, m_K)$ is encoded using an (N, K) Gabidulin code as follows.

1. Construct a polynomial $f(y) = \sum_{i=1}^{K} m_i y^{q^{i-1}}$.
2. Evaluate $f(y)$ at N linearly independent (over GF(q)) points $\{y_1, \ldots, y_N\} \subset$ GF(q^p) to obtain a codeword $(f(y_1), \ldots, f(y_N))$.

Decoding proceeds as follows.

1. Obtain any K evaluations at K linearly independent (over GF(q)) points. Otherwise, decoding fails.
2. Perform polynomial interpolation on the K evaluations and recover the original message $\boldsymbol{m}$ by solving a system of linear equations.

3.2 Repairable Fountain Codes

An (n, k) systematic RFC encodes a message $\boldsymbol{m} = (m_1, \ldots, m_k) \in \text{GF}(q^p)^k$, $q > k$, into a codeword $\boldsymbol{c} = (c_1, \ldots, c_n)$, where $c_i = m_i$ for $i = 1, \ldots, k$. The parity symbols c_i, $i = k+1, \ldots, n$, are constructed according to the following three-step procedure.

1. Successively select $\xi = O(\log k)$ message symbols independently and uniformly at random with replacement.
2. For each of the ξ message symbols, a coefficient is drawn uniformly at random from GF$(q) \subset$ GF(q^p).
3. The parity symbol is then obtained as the linear combination of the ξ chosen message symbols, weighted by the corresponding coefficients.

Each of the n code symbols is stored in a different storage node. From the code construction, each parity symbol is a weighted sum of at most ξ message symbols. A parity symbol and the corresponding (at most) ξ message symbols is referred to as a local group. The existence of local groups is a hallmark of any ECC having low locality. Unlike LRCs, which have only disjoint local groups, RFCs also have overlapping local groups [6]. Furthermore, for each systematic symbol there exist a number of disjoint local groups from which it can be reconstructed. This allows multiple parallel reads of the systematic symbol, accessing the disjoint local groups. When a storage node fails, it is repaired from one of its local groups. This requires the download of at most ξ symbols (from the other at most ξ nodes of the local group). Thus, RFCs have low locality, ξ, and their repair bandwidth is $\xi p \log q$. Also, RFCs are near maximum distance separable codes.

4 Secure Repairable Fountain Codes

In this section, we present the construction of RFCs that are secure against the (ℓ_1, ℓ_2) eavesdropper model. The proposed secure RFCs are obtained by concatenating a Gabidulin code and an RFC. More precisely, consider an $(n, \tilde{k})$ RFC such

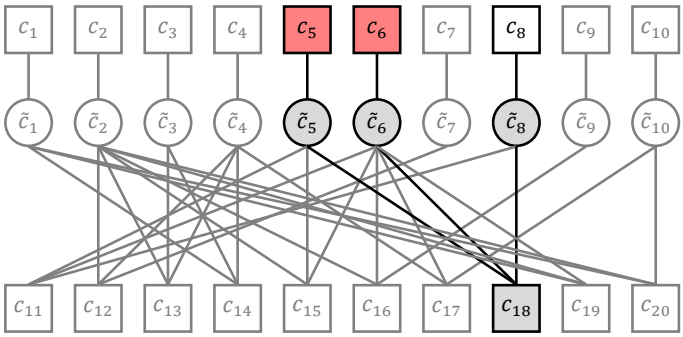

Figure II.2: A $(20, 10)$ secure RFC. Storage nodes $\mathcal{S}_1 = \{c_6\}$ and $\mathcal{S}_2 = \{c_5\}$ are eavesdropped. Gray symbols are the symbols that the eavesdropper obtains.

that each parity symbol is a random linear combination of up to ξ randomly chosen input symbols. Let $\boldsymbol{m}$ denote the message of length $k = \tilde{k} - \ell_1 - \xi\ell_2$ symbols. A codeword of the proposed secure RFC is constructed as follows.

1. Append to $\boldsymbol{m}$ a random vector $\boldsymbol{r} = (r_1, \dots, r_u)$ of length $u = \ell_1 + \xi\ell_2$ symbols, drawn independently and uniformly at random from $\mathrm{GF}(q^p)$, thus obtaining the vector $\tilde{\boldsymbol{m}} = (\boldsymbol{m}, \boldsymbol{r})$.
2. *Outer code.* Encode $\tilde{\boldsymbol{m}}$ using a $(\tilde{k}, \tilde{k})$ Gabidulin code to obtain the intermediate codeword $\tilde{\boldsymbol{c}} = (\tilde{c}_1, \dots, \tilde{c}_{\tilde{k}}) = (f(y_1), \dots, f(y_{\tilde{k}}))$.
3. *Inner code.* Encode $\tilde{\boldsymbol{c}}$ using an $(n, \tilde{k})$ RFC into the codeword $\boldsymbol{c} = (c_1, \dots, c_n)$. The n code symbols are then stored in n storage nodes.

Remark 1. *A $GF(q)$-linear combination of evaluations of a linearized polynomial $f(y) = \sum_{i=0}^{t} a_i y^{q^i}$ of some degree t over $GF(q^p)$ (see Definition 2) is itself an evaluation of the same linearized polynomial. In particular, $\sum_{j=1}^{\kappa} \gamma_j f(\beta_j) = f\left(\sum_{j=1}^{\kappa} \gamma_j \beta_j\right)$, where κ is a positive integer, $\gamma_j \in GF(q)$, and $\beta_j \in GF(q^p)$, i.e., $f(\cdot)$ is a linear map over $GF(q)$ [2, Remark 8]. Thus, each code symbol c_i, $i = 1, \dots, n$, is an evaluation of a linearized polynomial $f(\cdot)$ of degree at most $\tilde{k} - 1$ and with coefficients from $\tilde{\boldsymbol{m}}$ at some point $y_i \in GF(q^p)$, i.e., $c_i = f(y_i) = \sum_{j=1}^{\tilde{k}} \tilde{m}_j y_i^{q^{j-1}}$.*

Example 3. *Fig. II.2 depicts a toy example of a $(20, 10)$ secure RFC for a $(1, 1)$ eavesdropper. Here, $\tilde{\boldsymbol{m}}$ comprises $k = \tilde{k} - \ell_1 - \xi\ell_2 = 6$ message symbols and $u = \ell_1 + \xi\ell_2 = 4$ random symbols. $\tilde{\boldsymbol{m}}$ is encoded using the concatenation of a $(10, 10)$ Gabidulin code and a $(20, 10)$ RFC. Due to the outer encoding by the Gabidulin code, each code symbol c_i, $i = 1, \dots, 20$, is an evaluation of a linearized polynomial (see Remark 1). Another consequence is that the final code retains the repair properties of the inner code (the RFC). For this example, the code locality is $\xi = 3$.*

In the following, we show that the proposed secure RFCs achieve complete security for the (ℓ_1, ℓ_2) eavesdropper model. We first prove a sufficient and necessary condition for $I(\boldsymbol{m}; \boldsymbol{e}) = 0$ using an alternative proof to the one in [1, 2].

Theorem 1. *Let $\boldsymbol{m}$ be a message which is stored in a DSS by first appending to it a vector $\boldsymbol{r}$ of random symbols and then encoding $(\boldsymbol{m}, \boldsymbol{r})$ by an ECC. Also, let $\boldsymbol{e}$ be the vector of code symbols the eavesdropper has access to. Then, the information leakage to the eavesdropper is zero, i.e., $I(\boldsymbol{m}; \boldsymbol{e}) = 0$, if and only if $H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}) = H(\boldsymbol{r}) - H(\boldsymbol{e})$.*

Proof. We prove the theorem using simple information theory equalities,

$$\begin{aligned} I(\boldsymbol{m}; \boldsymbol{e}) &= H(\boldsymbol{m}) - H(\boldsymbol{m}|\boldsymbol{e}) \\ &\overset{(a)}{=} H(\boldsymbol{m}) - H(\boldsymbol{m}|\boldsymbol{e}) + H(\boldsymbol{e}|\boldsymbol{m}, \boldsymbol{r}) \\ &= H(\boldsymbol{m}) - H(\boldsymbol{m}|\boldsymbol{e}) + H(\boldsymbol{e}|\boldsymbol{m}) - I(\boldsymbol{r}; \boldsymbol{e}|\boldsymbol{m}) \\ &\overset{(b)}{=} H(\boldsymbol{e}) - I(\boldsymbol{r}; \boldsymbol{e}|\boldsymbol{m}) \\ &= H(\boldsymbol{e}) - H(\boldsymbol{r}|\boldsymbol{m}) + H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}) \\ &\overset{(c)}{=} H(\boldsymbol{e}) - H(\boldsymbol{r}) + H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}), \end{aligned}$$

where (a) follows from the fact that $H(\boldsymbol{e}|\boldsymbol{m}, \boldsymbol{r}) = 0$, since eavesdropped symbols are a function of $\boldsymbol{m}$ and $\boldsymbol{r}$, (b) follows from $H(\boldsymbol{e}) - H(\boldsymbol{e}|\boldsymbol{m}) = H(\boldsymbol{m}) - H(\boldsymbol{m}|\boldsymbol{e})$, and (c) follows from the fact that $\boldsymbol{r}$ and $\boldsymbol{m}$ are stochastically independent of each other, i.e., $H(\boldsymbol{r}|\boldsymbol{m}) = H(\boldsymbol{r})$. Thus,

$$I(\boldsymbol{m}; \boldsymbol{e}) = 0 \Leftrightarrow H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}) = H(\boldsymbol{r}) - H(\boldsymbol{e}). \tag{II.1}$$

□

We remark that in [1] and [2, Lem. 4] a sufficient condition on $I(\boldsymbol{m}; \boldsymbol{e}) = 0$ was proved, whereas Theorem 1 gives a sufficient and necessary condition. ECCs for which Theorem 1 is satisfied do not leak any information, i.e., they are completely secure.

In Theorem 2 below, we use the following lemma to prove that our proposed code construction is completely secure for the (ℓ_1, ℓ_2) eavesdropper model.

Lemma 1. *Consider the (ℓ_1, ℓ_2) eavesdropper model. For the proposed code construction (with $u = \ell_1 + \xi\ell_2$ random symbols $\boldsymbol{r}$), $H(\boldsymbol{e}) \leq H(\boldsymbol{r}) = (\ell_1 + \xi\ell_2)p \log q$, where $\boldsymbol{e}$ is the vector of code symbols the eavesdropper has access to.*

Proof. Consider the repair of a single storage node c_i in $\mathcal{S}_2$, and let $\Gamma^{(i)}$ denote *the* local group (there are many) used for the repair of storage node c_i. Each local group contains one inner code parity symbol and at most ξ inner code message symbols to which it is connected. Thus, $|\Gamma^{(i)}| \leq \xi + 1$. Since the inner code parity symbol is a GF(q)-weighted linear combination of the (at most) ξ inner code message symbols from the local group, $\Gamma^{(i)}$ contains at most ξ stochastically independent symbols. Considering the repair of all storage nodes in $\mathcal{S}_2$, it follows by the argument above that at most $\xi\ell_2$ stochastically independent inner code symbols are eavesdropped during the repair process. Also, since each storage node stores a single symbol, the eavesdropper has access to an additional ℓ_1 inner code symbols from the storage nodes in $\mathcal{S}_1$. Hence, in total, the eavesdropper has access to at most $\ell_1 + \xi\ell_2$ stochastically independent symbols from $\boldsymbol{c}$. Thus,

$H(\boldsymbol{e}) \leq (\ell_1 + \xi\ell_2)p \log q$. Furthermore, since $\boldsymbol{r}$ contains $u = \ell_1 + \xi\ell_2$ uniform independent random symbols, $H(\boldsymbol{r}) = (\ell_1 + \xi\ell_2)p \log q$, and the result follows. □

Theorem 2. *The code comprising of a Gabidulin code as its outer code and an RFC as its inner code, which encodes a vector $\tilde{\boldsymbol{m}} = (\boldsymbol{m}, \boldsymbol{r})$ that consists of a message $\boldsymbol{m}$ of length k and a random vector $\boldsymbol{r}$ of length $u = \ell_1 + \xi\ell_2$ is completely secure for the (ℓ_1, ℓ_2) eavesdropper model.*

Proof. To prove security, we show that $H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}) = H(\boldsymbol{r}) - H(\boldsymbol{e})$, which is *equivalent* to $I(\boldsymbol{m}; \boldsymbol{e}) = 0$ according to Theorem 1. Each eavesdropped symbol e_i, $i = 1, \ldots, w$, where $w = |\boldsymbol{e}|$, corresponds to a code symbol and therefore is an evaluation of $f(\cdot)$ at some point $z_i \in \{y_1, \ldots, y_n\} \subset \mathrm{GF}(q^p)$, where $c_i = f(y_i)$ (see Remark 1). Thus, for $i = 1, \ldots, w$,

$$e_i = f(z_i) = \sum_{j=1}^{\tilde{k}} \tilde{m}_j z_i^{q^{j-1}} = \sum_{j=1}^{k} m_j z_i^{q^{j-1}} + \sum_{j=1}^{u} r_j z_i^{q^{k+j-1}} \tag{II.2}$$

since $\tilde{\boldsymbol{m}} = (\boldsymbol{m}, \boldsymbol{r})$. In the following, $\boldsymbol{r} = (r_1, \ldots, r_u)$ is assumed to be the unknowns (the message $\boldsymbol{m}$ and the eavesdropper vector $\boldsymbol{e}$ are assumed to be known) in the linear system of equations defined in (II.2).

Let $1 \leq \nu \leq w$ (by definition) be the number of $\mathrm{GF}(q)$-linear independent symbols of $\{e_1, \ldots, e_w\}$, denoted by $\tilde{\boldsymbol{e}} = (\tilde{e}_1, \ldots, \tilde{e}_\nu)$. The corresponding vector of points from $\{z_1, \ldots, z_w\}$ is denoted by $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_\nu)$. From (II.2), $\tilde{\boldsymbol{e}} = \boldsymbol{b}(\tilde{\boldsymbol{z}}, \boldsymbol{m}) + \boldsymbol{r} \cdot \boldsymbol{A}(\tilde{\boldsymbol{z}})$, where $\boldsymbol{b}(\tilde{\boldsymbol{z}}, \boldsymbol{m})$ is a length-ν row vector and $\boldsymbol{A}(\tilde{\boldsymbol{z}})$ is a $u \times \nu$ matrix. Since $\{\tilde{e}_1, \ldots, \tilde{e}_\nu\}$ are $\mathrm{GF}(q)$-linear independent, the matrix $\boldsymbol{A}(\tilde{\boldsymbol{z}})$ is of full column-rank (i.e., its column space is a vector space over $\mathrm{GF}(q)$ of dimension ν), and since the u random symbols in $\boldsymbol{r}$ are chosen independently and uniformly at random from $\mathrm{GF}(q^p)$, $\{\tilde{e}_1, \ldots, \tilde{e}_\nu\}$ are also *stochastically* independent uniformly distributed random variables over $\mathrm{GF}(q^p)$ ($\tilde{e}_i$ is uniformly distributed over $\mathrm{GF}(q^p)$ for all i and $\tilde{\boldsymbol{e}}$ is uniformly distributed over $\mathrm{GF}(q^p)^\nu$). Finally, since $e \in \{e_1, \ldots, e_w\} \setminus \{\tilde{e}_1, \ldots, \tilde{e}_\nu\}$ can be written as a $\mathrm{GF}(q)$-linear combination of $\{\tilde{e}_1, \ldots, \tilde{e}_\nu\}$, it follows that $H(\boldsymbol{e}) = H(\tilde{\boldsymbol{e}}) = \nu \cdot p \log q$. From Lemma 1, $H(\boldsymbol{e}) \leq H(\boldsymbol{r})$. Thus, $u \geq \nu$ since $H(\boldsymbol{e}) = \nu \cdot p \log q$ and $H(\boldsymbol{r}) = u \cdot p \log q$. The conditional entropy $H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m})$ is equal to the logarithm (base-2) of the number of solutions of (II.2) when the number of unknowns u is larger than or equal to the number of independent equations ν, i.e., when $u \geq \nu$. Hence, $H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}) = (u - \nu)p \log q = H(\boldsymbol{r}) - H(\boldsymbol{e})$ from which the result follows from (II.1) (see Theorem 1). □

Example 4. *Consider the $(20, 10)$ secure RFC over $GF(q^p)$ in Fig. II.2 that encodes the message $\boldsymbol{m} = (m_1, \ldots, m_6)$ of 6 symbols and a vector $\boldsymbol{r} = (r_1, \ldots, r_4)$ of 4 random symbols. Each $c_i, i = 1, \ldots, 20$, is an evaluation of a linearized polynomial $f(\cdot)$ at y_i. For the $(1, 1)$ eavesdropper model, the scenario where $\mathcal{S}_1 = \{c_6\}$ and $\mathcal{S}_2 = \{c_5\}$, i.e., the eavesdropper gains access to the symbols $\boldsymbol{e} = (c_5, c_6, c_8, c_{18} = c_5 + c_6 + c_8)$, is depicted. It can easily be seen that $H(\boldsymbol{r}) = 4p \log q$, $H(\boldsymbol{e}) = 3p \log q$, and $H(\boldsymbol{r}|\boldsymbol{e}, \boldsymbol{m}) = (4 - 3)p \log q$. Therefore, there is no information leakage to the eavesdropper.*

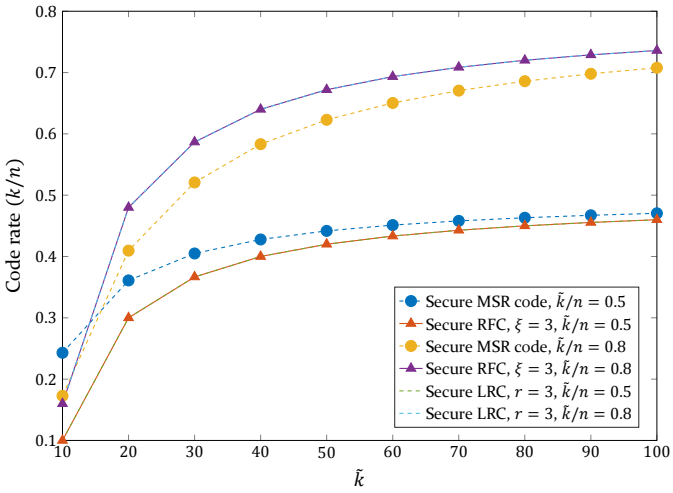

Figure II.3: Comparison of code rates for different classes of secure ECCs for the $(2, 2)$ eavesdropper model.

5 Numerical Results

We compare the proposed secure RFCs with the secure MSR codes and secure LRCs in [2] in terms of the maximum code rate k/n that allows to achieve security. In particular, we consider (r, δ) $d_{\min}-$optimal LRCs [2], where r is the code locality (and thus has an analogous meaning to ξ for RFCs) and $d_{\min}$ is the minimum distance of the code. Each local group of such a code can be seen as a subcode (punctured from the LRC) of minimum distance at least δ.

In Fig. II.3, we fix the code rate of the inner code (RFC, LRC, or MSR code), $\tilde{k}/n$, to 0.5 and 0.8, and plot the achievable code rates (the maximum k/n that allows to achieve security) for the $(2, 2)$ eavesdropper model as a function of $\tilde{k}$. Note that $\tilde{k}/n$ is an upper bound on the achievable code rate k/n, since to achieve security a number of random symbols needs to be appended to the message of length k. Note also that n is the total number of storage nodes. We remark that, unlike LRC- and RFC-based DSSs, where each code symbol is stored in a different storage node, for MSR codes each storage node stores $\alpha = (n - \tilde{k})^{\tilde{k}-1}$ code symbols. For a fair comparison between RFCs and LRCs, we set $r = \xi$ and $\delta = 2$. It can be seen that the achievable code rates for secure RFCs and secure LRCs are identical. On the other hand, secure RFCs yield higher achievable code rates compared to secure MSR codes for $\tilde{k}/n = 0.8$, while the opposite is observed for $\tilde{k}/n = 0.5$.

6 Conclusion

We proposed a code construction based on RFCs that is secure against the (ℓ_1, ℓ_2) eavesdropper model. We gave a necessary and sufficient condition for the information leakage to the eavesdropper to be zero, and subsequently proved that the proposed construction is completely secure. The proposed secure RFCs yield the

same achievable code rates as LRCs, and higher than MSR codes (when the code rate of the underlying code is high enough). An interesting extension of this work would be the design of secure and repair-efficient vector RFCs, i.e., code symbols are distributed over the storage nodes, each containing $\alpha > 1$ code symbols.

References

[1] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "Information-theoretically secure regenerating codes for distributed storage," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Houston, TX, Dec. 2011.

[2] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 60, no. 1, pp. 212–236, Jan. 2014.

[3] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems Inf. Transmiss.*, vol. 21, pp. 1–12, Jul. 1985.

[4] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Cambridge, MA, Jul. 2012.

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[6] M. Asteris and A. G. Dimakis, "Repairable fountain codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 1037–1047, May 2014.

Paper III

Achieving Maximum Distance Separable Private Information Retrieval Capacity With Linear Codes

Siddhartha Kumar, Hsuan-Yin Lin, Eirik Rosnes, and Alexandre Graell i Amat

Submitted to IEEE Transactions on Information Theory, revised August 2018.

The paper was presented in part at the IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, June 2017, at the IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, June 2018, and at the IEEE Information Theory Workshop (ITW), Guangzhou, China, November 2018.

The layout has been revised.

Abstract

We propose three private information retrieval (PIR) protocols for distributed storage systems (DSSs) where data is stored using an arbitrary linear code. The first two protocols, named Protocol 1 and Protocol 2, achieve privacy for the scenario with noncolluding nodes. Protocol 1 requires a file size that is exponential in the number of files in the system, while the file size required for Protocol 2 is independent of the number of files and is hence simpler. We prove that, for certain linear codes, Protocol 1 achieves the maximum distance separable (MDS) PIR capacity, i.e., the maximum PIR rate (the ratio of the amount of retrieved stored data per unit of downloaded data) for a DSS that uses an MDS code to store any given (finite and infinite) number of files, and Protocol 2 achieves the *asymptotic* MDS-PIR capacity (with infinitely large number of files in the DSS). In particular, we provide a necessary and a sufficient condition for a code to achieve the MDS-PIR capacity and prove that cyclic codes, Reed-Muller (RM) codes, and a class of distance-optimal local reconstruction codes achieve both the *finite* MDS-PIR capacity (i.e., with any given number of files) and the asymptotic MDS-PIR capacity with Protocols 1 and 2, respectively. Furthermore, we present a third protocol, Protocol 3, for the scenario with multiple colluding nodes, which can be seen as an improvement of a protocol recently introduced by Freij-Hollanti *et al.* Similar to the noncolluding case, we provide a necessary and a sufficient condition to achieve the maximum possible PIR rate of Protocol 3. Moreover, we provide a particular class of codes that is suitable for this protocol and show that RM codes achieve the maximum possible PIR rate for the protocol. For all three protocols, we present an algorithm to optimize their PIR rates.

1 Introduction

In data storage applications, besides resilience against disk failures and data protection against illegitimate users, the privacy may also be of concern. For instance, one may be interested in designing a storage system in which a file can be downloaded without revealing any information of which file is actually downloaded to the servers storing it. This form of privacy is usually referred to as *private information retrieval* (PIR). PIR is important to, e.g., protect users from surveillance and monitoring.

PIR protocols were first studied in the computer science literature by Chor *et al.* in [1, 2], which introduced the concept of an n-server PIR protocol, where a binary storage node is replicated among n servers (referred to as nodes) and the aim is to privately retrieve a single bit from the storage nodes while minimizing

the total upload and download communication cost. Additionally, an n-server PIR protocol assumes that the n nodes do not collude in order to reveal the identity of the requested bit. The communication cost in [1] was further reduced in [3–5]. Since then, coded PIR schemes have been introduced, where data is encoded (as opposed to simply being replicated) across several nodes [6]. With the advent of distributed storage systems (DSSs), where the user data is encoded and then stored on n nodes, there has been an increasing interest in implementing coded PIR protocols for these systems.

In recent years PIR has become an active research area in the information theory community with a fundamental difference in the measurement of efficiency. In the information-theoretic sense, the message sizes are much larger than the size of all queries sent to the storage nodes. Thus, rather than accounting for both the upload and the download cost, download cost forms the primary measurement of efficiency as the upload cost can be neglected. The ratio between the requested file size to the amount of downloaded data is referred to as the PIR rate, where higher PIR rates mean higher efficiency. The highest achievable PIR rate for any n-server PIR protocol is referred to as the PIR capacity.

In the information theory literature, the authors in [7] were the first to present PIR protocols for DSSs where data is stored using codes from two explicit linear code constructions. In [8], the authors presented upper bounds on the tradeoff between the storage and the PIR rates for a certain class of linear PIR protocols. In [9], Fazeli *et al.* introduced PIR codes which, when used in conjunction with traditional n-server PIR protocols, allow to achieve PIR on DSSs. These codes achieve high code rates without sacrificing on the communication cost of an n-server PIR protocol. In [10], given an arbitrary number of files, the authors derived the PIR capacity for noncolluding and replicated databases, where the data can be seen as being encoded by a particular class of maximum distance separable (MDS) codes, the repetition codes. For the case of noncolluding nodes, Banawan and Ulukus [11] derived the PIR capacity for DSSs using an $[n, k]$ MDS code to store a given number of files, referred to as the MDS-PIR capacity. In this paper, we will refer to the MDS-PIR capacity for a given finite number of files as the *finite MDS-PIR capacity*. In [12], a PIR protocol for MDS-coded DSSs and noncolluding nodes was proposed and shown to achieve the finite MDS-PIR capacity. A PIR protocol for the case of colluding nodes was also proposed in [12]. Here, we will refer to the MDS-PIR capacity for an infinite number of files as the *asymptotic MDS-PIR capacity*. Note that the MDS-PIR capacity for the colluding case is still unknown in general, except for some special cases [13] and for repetition codes [14]. The problem of *symmetric* PIR for DSSs was recently considered in [15], where an expression for the symmetric PIR capacity for linear schemes in the general case of colluding nodes and an MDS linear storage code was derived. In the symmetric case, the user should not only be able to privately retrieve the requested file from the system, but also learn nothing about the other files stored from the retrieved data. See also the related work [16], which deals with replicated databases. The PIR capacity for the case where a given number of storage nodes fail to respond (so-called robust PIR) was given in [14] for the scenario of colluding servers with replication coding.

In the storage community, it is well-known that MDS codes are inefficient in

the repair of failed nodes. In particular, they have large repair locality, i.e., the repair of a failed node requires contacting a large number of nodes.[1] Repair is essential to maintain the initial state of reliability of the DSS. To address low repair locality, Pyramid codes [19], locally repairable codes [20], local reconstruction codes (LRCs) [21, 22], and locally recoverable codes [23] are some non-MDS codes that have been proposed. These four classes of codes follow the same design philosophy and for simplicity, we will refer to them generically as LRCs. Following the motivation of using non-MDS codes in DSSs, the authors of [24] presented a PIR protocol for DSSs that store data using arbitrary linear codes for the scenario of noncolluding nodes. Independently, Freij-Hollanti *et al.* in [25] presented a PIR protocol that ensures privacy even when a subset of at most $n - k$ nodes collude. The protocol is based on two codes, the storage code and the *query code*, which defines the queries. The retrieval process is then characterized by the *retrieval code*, which is the Hadamard product of these two codes. The PIR rate of the protocol is upperbounded by $(n - \tilde{k})/n$, where $\tilde{k}$ is the dimension of the retrieval code. The authors showed that with generalized RS (GRS) codes for the storage and query codes, the upper bound on the PIR rate is achieved. To the best of our knowledge, in the asymptotic regime when the number of files tends to infinity, the PIR rate $(n - \tilde{k})/n$ is the highest achievable PIR rate known so far. Moreover, they showed that their protocol could work with certain non-MDS codes. However, for non-MDS codes (e.g., Reed-Muller (RM) codes where considered in [26]) the PIR rates that can be achieved by the protocol in [25] are lower than the upper bound $(n - \tilde{k})/n$.

In this paper, as an extension of [24], we present three PIR protocols for DSSs using arbitrary linear codes. These protocols share the fact that all of them are constructed by making use of correctable erasure patterns and information sets of the underlying storage code. We first focus on the noncolluding scenario and propose two PIR protocols, referred to as Protocol 1 and Protocol 2. Protocol 1 requires a file size that is exponential in the number of files in the system, while the file size required for Protocol 2 is independent of the number of files and is therefore simpler. Furthermore, Protocol 1 is designed such that its PIR rate depends on the number of files in the system, while Protocol 2 is such that its PIR rate is independent of the number of files. We prove that, interestingly, for certain non-MDS code families, Protocol 1 achieves the finite MDS-PIR capacity (and hence the asymptotic MDS-PIR capacity as well) and Protocol 2 achieves the asymptotic (i.e., when the number of files tends to infinity) MDS-PIR capacity. Thus, we show that the MDS property required to achieve the MDS-PIR capacity in [10–12] is not necessary and overly restrictive. In particular, we give a sufficient condition based on automorphisms and a necessary condition connected to the generalized Hamming weights of the underlying storage code to achieve the MDS-PIR capacity for Protocols 1 and 2. We prove that cyclic codes, RM codes, and distance-optimal information locality codes achieve the finite MDS-PIR capacity (and thus the asymptotic MDS-PIR capacity, too) with Protocol 1 and the asymptotic MDS-PIR capacity with Protocol 2. For other codes, we present an

[1] In a parallell line of work, schemes for efficient repair (in terms of repair bandwidth) of Reed-Solomon (RS) codes have been proposed [17, 18].

optimization algorithm for Protocols 1 and 2 such that they can achieve the best possible PIR rates.

We furthermore present a third protocol, Protocol 3, for the scenario of multiple colluding nodes and non-MDS storage codes. This protocol is based on and improves the protocol in [25, 26], in the sense that it achieves higher PIR rates. We extend the necessary and the sufficient condition from the noncolluding case to provide joint conditions on the storage and query codes to achieve the upper bound $(n - \tilde{k})/n$ on the PIR rate of Protocol 3. Moreover, we show that Protocol 3 achieves the upper bound $(n - \tilde{k})/n$ on the PIR rate for RM codes and some non-MDS codes. We also provide an optimization algorithm for the protocol to optimize the PIR rate. Such an optimization is in itself an extension of the optimization algorithm for Protocols 1 and 2 for the case of noncolluding nodes. Besides GRS and RM codes as in [25, 26], we also prove that $(\mathcal{U}|\mathcal{U}+\mathcal{V})$ codes [27] with $\mathcal{U}$ being an arbitrary linear code and $\mathcal{V}$ a repetition code can be used in conjunction with Protocol 3. We finally give examples of all-symbol locality LRCs with good PIR rates.

The main contributions of the paper are summarized in the following:

- For the noncolluding case, we propose two PIR protocols, Protocol 1 and Protocol 2 (Sections 4 and 5), and provide a necessary and a sufficient condition for a code to achieve the MDS-PIR capacity under these protocols (Theorems 3 and 4, respectively, in Section 6).
- For the noncolluding case, we show that important classes of non-MDS codes, namely cyclic codes, RM codes, and distance-optimal information locality codes achieve the finite MDS-PIR capacity and the asymptotic MDS-PIR capacity under Protocols 1 and 2, respectively (Corollaries 7 and 8 and Theorem 5, respectively, in Section 6).
- For the colluding case, we propose Protocol 3 that achieves higher asymptotic PIR rates for non-MDS codes (equal to its upper bound) than the best known protocol [25, 26]. Similar to the noncolluding case, a necessary and a sufficient condition for the protocol to achieve PIR rates equal to its upper bound is provided (Corollary 10 and Theorem 8, respectively, in Section 8). Independently of this work, by using an approach similar to ours, in [28] the authors modified the protocol in [26] and showed that the PIR rate $(n-\tilde{k})/n$ is achievable for *transitive codes*.[2] However, the protocol in [28] requires a much larger number of stripes and query sizes than our proposed Protocol 3, since it is based on transitive subgroups of the automorphishm groups of the storage and query codes, and thus is less practical.
- For both the noncolluding and colluding cases, we provide an algorithm that optimizes the PIR rate of the underlying code (Sections 7 and 8.5).

The remainder of this paper is organized as follows. We provide some definitions and preliminaries in Section 2. In Section 3, we provide a general system

[2]Note that the proposed sufficient condition (Theorem 8) is not equivalent to the concept of transitive codes in [28] when there are at least 2 colluding nodes (in the noncolluding case the concept of transitive codes in [28] reduces to our sufficient condition (Theorem 4)).

model for the three PIR protocols proposed in the paper. In Section 4 and **??**, we present Protocols 1 and 2 for the scenario with noncolluding nodes. In Section 6, we give a necessary and a sufficient condition for codes to achieve the MDS-PIR capacity and prove that several families of codes achieve it. In Section 7, we give an optimization algorithm to optimize the PIR rate. In Section 8, we consider the scenario with colluding nodes and propose Protocol 3. In the same section, we also present a family of storage codes that can be used with this protocol. Lastly, we provide a necessary and a sufficient condition to achieve an upper bound on the PIR rate for this protocol, and we show that RM codes satisfy the sufficient condition and thus achieve the upper bound on the PIR rate of Protocol 3. In Section 9, we optimize Protocols 1, 2, and 3 to maximize their PIR rates for different storage codes under the scenarios of noncolluding and colluding nodes. Finally, some conclusions are drawn in Section 10.

1.1 Notations and Conventions

In this paper, we use the following notations. We use lower case bold letters to denote vectors, upper case bold letters to denote matrices, and calligraphic upper case letters to denote sets. For example: $\boldsymbol{x}$, $\boldsymbol{X}$, and $\mathcal{X}$ denote a vector, a matrix, and a set, respectively. An identity matrix of order m is denoted as $\boldsymbol{I}_m$. An all-zero matrix of dimensions $a \times b$ is denoted as $\boldsymbol{0}_{a\times b}$, while an all-one matrix of dimensions $a \times b$ is referred to as $\boldsymbol{1}_{a\times b}$. $(\cdot)^\mathsf{T}$ represents the transpose of its argument and $\langle \cdot, \cdot \rangle$ denotes the scalar dot product between two vectors. The operator $\circ$ represents the Hadamard product. As such, $\boldsymbol{x} \circ \boldsymbol{y}$ represents the Hadamard product of two length-n vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Consider the column vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_a$, then $(\boldsymbol{x}_1 | \ldots | \boldsymbol{x}_a)$ represents the horizontal concatenation of the column vectors. Similarly, the horizontal concatenation of the matrices $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_a$, all with the same number of rows, will be denoted by $(\boldsymbol{X}_1 | \ldots | \boldsymbol{X}_a)$. We represent a submatrix of $\boldsymbol{X}$ that is restricted in columns by the set $\mathcal{I}$ and in rows by the set $\mathcal{J}$ by $\boldsymbol{X}|_\mathcal{I}^\mathcal{J}$, and the matrix rank of $\boldsymbol{X}$ by $\mathrm{rank}(\boldsymbol{X})$. The function $\mathsf{LCM}(a, b)$ computes the lowest common multiple of two real numbers a and b, and $a \mid b$ denotes that a is a divisor of b, while the function $\mathsf{H}(\cdot)$ represents the entropy of its argument.

In the rest of the paper, $\mathcal{C}$ will denote a linear code over a finite field $\mathrm{GF}(q)$. The operations over $\mathrm{GF}(q)$, such as addition, multiplication, etc., will be clearly understood from the context. We use the customary code parameters $[n, k]$ to refer to a code of blocklength n and dimension k, having a code rate $R^\mathcal{C} = k/n$. The dimension of a code $\mathcal{C}$ will sometimes be denoted by $\dim(\mathcal{C})$. Furthermore, $[n, k, d_{\min}^\mathcal{C}]$ represents an $[n, k]$ code of minimum Hamming distance $d_{\min}^\mathcal{C}$. Since a code $\mathcal{C}$ can be seen as a codebook matrix, the shortened and punctured codes are denoted by $\mathcal{C}|_\mathcal{I}^\mathcal{J}$, with column indices $\mathcal{I}$ and row coordinates $\mathcal{J}$. In addition, $\boldsymbol{H}^\mathcal{C}$, $\boldsymbol{G}^\mathcal{C}$, and $\mathcal{C}^\perp$ represent a parity-check matrix, a generator matrix, and the dual code, respectively, of $\mathcal{C}$. We denote by $\mathbb{N}$ the set of all positive integers, $\mathbb{N}_a \triangleq \{1, 2, \ldots, a\}$, and $\mathbb{N}_{n_1:n_2} \triangleq \{n_1, n_1 + 1, \ldots, n_2\}$ for two positive integers $n_1 \leq n_2$, $n_1, n_2 \in \mathbb{N}$. The Hamming weight of a binary vector $\boldsymbol{x}$ is denoted by $w_\mathsf{H}(\boldsymbol{x})$, while the support of a vector $\boldsymbol{x}$ (either binary or nonbinary), i.e., the set of nonzero entries of $\boldsymbol{x}$, will be denoted by $\chi(\boldsymbol{x})$. Note that sometimes, for the sake of convenience, we

will omit the superscripts and/or the subscripts if the arguments we refer to are contextually unambiguous. Also, with some abuse of language, the index of a coordinate of a vector is sometimes referred to simply as the coordinate. With some abuse of language, we sometimes interchangeably refer to binary vectors as erasure patterns under the implicit assumption that the ones represent erasures. An erasure pattern (or binary vector) $\boldsymbol{x}$ is said to be correctable by a code $\mathcal{C}$ if matrix $\boldsymbol{H}^{\mathcal{C}}|_{\chi(\boldsymbol{x})}$ has rank $|\chi(\boldsymbol{x})|$. Finally, for ease of notation, we will refer to a matrix with constant row weight, constant column weight, and constant row and column weight equal to a as an a-row regular, a-column regular, and a-regular matrix, respectively.

2 Definitions and Preliminaries

In this section, we review some basic notions in coding theory and some classes of codes that will be used throughout the paper.

Definition 1. *Let $\mathcal{C}$ be an $[n, k]$ code defined over $\mathrm{GF}(q)$. A set of coordinates of $\mathcal{C}$, $\mathcal{I} \subseteq \mathbb{N}_n$, of size k is said to be an* information set *if and only if $\boldsymbol{G}^{\mathcal{C}}|_{\mathcal{I}}$ is invertible.*

Definition 2. *Let $\mathcal{D}$ be a subcode of an $[n, k]$ code $\mathcal{C}$. The* support *of $\mathcal{D}$ is defined as*

$$\chi(\mathcal{D}) \triangleq \{j \in \mathbb{N}_n \colon \exists\, \boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{D}, x_j \neq 0\}.$$

It is noted that

$$\chi(\mathcal{D}) = \bigcup_{\boldsymbol{x} \in \mathcal{D}} \chi(\boldsymbol{x}),$$

where $\chi(\boldsymbol{x})$ denotes the support of $\boldsymbol{x}$.

Next, we introduce the concept of generalized Hamming weights [29].

Definition 3. *The s-th generalized Hamming weight of an $[n, k]$ code $\mathcal{C}$, denoted by $d_s^{\mathcal{C}}$, $s \in \mathbb{N}_k$, is defined as the cardinality of the smallest support of an s-dimensional subcode of $\mathcal{C}$, i.e.,*

$$d_s^{\mathcal{C}} \triangleq \min\left\{\left|\chi(\mathcal{D})\right| : \mathcal{D} \text{ is an } [n, s] \text{ subcode of } \mathcal{C}\right\}.$$

For the sequel, we introduce the notion of Hadamard product [30] of vector spaces.

Definition 4. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two vector spaces in $\mathrm{GF}(q)^n$. The Hadamard product of $\mathcal{X}$ and $\mathcal{Y}$, denoted by $\mathcal{X} \circ \mathcal{Y}$, is defined as the space in $\mathrm{GF}(q)^n$ generated by the Hadamard products $\boldsymbol{x} \circ \boldsymbol{y}$ for all $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$.*

2.1 Reed-Muller Codes

We review the family of binary linear RM codes [31] and then quickly summarize a result related to information sets of an RM code. We adapt the concept and definition from [27, Ch. 13], and the details can be found therein.

Definition 5. *For a given $m \in \mathbb{N}$, the v-th order binary RM code $\mathcal{R}(v,m)$ is an $[n,k]$ code with length $n = 2^m$ and code dimension $k = \sum_{i=0}^{v} \binom{m}{i}$ for $v \in \{0\} \cup \mathbb{N}_m$, constructed as the linear space spanned by the set of all m-variable Boolean monomials of degree at most v.*

For example, $\mathcal{R}(2,3)$ can be viewed as the linear space spanned by the set of Boolean monomials $\{1, z_1, z_2, z_3, z_1z_2, z_1z_3, z_2z_3\}$.

Next, we introduce a way to number the coordinate index of an RM codeword. Without loss of generality, since there are in total $n = 2^m$ codeword coordinates, each coordinate index $j \in \mathbb{N}_{2^m}$ can be described by a binary column m-tuple $\boldsymbol{\mu} = (\mu_1, \dots, \mu_m)^\mathsf{T}$, $\mu_i \in \mathrm{GF}(2)$, such that

$$j \triangleq 1 + \sum_{i=1}^{m} \mu_i 2^{i-1}. \tag{III.1}$$

For instance, for $m = 4$, the 7-th coordinate of an RM code corresponds to $(0\ 1\ 1\ 0)^\mathsf{T}$. Hence, a set of coordinates $\mathcal{I} \subseteq \mathbb{N}_n$ can alternatively be written as a set of corresponding m-tuples for RM codes.

Let $\boldsymbol{V}$ be an $m \times m$ invertible matrix over $\mathrm{GF}(2)$ and $\boldsymbol{\sigma} \in \mathrm{GF}(2)^{m\times 1}$ be a length-m binary column vector. It is well-known that the coordinate transformation mapping $\boldsymbol{\mu}$ onto $\boldsymbol{V}\boldsymbol{\mu} + \boldsymbol{\sigma}$ is an *automorphism* for the RM code [27, Ch. 13]. Hence, we obtain the following important property concerning information sets for RM codes.

Proposition 2. *Let $\mathcal{I} = \{\boldsymbol{\mu}_i\}_{i\in\mathbb{N}_k} \subseteq \mathrm{GF}(2)^{m\times 1}$ be an information set of $\mathcal{R}(v,m)$, and define the following coordinate set*

$$\mathcal{I}' \triangleq \{g(\boldsymbol{\mu}_i) \colon \boldsymbol{\mu}_i \in \mathcal{I}\},$$

where $g(\boldsymbol{\mu}) \triangleq \boldsymbol{V}\boldsymbol{\mu} + \boldsymbol{\sigma}$ for an arbitrary $m \times m$ invertible binary matrix $\boldsymbol{V}$ and an arbitrary $\boldsymbol{\sigma} \in \mathrm{GF}(2)^{m\times 1}$. Then, $\mathcal{I}'$ also forms an information set of the RM code.

For the sake of simplicity, throughout the paper we assume $\boldsymbol{V} = \boldsymbol{I}_m$.

Example 1. *Consider the RM code $\mathcal{R}(1,3)$ with generator matrix*

$$\boldsymbol{G}^{\mathcal{R}(1,3)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The i-th row of $\boldsymbol{G}^{\mathcal{R}(1,3)}$ corresponds to the i-th monomial of the set of Boolean monomials $\{1, z_1, z_2, z_3\}$, $i \in \mathbb{N}_4$. It can be seen that any codeword of $\mathcal{R}(1,3)$ corresponds to a linear combination of the Boolean monomials as $w_0 1 + w_1 z_1 + w_2 z_3 + w_3 z_3$, where $w_i \in \mathrm{GF}(2)$, $i \in \mathbb{N}_4$. Clearly, $\mathcal{I} = \{(0\ 0\ 0)^\mathsf{T}, (1\ 0\ 0)^\mathsf{T}, (0\ 1\ 0)^\mathsf{T}, (0\ 0\ 1)^\mathsf{T}\}$ forms an information set for $\mathcal{R}(1,3)$. Pick an automorphism g with $\boldsymbol{V} = \boldsymbol{I}_3$ being the identity matrix and $\boldsymbol{\sigma} = (0\ 0\ 1)^\mathsf{T}$. Then,

$$\begin{aligned} \mathcal{I}' &= \{g(\boldsymbol{\mu}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \colon \boldsymbol{\mu} \in \mathcal{I}\} \\ &= \{(0\ 0\ 1)^\mathsf{T}, (1\ 0\ 1)^\mathsf{T}, (0\ 1\ 1)^\mathsf{T}, (0\ 0\ 0)^\mathsf{T}\} \end{aligned}$$

is also an information set of $\mathcal{R}(1,3)$.

The following lemma shows how to determine an information set for an RM code.

Lemma 1. *Consider the RM code $\mathcal{R}(v,m)$ with $v \in \{0\} \cup \mathbb{N}_m$, $m \in \mathbb{N}$. Then, the set of m-tuples given by*

$$\mathcal{I} \triangleq \{\boldsymbol{\mu} \in \mathrm{GF}(2)^{m\times 1} \colon w_{\mathsf{H}}(\boldsymbol{\mu}) \leq v\}$$

is an information set for $\mathcal{R}(v,m)$.

Proof. The proof is based on the definition of RM codes. The details are given in Appendix A. □

Lemma 1 can be extended to nonbinary generalized RM codes (see the comprehensive work in [32] that determines the information sets for generalized RM codes).

2.2 Local Reconstruction Codes

LRCs are a family of codes that are used in DSSs because of their low repair locality, i.e., they need to contact a relatively low number of nodes in order to repair a failed node. Systematic codes that focus on lowering the locality for the systematic nodes (i.e., the nodes that store the systematic code symbols; see the system model in Section 3) are referred to as *information locality* codes. Examples of such codes are presented in [19, 21]. On the contrary, LRCs that achieve low locality for all nodes are referred to as *all-symbol locality* codes. The codes presented in [23] are examples of all-symbol locality codes. Formally, information locality codes are defined as follows.

Definition 6 ((r,δ) information locality code[22, Def. 2])**.** *An $[n,k]$ code is said to be an (r,δ) information locality code if there exist L_{c} punctured codes $\mathcal{C}_j \triangleq \mathcal{C}|_{\mathcal{S}_j}$ of $\mathcal{C}$ with column coordinate set $\mathcal{S}_j \subset \mathbb{N}_n$ for $j \in \mathbb{N}_{L_{\mathsf{c}}}$. Furthermore, $\{\mathcal{C}|_{\mathcal{S}_j}\}_{j\in\mathbb{N}_{L_{\mathsf{c}}}}$ must satisfy the following conditions:*

1. $|\mathcal{S}_j| \leq r+\delta-1$, $\forall\, j \in \mathbb{N}_{L_{\mathsf{c}}}$,
2. $d_{\mathsf{min}}^{\mathcal{C}_j} \geq \delta$, $\forall\, j \in \mathbb{N}_{L_{\mathsf{c}}}$, *and*
3. $\mathrm{rank}\big(\boldsymbol{G}|_{\bigcup_j \mathcal{S}_j}\big) = k$.

The overall code $\mathcal{C}$ has minimum Hamming distance $d_{\mathsf{min}}^{\mathcal{C}} \leq n-k+1-(\lceil k/r\rceil-1)(\delta-1)$ and can repair up to $\delta-1$ systematic nodes by contacting r storage nodes. In other words, Definition 6 says that there are L_{c} local codes in $\mathcal{C}$ each having a block length of at most $r+\delta-1$, a minimum Hamming distance of at least δ, and the union of all local codes forms an information set. Codes that achieve the upper bound on the minimum Hamming distance are known as distance-optimal (r,δ) information locality codes and have the following structure.

Definition 7 (Distance-optimal (r,δ) information locality code [22, Th. 2.2])**.** *Let $r \mid k$ such that $L_{\mathsf{c}} = k/r$. An (r,δ) information locality code $\mathcal{C}$ as defined in Definition 6 is distance-optimal if:*

1. *Each local code* $\mathcal{C}|_{\mathcal{S}_j}$, $j \in \mathbb{N}_{L_c}$, *is an* $[r+\delta-1, r]$ *MDS code defined by a parity-check matrix* $\boldsymbol{H}^{\mathcal{C}|_{\mathcal{S}_j}} = (\boldsymbol{P}_j | \boldsymbol{I}_{\delta-1})$ *of dimensions* $(\delta-1) \times (r+\delta-1)$ *and minimum Hamming distance* $d_{\min}^{\mathcal{C}|_{\mathcal{S}_j}} = \delta$.
2. *The sets* $\{\mathcal{S}_j\}_{j \in \mathbb{N}_{L_c}}$ *are disjoint, i.e.,* $\mathcal{S}_j \cap \mathcal{S}_{j'} = \emptyset$ *for all* $j, j' \in \mathbb{N}_{L_c}$, $j \neq j'$.
3. *The code* $\mathcal{C}$ *has a parity-check matrix of the form*

$$\boldsymbol{H} = \left(\begin{array}{cccccccc|c} \boldsymbol{P}_1 & \boldsymbol{I}_{\delta-1} & & & & & & & \\ & & \boldsymbol{P}_2 & \boldsymbol{I}_{\delta-1} & & & & & \\ & & & & \ddots & & & & \\ & & & & & & \boldsymbol{P}_{L_c} & \boldsymbol{I}_{\delta-1} & \\ \hline \boldsymbol{M}_1 & \boldsymbol{0} & \boldsymbol{M}_2 & \boldsymbol{0} & \cdots & \boldsymbol{M}_{L_c} & \boldsymbol{0} & \boldsymbol{I}_a \end{array}\right) \tag{III.2}$$

where the matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_{L_c}$ *are arbitrary matrices in* $\mathrm{GF}(q)$ *of dimensions* $(n - L_c(r+\delta-1)) \times r$, *and* $a \triangleq n - L_c(r+\delta-1)$.

For ease of exposition, we refer to the local parities as the parity symbols that take part in local codes, while the parity symbols that are not part of the L_c local codes are referred to as global parity symbols. According to Definition 7, there exist $n - L_c(r+\delta-1)$ global parities and $L_c(\delta-1)$ local parities. We partition the coordinates of these parities into $L+1$ sets, where $L \triangleq \left\lfloor \frac{n}{r+\delta-1} \right\rfloor$. For $i \in \mathbb{N}_{L+1}$, we have

$$\mathcal{P}_j = \begin{cases} \{(j-1)n_c + r + 1, \ldots, jn_c\} & \text{if } j \in \mathbb{N}_{L_c}, \\ \{(j-1)n_c + 1, \ldots, jn_c\} & \text{if } j \in \mathbb{N}_{L_c+1:L}, \\ \{Ln_c + 1, \ldots, n\} & \text{if } j = L+1, \end{cases} \tag{III.3}$$

where $n_c \triangleq r + \delta - 1$ is the block length of each local code. The sets $\mathcal{P}_j$, $j \in \mathbb{N}_{L_c}$, represent the coordinates of the local parities of the j-th local code $\mathcal{C}_j$. The remaining sets $\mathcal{P}_j$, $j \in \mathbb{N}_{L_c+1:L+1}$, represent the coordinates of the global parities of $\mathcal{C}$. As such, the set $\mathcal{P} = \bigcup_{j=1}^{L+1} \mathcal{P}_j$ represents the parity coordinates of $\mathcal{C}$.

2.3 UUV Codes

Consider an $[n_1, k_1]$ code $\mathcal{U}$ and an $[n_1, k_2]$ code $\mathcal{V}$ both over $\mathrm{GF}(q)$. An $[n = 2n_1, k = k_1 + k_2]$ $(\mathcal{U} \mid \mathcal{U} + \mathcal{V})$ code [27] (herein referred to as a UUV code) has codewords of the form $(\boldsymbol{u} \mid \boldsymbol{u} + \boldsymbol{v})$, where $\boldsymbol{u} \in \mathcal{U}$ and $\boldsymbol{v} \in \mathcal{V}$. A UUV code has generator matrix

$$\boldsymbol{G}^{\mathrm{UUV}} = \begin{pmatrix} \boldsymbol{G}^{\mathcal{U}} & \boldsymbol{G}^{\mathcal{U}} \\ \boldsymbol{0}_{k_2 \times n_1} & \boldsymbol{G}^{\mathcal{V}} \end{pmatrix},$$

where $\boldsymbol{G}^{\mathcal{U}}$ and $\boldsymbol{G}^{\mathcal{V}}$ are the generator matrices of $\mathcal{U}$ and $\mathcal{V}$, respectively. One can construct RM codes using UUV codes in an iterative manner [27, p. 374].

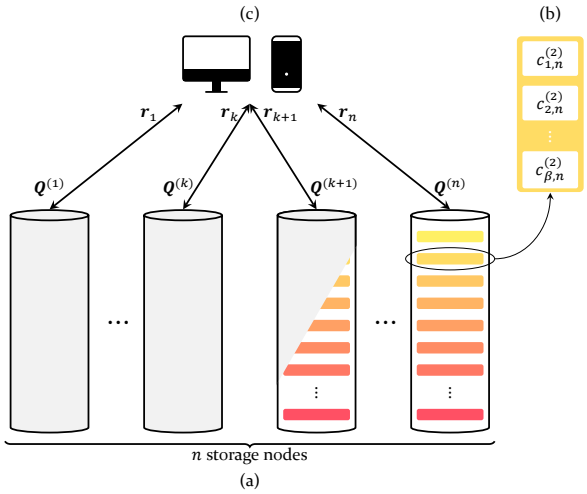

Figure III.1: System Model. (a) The colored boxes in each storage node represent the f coded chunks pertaining to the f files. (b) Coded chunk corresponding to the 2nd file in the n-th node. It consists of β code symbols, $c^{(2)}_{i,n}$, $i \in \mathbb{N}_\beta$. (c) The user sends the queries $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_n$, to the storage nodes and receives responses $\boldsymbol{r}_l$.

3 System Model

We consider a DSS that stores f files $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(f)}$, where each file $\boldsymbol{X}^{(m)} = (x^{(m)}_{i,j})$, $m \in \mathbb{N}_f$, can be seen as a $\beta \times k$ matrix over $\mathrm{GF}(p^{\alpha\ell})$, with β, k, α, $\ell \in \mathbb{N}$, and p some prime number. Each file is encoded using a linear code as follows. Let $\boldsymbol{x}^{(m)}_i = (x^{(m)}_{i,1}, \ldots, x^{(m)}_{i,k})$, $i \in \mathbb{N}_\beta$, be a message vector corresponding to the i-th row of $\boldsymbol{X}^{(m)}$. Each $\boldsymbol{x}^{(m)}_i$ is encoded by an $[n, k]$ code $\mathcal{C}$ over $\mathrm{GF}(q)$ with $q \triangleq p^\alpha$, having *subpacketization* α, into a length-n codeword $\boldsymbol{c}^{(m)}_i = \left(c^{(m)}_{i,1}, \ldots, c^{(m)}_{i,n}\right)$, where $c^{(m)}_{i,j} \in \mathrm{GF}(q^\ell)$, $j \in \mathbb{N}_n$. For $\alpha = 1$, the code $\mathcal{C}$ is referred to as a *scalar* code. Otherwise, the code is called a *vector* code [33]. The βf generated codewords $\boldsymbol{c}^{(m)}_i$ are then arranged in the array $\boldsymbol{C} = \left((\boldsymbol{C}^{(1)})^\top | \ldots | (\boldsymbol{C}^{(f)})^\top\right)^\top$ of dimensions $\beta f \times n$, where $\boldsymbol{C}^{(m)} = \left((\boldsymbol{c}^{(m)}_1)^\top | \ldots | (\boldsymbol{c}^{(m)}_\beta)^\top\right)^\top$ for $m \in \mathbb{N}_f$. For a given column j of $\boldsymbol{C}$, we denote the column vector $\left(c^{(m)}_{1,j}, \ldots, c^{(m)}_{\beta,j}\right)$ as a coded chunk pertaining to file $\boldsymbol{X}^{(m)}$. The f coded chunks in column j are stored on the j-th storage node, $j \in \mathbb{N}_n$, as shown in **??**(a). In case the $[n, k]$ code $\mathcal{C}$ is systematic, the nodes that store the systematic code symbols are referred to as systematic nodes.

3.1 Privacy Model

We consider a DSS where a set of T nodes may act as spies. Further, they might collude and hence they are referred to as colluding nodes. In addition, it is assumed that the remaining nonspy nodes do not collaborate with the spy nodes. The scenario of a single spy node ($T = 1$) in the DSS is analogous to having a sys-

tem with no colluding nodes. Let $\mathcal{T} \subset \mathbb{N}_n$, $|\mathcal{T}| = T$, denote the set of spy nodes in the DSS. The role of the spy nodes is to determine which file $\boldsymbol{X}^{(m)}$ is accessed by the user. We assume that the user does not know $\mathcal{T}$, since otherwise it can trivially achieve PIR by avoiding contacting the spy nodes. To retrieve file $\boldsymbol{X}^{(m)}$ from the DSS, the user sends a $d \times \beta f$ matrix query $\boldsymbol{Q}^{(l)}$ over $\mathrm{GF}(q) \subseteq \mathrm{GF}(q^\ell)$ to the l-th node for all $l \in \mathbb{N}_n$. The query matrices are represented in the form of d subquery vectors $\boldsymbol{q}_i^{(l)}$ of length βf as

$$\boldsymbol{Q}^{(l)} = \begin{pmatrix} \boldsymbol{q}_1^{(l)} \\ \boldsymbol{q}_2^{(l)} \\ \vdots \\ \boldsymbol{q}_d^{(l)} \end{pmatrix} = \begin{pmatrix} q_{1,1}^{(l)} & q_{1,2}^{(l)} & \cdots & q_{1,\beta f}^{(l)} \\ q_{2,1}^{(l)} & q_{2,2}^{(l)} & \cdots & q_{2,\beta f}^{(l)} \\ \vdots & \vdots & \cdots & \vdots \\ q_{d,1}^{(l)} & q_{d,2}^{(l)} & \cdots & q_{d,\beta f}^{(l)} \end{pmatrix}.$$

The i-th subqueries $\boldsymbol{q}_i^{(l)}$, $l \in \mathbb{N}_n$, of the n queries aim at recovering Γ unique code symbols[3] of the file $\boldsymbol{X}^{(m)}$. In response to the received query matrix, node l sends the column vector

$$\boldsymbol{r}_l = (r_{l,1}, \ldots, r_{l,d})^\mathsf{T} = \boldsymbol{Q}^{(l)}\big(c_{1,l}^{(1)}, \ldots, c_{\beta,l}^{(1)}, \ldots, c_{\beta,l}^{(f)}\big)^\mathsf{T}, \tag{III.4}$$

referred to as the response vector, back to the user as illustrated in **??**(c). We refer to $r_{l,i}$ as the i-th subresponse of the l-th node. Perfect information-theoretic PIR for such a scheme is defined in the following.

Definition 8. *Consider a DSS with n nodes storing f files in which a set of T nodes $\mathcal{T} = \{t_1, \ldots, t_T\} \subset \mathbb{N}_n$, $1 \leq |\mathcal{T}| = T \leq n - k$, act as colluding spies. A user who wishes to retrieve the m-th file sends the queries $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_n$, to the storage nodes, which return the responses $\boldsymbol{r}_l$. This scheme achieves perfect information-theoretic PIR if and only if*

$$\textit{Privacy:} \qquad \mathsf{H}\big(m|\boldsymbol{Q}^{(t_1)}, \ldots, \boldsymbol{Q}^{(t_T)}\big) = \mathsf{H}(m); \tag{III.5a}$$

$$\textit{Recovery:} \qquad \mathsf{H}\big(\boldsymbol{X}^{(m)}|\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n\big) = 0. \tag{III.5b}$$

Queries satisfying (III.5a) ensure that the file requested by the user is independent of the queries. Thus, the colluding nodes in $\mathcal{T}$ do not gain any additional information regarding which file is requested by the user by observing the queries. The recovery constraint in (III.5b) ensures that the user is able to recover the requested file from the responses sent by the DSS.

The efficiency of a PIR protocol is defined as the amount of retrieved data per unit of total amount of downloaded data, since it is assumed that the content of the retrieved file dominates the total communication cost [8, 12].

Definition 9. *The PIR rate of a PIR protocol, denoted by* R*, is the amount of information retrieved per downloaded symbol, i.e.,*

$$\mathsf{R} \triangleq \frac{\beta k}{nd}.$$

[3] In general, the i-th subqueries recover Γ_i unique code symbols such that among the $\sum_i \Gamma_i$ recovered code symbols there are βk distinct information symbols. However, for the sake of simplicity, we assume $\Gamma_i = \Gamma$ for all i.

Since the size of each file is βk, the parameters d and Γ should be chosen such that $\beta k = \Gamma d$. For the (file-independent) Protocols 2 and 3 in Sections 5 and 8 to be practical, we may select

$$\beta = \frac{\mathsf{LCM}(k,\Gamma)}{k} \quad \text{and} \quad d = \frac{\mathsf{LCM}(k,\Gamma)}{\Gamma}, \tag{III.6}$$

as it ensures the smallest values of β and d. This is not the case for Protocol 1 in Section 4, where β is exponential in the number of files in order achieve optimal PIR rates. By choosing the values above for β and d, the PIR rate for Protocols 2 and 3 become

$$\mathsf{R} = \frac{\Gamma}{n}.$$

We will write $\mathsf{R}(\mathcal{C})$ to highlight that the PIR rate depends on the underlying storage code $\mathcal{C}$. The maximum achievable PIR rate is the PIR capacity. It was shown in [11] that for the noncolluding case and for a given number of files f stored using an $[n,k]$ MDS code, the MDS-PIR capacity, denoted by C_f, is

$$\mathsf{C}_f \triangleq \frac{n-k}{n}\left[1-\left(\frac{k}{n}\right)^f\right]^{-1}. \tag{III.7}$$

Throughout the paper we refer to the capacity in (III.7) as the *finite MDS-PIR capacity* as it depends on the number of files. On the contrary when the number of files $f \to \infty$, the *asymptotic MDS-PIR capacity* is

$$\mathsf{C}_\infty \triangleq \frac{n-k}{n}. \tag{III.8}$$

It was shown in [8, Th. 3] that the PIR rate for a DSS with noncolluding nodes is upperbounded by C_∞ for a special class of linear retrieval schemes. In the case of colluding nodes, an explicit upper bound is currently unknown, as well as an expression for the MDS-PIR capacity. Some initial work for the case of two colluding nodes has recently been presented in [13].

4 Finite MDS-PIR Capacity-Achieving Protocol for the Noncolluding Case

In this section, we propose a capacity-achieving protocol, named Protocol 1, that achieves the finite MDS-PIR capacity in (III.7) for the scenario of noncolluding nodes. The protocol is inspired by the protocol introduced in [11].

4.1 PIR Achievable Rate Matrix

In [10], the concept of exploiting *side information* for PIR problems was introduced. By side information we mean additional redundant symbols not related to the requested file but downloaded by the user in order to maintain privacy.

These symbols can be exploited by the user to retrieve the requested file from the responses of the storage nodes. In [11, Sec. V.A], it was shown that a $[5, 3, 3]$ MDS storage code can be used to achieve the finite MDS-PIR capacity, where the side information is decoded by utilizing other code coordinates forming an information set in the code array. For instance, the authors chose the $\nu = 5$ information sets $\mathcal{I}_1 = \{1, 2, 3\}$, $\mathcal{I}_2 = \{1, 4, 5\}$, $\mathcal{I}_3 = \{2, 3, 4\}$, $\mathcal{I}_4 = \{1, 2, 5\}$, and $\mathcal{I}_5 = \{3, 4, 5\}$ of the $[5, 3, 3]$ MDS code in their PIR achievable scheme. Observe that in $\{\mathcal{I}_i\}_{i\in\mathbb{N}_5}$ each coordinate of the $[5, 3, 3]$ code appears exactly $\kappa = 3$ times. This motivates the following definition.

Definition 10. *Let $\mathcal{C}$ be an arbitrary $[n, k]$ code. A $\nu \times n$ binary matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ is said to be a* PIR achievable rate matrix *for $\mathcal{C}$ if the following conditions are satisfied.*

1. *The Hamming weight of each column of $\boldsymbol{\Lambda}_{\kappa,\nu}$ is κ, and*
2. *for each matrix row $\boldsymbol{\lambda}_i$, $i \in \mathbb{N}_\nu$, $\chi(\boldsymbol{\lambda}_i)$ always contains an information set.*

In other words, each coordinate j of $\mathcal{C}$, $j \in \mathbb{N}_n$, appears exactly κ times in $\{\chi(\boldsymbol{\lambda}_i)\}_{i\in\mathbb{N}_\nu}$, and every set $\chi(\boldsymbol{\lambda}_i)$ contains an information set.

Lemma 2. *If a matrix $\boldsymbol{\Lambda}_{\nu,\kappa}(\mathcal{C})$ exists for an $[n, k]$ code $\mathcal{C}$, then we have*

$$\frac{\kappa}{\nu} \geq \frac{k}{n},$$

where equality holds if $\chi(\boldsymbol{\lambda}_i)$, $i \in \mathbb{N}_\nu$, are all information sets.

Proof. Since by definition each row $\boldsymbol{\lambda}_i$ of $\boldsymbol{\Lambda}_{\nu,\kappa}$ always contains an information set, we have $w_{\mathrm{H}}(\boldsymbol{\lambda}_i) \geq k$, $i \in \mathbb{N}_\nu$. Let $\boldsymbol{v}_j$, $j \in \mathbb{N}_n$, be the j-th column of $\boldsymbol{\Lambda}_{\kappa,\nu}$. If we look at $\boldsymbol{\Lambda}_{\kappa,\nu}$ from both a row-wise and a column-wise point of view, we obtain

$$\nu k \leq \sum_{i=1}^{\nu} w_{\mathrm{H}}(\boldsymbol{\lambda}_i) = \sum_{j=1}^{n} w_{\mathrm{H}}(\boldsymbol{v}_j) = \kappa n,$$

from which the result follows. Clearly, equality holds if $\chi(\boldsymbol{\lambda}_i)$, $i \in \mathbb{N}_\nu$, are all information sets. □

Example 2. *Consider the* $[5, 3, 2]$ *systematic code with generator matrix*

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

One can easily verify that

$$\boldsymbol{\Lambda}_{2,3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

is a PIR achievable rate matrix for this code.

Before we state our main results, in order to clearly illustrate our example and the following achievability proof, we first introduce the following definition.

Definition 11. *For a given $\nu \times n$ PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C}) = (\lambda_{u,j})$, we define the PIR interference matrices $\boldsymbol{A}_{\kappa\times n} = (a_{i,j})$ and $\boldsymbol{B}_{(\nu-\kappa)\times n} = (b_{i,j})$ for the code $\mathcal{C}$ with*

$$a_{i,j} \triangleq u \text{ if } \lambda_{u,j} = 1, \forall j \in \mathbb{N}_n, i \in \mathbb{N}_\kappa, u \in \mathbb{N}_\nu,$$
$$b_{i,j} \triangleq u \text{ if } \lambda_{u,j} = 0, \forall j \in \mathbb{N}_n, i \in \mathbb{N}_{\nu-\kappa}, u \in \mathbb{N}_\nu.$$

Note that in Definition 11, for each $j \in \mathbb{N}_n$, distinct values of $u \in \mathbb{N}_\nu$ should be assigned for all i. Thus, the assignment is not unique in the sense that the order of the entries of each column of $\boldsymbol{A}$ and $\boldsymbol{B}$ can be permuted. For $j \in \mathbb{N}_n$, let $\mathcal{A}_j \triangleq \{a_{i,j} \colon i \in \mathbb{N}_\kappa\}$ and $\mathcal{B}_j \triangleq \{b_{i,j} \colon i \in \mathbb{N}_{\nu-\kappa}\}$. Note that the j-th column of $\boldsymbol{A}$ contains the row indices of $\boldsymbol{\Lambda}$ whose entries in the j-th column are equal to 1, while $\boldsymbol{B}$ contains the remaining row indices of $\boldsymbol{\Lambda}$. Hence, it can be observed that $\mathcal{B}_j = \mathbb{N}_\nu \setminus \mathcal{A}_j$, $\forall j \in \mathbb{N}_n$.

Definition 12. *By $\mathcal{S}(a|\boldsymbol{A}_{\kappa\times n})$ we denote the set of column coordinates of matrix $\boldsymbol{A}_{\kappa\times n} = (a_{i,j})$ in which at least one of its entries is equal to a, i.e.,*

$$\mathcal{S}(a|\boldsymbol{A}_{\kappa\times n}) \triangleq \{j \in \mathbb{N}_n \colon \exists\, a_{i,j} = a, i \in \mathbb{N}_\kappa\}.$$

The following claim can be directly verified.

Claim 1. *$\mathcal{S}(a|\boldsymbol{A}_{\kappa\times n})$ contains an information set of code $\mathcal{C}$, $\forall\, a \in \mathbb{N}_\nu$. Moreover, for an arbitrary entry $b_{i,j}$ of $\boldsymbol{B}_{(\nu-\kappa)\times n}$, $\mathcal{S}(b_{i,j}|\boldsymbol{A}_{\kappa\times n}) = \mathcal{S}(a|\boldsymbol{A}_{\kappa\times n}) \subseteq \mathbb{N}_n \setminus \{j\}$ if $b_{i,j} = a$.*

We illustrate the previous points in the following example.

Example 3. *Continuing with Example 2 and following Definition 11, we obtain*

$$\boldsymbol{A}_{2\times 5} = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 \\ 3 & 3 & 3 & 2 & 2 \end{pmatrix} \text{ and } \boldsymbol{B}_{1\times 5} = \begin{pmatrix} 1 & 2 & 2 & 3 & 3 \end{pmatrix}$$

for $\boldsymbol{\Lambda}_{2,3}$. One can see that $\mathcal{A}_j \cup \mathcal{B}_j = \mathbb{N}_3$, $\forall\, j \in \mathbb{N}_5$. Moreover, for instance, take $a = 1$, then $\mathcal{S}(1|\boldsymbol{A}_{2\times 5}) = \{2,3,4,5\}$ contains an information set of the $[5,3,2]$ systematic code of Example 2.

Now consider the two matrices

$$\begin{pmatrix} c^{(m)}_{\mu+a_{1,1},1} & c^{(m)}_{\mu+a_{1,2},2} & \cdots & c^{(m)}_{\mu+a_{1,n},n} \\ \vdots & & & \\ c^{(m)}_{\mu+a_{\kappa,1},1} & c^{(m)}_{\mu+a_{\kappa,2},2} & \cdots & c^{(m)}_{\mu+a_{\kappa,n},n} \end{pmatrix} \text{ and } \begin{pmatrix} c^{(m)}_{\mu+b_{1,1},1} & c^{(m)}_{\mu+b_{1,2},2} & \cdots & c^{(m)}_{\mu+b_{1,n},n} \\ \vdots & & & \\ c^{(m)}_{\mu+b_{\nu-\kappa,1},1} & c^{(m)}_{\mu+b_{\nu-\kappa,2},2} & \cdots & c^{(m)}_{\mu+b_{\nu-\kappa,n},n} \end{pmatrix}$$

of code symbols of the m-th file, where $\mu \in \mathbb{N}_{\beta-\nu} \cup \{0\}$. Observe that if the user knows the first matrix of code symbols, from Claim 1, since the coordinate set $\mathcal{S}(b_{i,j}|\boldsymbol{A}_{\kappa\times n}) \subseteq \mathbb{N}_n \setminus \{j\}$ contains an information set and the user knows the structure of the storage code $\mathcal{C}$, the code symbols $c^{(m)}_{\mu+b_{i,j},j}$ of the second matrix can be obtained. The intuition behind the definition of the interference matrices $\boldsymbol{A}$

and $\boldsymbol{B}$ is as follows. Assume that $\boldsymbol{X}^{(1)}$ is requested. Protocol 1 requires the user to download the side information $\sum_{m\neq 1} c^{(m)}_{\mu+a_{i,j},j}$ based on $\boldsymbol{A}$ and also to download code symbols as sums of code symbols from the requested file and the side information $\sum_{m\neq 1} c^{(m)}_{\mu+b_{i,j},j}$ based on $\boldsymbol{B}$. Claim 1 then indicates that the side information $\sum_{m\neq 1} c^{(m)}_{\mu+b_{i,j},j}$ based on $\boldsymbol{B}$ can be reliably decoded and hence we can obtain the requested file by cancelling the side information. Here, the entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ are respectively marked in red and blue. We are now ready to state Protocol 1.

4.2 Protocol 1

The proposed Protocol 1 generalizes the MDS-coded PIR protocol in [11] to DSSs where files are stored using an arbitrary linear code. Inspired by [10] and [11], a PIR capacity-achievable scheme should follow three important principles: 1) enforcing symmetry across storage nodes, 2) enforcing file symmetry within each storage node, and 3) exploiting side information of undesired symbols to retrieve new desired symbols. Note that principle 1) is in general not a necessary requirement for a feasible PIR protocol. However, as pointed out in [11] and [13], any PIR scheme can be made symmetric, hence we keep this principle for the purpose of simplifying the implementation.

The PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}$ for the given storage code $\mathcal{C}$ plays a central role in the proposed PIR protocol. Moreover, the protocol requires $\beta = \nu^f$ stripes and exploits the corresponding PIR interference matrices $\boldsymbol{A}_{\kappa\times n}$ and $\boldsymbol{B}_{(\nu-\kappa)\times n}$. Note that the number of stripes depends on the number of files f, hence Protocol 1 depends on f as well. We first outline the steps of the protocol, and then we will prove that the proposed protocol satisfies the perfect privacy condition of (III.5a) and results in the PIR rate of Theorem 1 below. Without loss of generality, we assume that the user wants to download the first file, i.e., $m = 1$. The algorithm is composed of four steps as described below. In Appendix B, we show that the algorithm generates $d \times \beta f$ query matrices $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_n$, with

$$d = \frac{\kappa}{\nu-\kappa}\left[\nu^f - \kappa^f\right].$$

Step 1. **Index Preparation**

For all files, the user interleaves the query indices for requesting the rows of $\boldsymbol{C}^{(m)}$ randomly and independently of each other. This is equivalent to generating the interleaved code array $\boldsymbol{Y}^{(m)} = \left((\boldsymbol{y}_1^{(m)})^\top | \ldots | (\boldsymbol{y}_\beta^{(m)})^\top\right)^\top$, $\forall\, m \in \mathbb{N}_f$, with rows

$$\boldsymbol{y}_i^{(m)} = \boldsymbol{c}_{\pi(i)}^{(m)}, \quad i \in \mathbb{N}_\beta,$$

where $\pi(\cdot) : \mathbb{N}_\beta \to \mathbb{N}_\beta$ is a random permutation, which is privately known to the user only. Therefore, when the user requests code symbols from each storage node, this procedure is designed to make the requested row indices to be random and independent of the requested file index.

Step 2. **Download Symbols in the i-th Repetition**

The user downloads the needed symbols in κ repetitions. In the i-th repetition, $i \in \mathbb{N}_\kappa$, the user downloads the required symbols in a total of f rounds. Each repetition comprises f rounds. In the m-th round, the user downloads symbols that are linear sums of code symbols from any m files, $m \in \mathbb{N}_f$. Using the terminology in [11], the user downloads two types of symbols in each round, *desired symbols*, which are directly related to the requested file index $m = 1$, and *undesired symbols*, which are not related to the requested file index $m = 1$, but are exploited to decode the requested file from the desired symbols. For the desired symbols, we will distinguish between round $\ell = 1$ and round $\ell \in \mathbb{N}_{2:f}$.

Undesired symbols. The undesired symbols refer to sums of code symbols which do not contain symbols from the requested file. For every round ℓ, $\ell \in \mathbb{N}_{f-1}$, the user downloads the code symbols

$$\Bigg\{ \sum_{m' \in \mathcal{M}} y^{(m')}_{((i-1)\mathsf{U}(f-1)+\mathsf{U}(\ell-1))\cdot v + a_{1,j}, j}, \cdots, \sum_{m' \in \mathcal{M}} y^{(m')}_{((i-1)\mathsf{U}(f-1)+\mathsf{U}(\ell-1))\cdot v + a_{\kappa,j}, j}, \\ \cdots, \sum_{m' \in \mathcal{M}} y^{(m')}_{((i-1)\mathsf{U}(f-1)+\mathsf{U}(\ell)-1)\cdot v + a_{1,j}, j}, \cdots, \sum_{m' \in \mathcal{M}} y^{(m')}_{((i-1)\mathsf{U}(f-1)+\mathsf{U}(\ell)-1)\cdot v + a_{\kappa,j}, j} \Bigg\} \tag{III.9}$$

for all $j \in \mathbb{N}_n$ and for all possible subsets $\mathcal{M} \subseteq \mathbb{N}_{2:f}$, where $|\mathcal{M}| = \ell$ and

$$\mathsf{U}(\ell) \triangleq \sum_{h=1}^{\ell} \kappa^{f-(h+1)} (v-\kappa)^{h-1}.$$

In contrast to undesired symbols, desired symbols are sums of code symbols which contain symbols of the requested file. The main idea of the protocol is that the user downloads desired symbols that are linear sums of requested symbols and undesired symbols from the previous round.

Desired symbols in the first round. In the first round, the user downloads $\kappa \cdot \mathsf{U}(1) = \kappa\kappa^{f-(1+1)}(v-\kappa)^{1-1} = \kappa^{f-1}$ undesired symbols from each storage node. However, these symbols cannot be exploited directly. Hence, due to symmetry, in round $\ell = 1$, the user downloads the κ^{f-1} desired symbols

$$\left\{ y^{(1)}_{\kappa^{f-1}(a_{i,j}-1)+1,j}, \cdots, y^{(1)}_{\kappa^{f-1}(a_{i,j}-1)+\kappa^{f-1},j} \right\} \tag{III.10}$$

from the j-th storage node, $j \in \mathbb{N}_n$, i.e., the user also downloads κ^{f-1} symbols for $m = 1$ from each storage node.

Desired symbols in higher rounds. In the $(\ell+1)$-th round, $\ell \in \mathbb{N}_{f-1}$, in order to exploit the side information, i.e., the undesired symbols from the previous

round, the user downloads the symbols

$$\Bigg\{ y^{(1)}_{\mathrm{D}(\ell-1)\cdot v+a_{i,j},j} + \sum_{m'\in\mathcal{M}_1} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell-1))\cdot v+b_{1,j},j}, \cdots,$$
$$y^{(1)}_{(\mathrm{D}(\ell-1)+(v-\kappa)-1)\cdot v+a_{i,j},j} + \sum_{m'\in\mathcal{M}_1} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell-1))\cdot v+b_{v-\kappa,j},j},$$
$$y^{(1)}_{(\mathrm{D}(\ell-1)+(v-\kappa))\cdot v+a_{i,j},j} + \sum_{m'\in\mathcal{M}_1} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell-1)+1)\cdot v+b_{1,j},j}, \cdots,$$
$$y^{(1)}_{\left[\mathrm{D}(\ell-1)+(\mathrm{U}(\ell)-\mathrm{U}(\ell-1))(v-\kappa)-1\right]\cdot v+a_{i,j},j} + \sum_{m'\in\mathcal{M}_1} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell)-1)\cdot v+b_{v-\kappa,j},j}, \cdots,$$
$$y^{(1)}_{(\mathrm{D}(\ell)-(v-\kappa))\cdot v+a_{i,j},j} + \sum_{m'\in\mathcal{M}_{\mathrm{N}(\ell)}} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell)-1)\cdot v+b_{1,j},j}, \cdots,$$
$$y^{(1)}_{(\mathrm{D}(\ell)-1)\cdot v+a_{i,j},j} + \sum_{m'\in\mathcal{M}_{\mathrm{N}(\ell)}} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell)-1)\cdot v+b_{v-\kappa,j},j} \Bigg\} \tag{III.11}$$

for all distinct ℓ-sized subsets $\mathcal{M}_1, \ldots, \mathcal{M}_{\mathrm{N}(\ell)} \subseteq \mathbb{N}_{2:f}$, where $j \in \mathbb{N}_n$, $\mathrm{N}(\ell) \triangleq \binom{f-1}{\ell}$, and

$$\mathrm{D}(\ell) \triangleq \kappa^{f-1} + \sum_{h=1}^{\ell} \binom{f-1}{h} \kappa^{f-(h+1)} (v-\kappa)^h.$$

This indicates that for each combination of files $\mathcal{M}_l$, $l \in \mathbb{N}_{\mathrm{N}(\ell)}$, the user downloads $\left[\mathrm{U}(\ell) - 1 - \mathrm{U}(\ell-1) + 1\right](v-\kappa)$ new desired symbols from each storage node, and since there are in total $\mathrm{N}(\ell)$ combinations of files, in each round $\mathrm{D}(\ell) - 1 - \mathrm{D}(\ell-1) + 1$ extra desired symbols are downloaded from each storage node.

Exploiting the side information. Using the fact that for a linear code $\mathcal{C}$ any linear combination of codewords is also a codeword, and together with Claim 1, it is not too hard to see that by fixing an arbitrary coordinate $j \in \mathbb{N}_n$, there always exist some coordinates $\mathcal{S} \subset \mathbb{N}_n \setminus \{j\}$ (see Claim 1) such that for a subset $\mathcal{M} \subseteq \mathbb{N}_{2:f}$ with $|\mathcal{M}| = \ell$, the so-called *aligned sum*

$$\left\{ \sum_{m'\in\mathcal{M}} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell-1))\cdot v+b_{1,j},j}, \cdots, \sum_{m'\in\mathcal{M}} y^{(m')}_{((i-1)\mathrm{U}(f-1)+\mathrm{U}(\ell)-1)\cdot v+b_{v-\kappa,j},j} \right\}$$

for $\ell \in \mathbb{N}_{f-1}$ and $i \in \mathbb{N}_\kappa$, can be decoded. Consequently, in the $(\ell+1)$-th round, from each storage node j we can collect code symbols related to $m = 1$ from the desired symbols, i.e.,

$$\left\{ y^{(1)}_{\mathrm{D}(\ell-1)\cdot v+a_{i,j},j}, \cdots, y^{(1)}_{(\mathrm{D}(\ell)-1)\cdot v+a_{i,j},j} \right\} \tag{III.12}$$

is obtained.

Symmetry across storage nodes. In the previous steps, since the user downloads the same amount of required symbols for each $j \in \mathbb{N}_n$ and for every round, symmetry across storage nodes is ensured.

File symmetry within each storage node. To ensure that the privacy condition of (III.5a) is fulfilled, we have to make sure that in each round $\ell \in \mathbb{N}_f$ of each repetition, for each storage node and for every combination of files $\mathcal{M} \subseteq \mathbb{N}_f$ with $|\mathcal{M}| = \ell$, the user requests the same number of linear sums $\eta(\mathcal{M}) \triangleq \sum_{m\in\mathcal{M}} y^{(m)}_{\eta_m,j}$, where η_m depends on m. This will be shown to be inherent from the protocol (see proof of Theorem 1 in Appendix B). In addition, since the user always requests the same number of linear sums for every combination of files, the scheme also implies that the frequencies of requested code symbols pertaining to each individual file index $m \in \mathbb{N}_f$ among all the linear sums are the same for each storage node.

Step 3. **Complete κ Repetitions**

The user repeats Step 2 until $i = \kappa$. We will show that by our designed parameters $\mathrm{U}(\ell)$ and $\mathrm{D}(\ell)$, the user indeed downloads in total $\beta = \nu^f$ stripes for the requested file (see again Appendix B).

Step 4. **Shuffling the Order of Queries to Each Node**

The order of the queries to each storage node is uniformly shuffled to prevent the storage node to be able to identify which file is requested from the index of the first downloaded symbol.

4.3 Achievable PIR Rate

The PIR rate, $\mathrm{R}(\mathcal{C})$, of Protocol 1 in Section 4.2 for a DSS where f files are stored using an arbitrary $[n, k]$ code $\mathcal{C}$ is given in the following theorem.

Theorem 1. *Consider a DSS that uses an $[n, k]$ code $\mathcal{C}$ to store f files. If a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ exists, then the PIR rate*

$$\mathrm{R}(\mathcal{C}) = \frac{(\nu-\kappa)k}{\kappa n}\left[1-\left(\frac{\kappa}{\nu}\right)^f\right]^{-1} \tag{III.13}$$

is achievable.

Proof. See Appendix B. □

We remark that from Lemma 2, (III.13) is smaller than or equal to the finite MDS-PIR capacity in (III.7) since

$$\mathrm{R}(\mathcal{C}) = \frac{\frac{\nu k}{\kappa n}\left[1-\frac{\kappa}{\nu}\right]}{\left[1-\left(\frac{\kappa}{\nu}\right)^f\right]} = \frac{\nu k}{\kappa n}\left[1+\frac{\kappa}{\nu}+\cdots+\left(\frac{\kappa}{\nu}\right)^{f-1}\right]^{-1} \leq \left[1+\frac{k}{n}+\cdots+\left(\frac{k}{n}\right)^{f-1}\right]^{-1}, \tag{III.14}$$

and it becomes the finite MDS-PIR capacity in (III.7) if there exists a matrix $\boldsymbol{\Lambda}_{\kappa,\nu}$ for $\mathcal{C}$ with $\frac{\kappa}{\nu} = \frac{k}{n}$. The inequality in (III.14) follows from Lemma 2.

Corollary 1. *If a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ exists for an $[n,k]$ code $\mathcal{C}$, then the finite MDS-PIR capacity in* (III.7) *is achievable.*

This gives rise to the following definition.

Definition 13. *A PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ for an $[n,k]$ code $\mathcal{C}$ is called an* MDS-PIR capacity-achieving *matrix, and $\mathcal{C}$ is referred to as an* MDS-PIR capacity-achieving *code.*

We remark that there might exist codes that are MDS-PIR capacity-achieving for which an MDS-PIR capacity-achieving matrix does not exist.

Note that the largest achievable PIR rate in the noncolluding case where data is stored using an arbitrary linear code is still unknown. Interestingly, it is observed from Lemma 2 and (III.14) that the largest possible achievable PIR rate for an arbitrary linear code with Protocol 1 strongly depends on the smallest possible value of $\frac{\kappa}{\nu}$ for which a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}$ exists. We stress that the existence of an MDS-PIR capacity-achieving matrix $\boldsymbol{\Lambda}_{\kappa,\nu}$ does not necessarily require $(\nu,\kappa) = (n,k)$, but $\frac{\kappa}{\nu} = \frac{k}{n}$.

Since the existence of a PIR achievable rate matrix is connected to the information sets of a code, we review a widely known result in coding theory.

Proposition 3 ([34, Th. 1.4.15])**.** *Let $\mathcal{C}$ be an $[n,k,d_{\min}]$ code. Then, every set of $n-d_{\min}+1$ coordinates of $\mathcal{C}$ contains an information set. Furthermore, $n-d_{\min}+1$ is the smallest number of coordinates with this property.*

Lemma 3. *For a given $[n,k,d^{\mathcal{C}}_{\min}]$ code $\mathcal{C}$, there always exists a PIR achievable rate matrix $\boldsymbol{\Lambda}_{k,\nu}$ with*

$$\nu = k + \min\left(k, d^{\mathcal{C}}_{\min} - 1\right).$$

Proof. See Appendix C. □

A lower bound on the largest possible achievable PIR rate obtained from Theorem 1 and Lemma 3 is given as follows.

Corollary 2. *Consider a DSS that uses an $[n,k,d^{\mathcal{C}}_{\min}]$ code $\mathcal{C}$ to store f files. Then, the PIR rate*

$$\mathrm{R}(\mathcal{C}) = \frac{\min\left(k, d^{\mathcal{C}}_{\min} - 1\right)}{n}\left[1 - \left(\frac{k}{k + \min\left(k, d^{\mathcal{C}}_{\min} - 1\right)}\right)^{f}\right]^{-1}$$

is achievable.

We remark that because every set of k coordinates of an $[n,k]$ MDS code is an information set, we can construct n information sets by cyclically shifting an arbitrary information set n times, hence an MDS-PIR capacity-achieving matrix $\boldsymbol{\Lambda}_{k,n}$ of an MDS code can be easily constructed. In other words, Protocol 1 with MDS codes is MDS-PIR capacity-achieving (see Corollary 1) and MDS codes are a class of MDS-PIR capacity-achieving codes.

Remark 1. *Since minimum storage regenerating (MSR) codes are MDS codes [35], it follows that MSR codes are also MDS-PIR capacity-achieving codes.*

In Section 7, we will give a necessary and a sufficient condition for an arbitrary linear code to be MDS-PIR capacity-achieving under Protocol 1 and give certain families of MDS-PIR capacity-achieving codes. For illustration purposes, in the next subsection, we give an example of an MDS-PIR capacity-achieving code.

4.4 A $[5, 3, 2]$ MDS-PIR Capacity-Achieving Code for $f = 2$

In this subsection, we compute the PIR achievable rate of a $[5, 3, 2]$ non-MDS code for a DSS that stores two files, $f = 2$, and show that it is MDS-PIR capacity-achieving.

Let $\mathcal{C}$ be a non-MDS $[5, 3, 2]$ binary code with generator matrix

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{III.15}$$

One can see that the $\nu \times n = 5 \times 5$ matrix

$$\boldsymbol{\Lambda}_{3,5} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

is a PIR achievable rate matrix. From $\boldsymbol{\Lambda}_{3,5}$, we obtain the following sets:

$$\chi(\boldsymbol{\lambda}_1) = \{1, 2, 3\}, \chi(\boldsymbol{\lambda}_2) = \{1, 4, 5\}, \chi(\boldsymbol{\lambda}_3) = \{2, 4, 5\},$$
$$\chi(\boldsymbol{\lambda}_4) = \{2, 3, 4\}, \chi(\boldsymbol{\lambda}_5) = \{1, 3, 5\}.$$

All of these sets contain an information set of $\mathcal{C}$ (see Definition 10). Furthermore, we get the following PIR interference matrices

$$\boldsymbol{A}_{3\times 5} = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 \\ 2 & 3 & 4 & 3 & 3 \\ 5 & 4 & 5 & 4 & 5 \end{pmatrix}, \quad \boldsymbol{B}_{2\times 5} = \begin{pmatrix} 3 & 2 & 2 & 1 & 1 \\ 4 & 5 & 3 & 5 & 4 \end{pmatrix}.$$

One can see that Claim 1 holds. For example, $\mathcal{S}(3|\boldsymbol{A}_{3\times 5}) = \{2, 4, 5\}$ contains an information set for $\mathcal{C}$.

In the next step, for each $m \in \mathbb{N}_2$ and for $\beta = \nu^f = 5^2$, we first generate the interleaved code array $\boldsymbol{Y}^{(m)}$ with row vectors $\boldsymbol{y}_i^{(m)} = \boldsymbol{c}_{\pi(i)}^{(m)}$, $i \in \mathbb{N}_{5^2}$, by a randomly selected permutation function $\pi(\cdot)$. Suppose that the user wishes to obtain $\boldsymbol{X}^{(1)}$. We list all downloaded sums of code symbols in **??**, which is similar to [11, Table II]. Similar to the PIR protocol in [11], Protocol 1 requires $f = 2$ rounds in each repetition, and the scheme needs to be repeated $\kappa = 3$ times. Note that since the protocol requests an equal amount of code symbols associated with $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$, it is straightforward to see that the privacy constraint is satisfied.

		Server 1	Server 2	Server 3	Server 4	Server 5
repetition 1	round 1	$y^{(1)}_{3(1-1)+1,1}$ $y^{(1)}_{3(1-1)+2,1}$ $y^{(1)}_{3(1-1)+3,1}$	$y^{(1)}_{3(1-1)+1,2}$ $y^{(1)}_{3(1-1)+2,2}$ $y^{(1)}_{3(1-1)+3,2}$	$y^{(1)}_{3(1-1)+1,3}$ $y^{(1)}_{3(1-1)+2,3}$ $y^{(1)}_{3(1-1)+3,3}$	$y^{(1)}_{3(2-1)+1,4}$ $y^{(1)}_{3(2-1)+2,4}$ $y^{(1)}_{3(2-1)+3,4}$	$y^{(1)}_{3(2-1)+1,5}$ $y^{(1)}_{3(2-1)+2,5}$ $y^{(1)}_{3(2-1)+3,5}$
		$y^{(2)}_{5\cdot0+1,1}$ $y^{(2)}_{5\cdot0+2,1}$ $y^{(2)}_{5\cdot0+5,1}$	$y^{(2)}_{5\cdot0+1,2}$ $y^{(2)}_{5\cdot0+3,2}$ $y^{(2)}_{5\cdot0+4,2}$	$y^{(2)}_{5\cdot0+1,3}$ $y^{(2)}_{5\cdot0+4,3}$ $y^{(2)}_{5\cdot0+5,3}$	$y^{(2)}_{5\cdot0+2,4}$ $y^{(2)}_{5\cdot0+3,4}$ $y^{(2)}_{5\cdot0+4,4}$	$y^{(2)}_{5\cdot0+2,5}$ $y^{(2)}_{5\cdot0+3,5}$ $y^{(2)}_{5\cdot0+5,5}$
	rnd. 2	$y^{(1)}_{3\cdot5+1,1}+y^{(2)}_{5\cdot0+3,1}$ $y^{(1)}_{4\cdot5+1,1}+y^{(2)}_{5\cdot0+4,1}$	$y^{(1)}_{3\cdot5+1,2}+y^{(2)}_{5\cdot0+2,2}$ $y^{(1)}_{4\cdot5+1,2}+y^{(2)}_{5\cdot0+5,2}$	$y^{(1)}_{3\cdot5+1,3}+y^{(2)}_{5\cdot0+2,3}$ $y^{(1)}_{4\cdot5+1,3}+y^{(2)}_{5\cdot0+3,3}$	$y^{(1)}_{3\cdot5+2,4}+y^{(2)}_{5\cdot0+1,4}$ $y^{(1)}_{4\cdot5+2,4}+y^{(2)}_{5\cdot0+5,4}$	$y^{(1)}_{3\cdot5+2,5}+y^{(2)}_{5\cdot0+1,5}$ $y^{(1)}_{4\cdot5+2,5}+y^{(2)}_{5\cdot0+4,5}$
repetition 2	round 1	$y^{(1)}_{3(2-1)+1,1}$ $y^{(1)}_{3(2-1)+2,1}$ $y^{(1)}_{3(2-1)+3,1}$	$y^{(1)}_{3(3-1)+1,2}$ $y^{(1)}_{3(3-1)+2,2}$ $y^{(1)}_{3(3-1)+3,2}$	$y^{(1)}_{3(4-1)+1,3}$ $y^{(1)}_{3(4-1)+2,3}$ $y^{(1)}_{3(4-1)+3,3}$	$y^{(1)}_{3(3-1)+1,4}$ $y^{(1)}_{3(3-1)+2,4}$ $y^{(1)}_{3(3-1)+3,4}$	$y^{(1)}_{3(3-1)+1,5}$ $y^{(1)}_{3(3-1)+2,5}$ $y^{(1)}_{3(3-1)+3,5}$
		$y^{(2)}_{5\cdot1+1,1}$ $y^{(2)}_{5\cdot1+2,1}$ $y^{(2)}_{5\cdot1+5,1}$	$y^{(2)}_{5\cdot1+1,2}$ $y^{(2)}_{5\cdot1+3,2}$ $y^{(2)}_{5\cdot1+4,2}$	$y^{(2)}_{5\cdot1+1,3}$ $y^{(2)}_{5\cdot1+4,3}$ $y^{(2)}_{5\cdot1+5,3}$	$y^{(2)}_{5\cdot1+2,4}$ $y^{(2)}_{5\cdot1+3,4}$ $y^{(2)}_{5\cdot1+4,4}$	$y^{(2)}_{5\cdot1+2,5}$ $y^{(2)}_{5\cdot1+3,5}$ $y^{(2)}_{5\cdot1+5,5}$
	rnd. 2	$y^{(1)}_{3\cdot5+2,1}+y^{(2)}_{5+3,1}$ $y^{(1)}_{4\cdot5+2,1}+y^{(2)}_{5+4,1}$	$y^{(1)}_{3\cdot5+3,2}+y^{(2)}_{5+2,2}$ $y^{(1)}_{4\cdot5+3,2}+y^{(2)}_{5+5,2}$	$y^{(1)}_{3\cdot5+4,3}+y^{(2)}_{5+2,3}$ $y^{(1)}_{4\cdot5+4,1}+y^{(2)}_{5+3,3}$	$y^{(1)}_{3\cdot5+3,1}+y^{(2)}_{5+1,4}$ $y^{(1)}_{4\cdot5+3,1}+y^{(2)}_{5+5,4}$	$y^{(1)}_{3\cdot5+3,5}+y^{(2)}_{5+1,5}$ $y^{(1)}_{4\cdot5+3,5}+y^{(2)}_{5+4,5}$
repetition 3	round 1	$y^{(1)}_{3(5-1)+1,1}$ $y^{(1)}_{3(5-1)+2,1}$ $y^{(1)}_{3(5-1)+3,1}$	$y^{(1)}_{3(4-1)+1,2}$ $y^{(1)}_{3(4-1)+2,2}$ $y^{(1)}_{3(4-1)+3,2}$	$y^{(1)}_{3(5-1)+1,3}$ $y^{(1)}_{3(5-1)+2,3}$ $y^{(1)}_{3(5-1)+3,3}$	$y^{(1)}_{3(4-1)+1,4}$ $y^{(1)}_{3(4-1)+2,4}$ $y^{(1)}_{3(4-1)+3,4}$	$y^{(1)}_{3(5-1)+1,5}$ $y^{(1)}_{3(5-1)+2,5}$ $y^{(1)}_{3(5-1)+3,5}$
		$y^{(2)}_{5\cdot2+1,1}$ $y^{(2)}_{5\cdot2+2,1}$ $y^{(2)}_{5\cdot2+5,1}$	$y^{(2)}_{5\cdot2+1,2}$ $y^{(2)}_{5\cdot2+3,2}$ $y^{(2)}_{5\cdot2+4,2}$	$y^{(2)}_{5\cdot2+1,3}$ $y^{(2)}_{5\cdot2+4,3}$ $y^{(2)}_{5\cdot2+5,3}$	$y^{(2)}_{5\cdot2+2,4}$ $y^{(2)}_{5\cdot2+3,4}$ $y^{(2)}_{5\cdot2+4,4}$	$y^{(2)}_{5\cdot2+2,5}$ $y^{(2)}_{5\cdot2+3,5}$ $y^{(2)}_{5\cdot2+5,5}$
	rnd. 2	$y^{(1)}_{3\cdot5+5,1}+y^{(2)}_{5\cdot2+3,1}$ $y^{(1)}_{4\cdot5+5,1}+y^{(2)}_{5\cdot2+4,1}$	$y^{(1)}_{3\cdot5+4,2}+y^{(2)}_{5\cdot2+2,2}$ $y^{(1)}_{4\cdot5+4,2}+y^{(2)}_{5\cdot2+5,2}$	$y^{(1)}_{3\cdot5+5,3}+y^{(2)}_{5\cdot2+2,3}$ $y^{(1)}_{4\cdot5+5,3}+y^{(2)}_{5\cdot2+3,3}$	$y^{(1)}_{3\cdot5+4,1}+y^{(2)}_{5\cdot2+1,4}$ $y^{(1)}_{4\cdot5+4,4}+y^{(2)}_{5\cdot2+5,4}$	$y^{(1)}_{3\cdot5+5,5}+y^{(2)}_{5\cdot2+1,5}$ $y^{(1)}_{4\cdot5+5,5}+y^{(2)}_{5\cdot2+4,5}$

Table III.1: Protocol 1 with a $[5, 3, 2]$ non-MDS code for $f = 2$.

It should be mentioned that here we strongly make use of the PIR interference matrices. For example, in round 2 of repetition 1 (see **??**), since the user knows $\mathcal{C}$, the code symbols $y^{(2)}_{5\cdot0+3,1}$, $y^{(2)}_{5\cdot0+2,2}$, and $y^{(2)}_{5\cdot0+2,3}$ can be obtained by knowing $\{y^{(2)}_{5\cdot0+3,2}, y^{(2)}_{5\cdot0+3,4}, y^{(2)}_{5\cdot0+3,5}\}$ and $\{y^{(2)}_{5\cdot0+2,1}, y^{(2)}_{5\cdot0+2,4}, y^{(2)}_{5\cdot0+2,5}\}$, from which the corresponding coded symbols $\{y^{(1)}_{3\cdot5+1,1}, y^{(1)}_{3\cdot5+1,2}, y^{(1)}_{3\cdot5+1,3}\}$ can be obtained by cancelling the side information. Since $\{1, 2, 3\}$ is an information set, the corresponding requested file vector of length k can also be decoded. Hence, in summary, it is sufficient to reliably decode $5^2 = 25$ different length-k requested file vectors for $m = 1$. In summary, for $f = 2$, the user downloads 3×5 undesired symbols based on (III.9) and $(3 + 2) \times 5 = 25$ desired symbols according to (III.10) and (III.11) in each repetition. Hence, the PIR achievable rate is equal to

$$\mathrm{R} = \frac{3 \cdot 25}{3 \cdot (25 + 15)} = \frac{5}{8} = \frac{1 - \frac{3}{5}}{1 - \left(\frac{3}{5}\right)^2},$$

which corresponds to the finite MDS-PIR capacity in (III.7) with $f = 2$, i.e., the $[5, 3, 2]$ non-MDS code given by (III.15) is MDS-PIR capacity-achieving.

5 Asymptotic MDS-PIR Capacity-Achieving Protocol for the Noncolluding Case

In this section, we present Protocol 2, a file-independent PIR protocol that achieves the asymptotic MDS-PIR capacity in (III.8) for the case of noncolluding nodes. We assume that the DSS uses an $[n, k]$ code $\mathcal{C}$ over $\mathrm{GF}(q)$ of rate $R^{\mathcal{C}}$ and subpacketization α. For such a code $\mathcal{C}$, the user designs the l-th, $l \in \mathbb{N}_n$, query as

$$\boldsymbol{Q}^{(l)} = \boldsymbol{U} + \boldsymbol{V}^{(l)}, \tag{III.16}$$

where $\boldsymbol{U} = (u_{i,j})$ is a $d \times \beta f$ matrix whose elements $u_{i,j}$ are chosen independently and uniformly at random from $\mathrm{GF}(q)$ and whose purpose is to make $\boldsymbol{Q}^{(l)}$ appear random and thus ensure privacy. $\boldsymbol{V}^{(l)} = \left(v_{i,j}^{(l)}\right)$ is a $d \times \beta f$ deterministic binary matrix over $\mathrm{GF}(q)$, where $v_{i,j}^{(l)} = 1$ means that the j-th symbol in node l is accessed by the i-th subquery of $\boldsymbol{Q}^{(l)}$, that allows recovery of the requested data by the user. Matrix $\boldsymbol{V}^{(l)}$ is constructed from a $d \times n$ matrix $\hat{\boldsymbol{E}}$, as explained below.

Let $\mathcal{I}_1, \ldots, \mathcal{I}_\beta$ be β information sets for $\mathcal{C}$ (which are implicitly linked to the β stripes of each file) and define $\mathcal{F}_l \triangleq \{i \in \mathbb{N}_\beta \colon l \in \mathcal{I}_i\}$ to be the set of indices of the information sets $\mathcal{I}_1, \ldots, \mathcal{I}_\beta$ containing the l-th coordinate of $\mathcal{C}$. Then $\hat{\boldsymbol{E}} = (\hat{e}_{i,l})$ is a binary matrix of size $d \times n$ that has the following structure.

C1. Each row, denoted by $\hat{\boldsymbol{e}}_i$, $i \in \mathbb{N}_d$, has Hamming weight $w_{\mathsf{H}}(\hat{\boldsymbol{e}}) = \Gamma$.

C2. Each row $\hat{\boldsymbol{e}}_i$ is an erasure pattern that is correctable by $\mathcal{C}$.

C3. Each column, denoted by $\boldsymbol{t}_l$, $l \in \mathbb{N}_n$, has weight $w_{\mathsf{H}}(\boldsymbol{t}_l) = |\mathcal{F}_l|$, i.e., the weight of the l-th column of $\hat{\boldsymbol{E}}$ is the number of times the l-th coordinate of the storage code $\mathcal{C}$ appears in the β information sets $\mathcal{I}_1, \ldots, \mathcal{I}_\beta$.

For later use, we call the vector $(w_{\mathsf{H}}(\boldsymbol{t}_1), \ldots, w_{\mathsf{H}}(\boldsymbol{t}_n))$ the *column weight profile of* $\hat{\boldsymbol{E}}$.

Matrix $\boldsymbol{V}^{(l)}$ is constructed from $\hat{\boldsymbol{E}}$ such that if $\hat{e}_{i,l} = 1$, then the i-th subquery of the l-th query, $\boldsymbol{q}_i^{(l)}$, accesses a code symbol stored in the l-th node. Additionally, $\hat{\boldsymbol{E}}$ is a matrix having strictly Γd nonzero entries, ensuring that Γd code symbols are downloaded by the protocol. We defer the intuition behind the three conditions above until later in this section. More precisely, matrix $\boldsymbol{V}^{(l)}$ is constructed from $\hat{\boldsymbol{E}}$ as follows. For $l \in \mathbb{N}_n$, $\boldsymbol{V}^{(l)}$ has the form

$$\boldsymbol{V}^{(l)} = \left(\boldsymbol{0}_{d\times(m-1)\beta} \mid \boldsymbol{\Delta}_l \mid \boldsymbol{0}_{d\times(f-m)\beta}\right),$$

where $\boldsymbol{\Delta}_l$ is the $d \times \beta$ binary matrix

$$\boldsymbol{\Delta}_l = \left(\boldsymbol{\omega}_{j_1^{(l)}}^{\mathsf{T}} \mid \boldsymbol{\omega}_{j_2^{(l)}}^{\mathsf{T}} \mid \ldots \mid \boldsymbol{\omega}_{j_d^{(l)}}^{\mathsf{T}}\right)^{\mathsf{T}}, \tag{III.17}$$

with $\boldsymbol{\omega}_i$, $i \in \mathbb{N}_\beta$, being the i-th β-dimensional unit vector, i.e., a length-β weight-1 binary vector with a single 1 at the i-th position and $\boldsymbol{\omega}_0 = \boldsymbol{0}_{1\times\beta}$. Also, given a

chosen $d \times n$ matrix $\hat{\boldsymbol{E}}$,

$$j_i^{(l)} = \begin{cases} s_i^{(l)} & \text{if } \hat{e}_{i,l} = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{III.18}$$

where $s_i^{(l)} \in \mathcal{F}_l$ and $s_i^{(l)} \neq s_{i'}^{(l)}$ for $i \neq i'$, $i, i' \in \mathbb{N}_d$. This completes the construction of the protocol.

Now, we provide the intuition behind conditions C1, C2, and C3 above.

- Condition C1 stems from the fact that the user should be able to recover Γ unique code symbols of the requested file $\boldsymbol{X}^{(m)}$ from the i-th subqueries $\boldsymbol{q}_i^{(l)}$ that are sent to the n nodes. Thus, each row of $\hat{\boldsymbol{E}}$ should have exactly Γ ones.
- For C2, consider an arbitrary row $\hat{\boldsymbol{e}}_i$ of $\hat{\boldsymbol{E}}$. The corresponding set of n subqueries $\{\boldsymbol{q}_i^{(1)}, \ldots, \boldsymbol{q}_i^{(n)}\}$ trigger a response from the n nodes of the form

  $$r_{l,i} = \begin{cases} Y_l + \phi_l & \text{if } \hat{e}_{i,l} = 1, \\ Y_l & \text{otherwise,} \end{cases}$$

  where ϕ_l represents an arbitrary code symbol present in the l-th node, and Y_l is some interference symbol generated due to the product between $\boldsymbol{q}_i^{(l)}$ and the content of the l-th node. The vector $(Y_1, \ldots, Y_n)$ represents a codeword of $\mathcal{C}$ (see also Theorem 2 below and its proof in Appendix D for further details). In order to recover ϕ_l, $l \in \chi(\hat{\boldsymbol{e}}_i)$, we need to know Y_l. This can be seen as a decoding problem over the binary erasure channel. In other words, the i-th row of $\hat{\boldsymbol{E}}$ should correspond to an erasure pattern that is correctable by $\mathcal{C}$.
- Condition C3 comes from the fact that the protocol should be able to recover $w_{\mathsf{H}}(\boldsymbol{t}_l)$ unique code symbols from the l-th node.

We remark that for a code $\mathcal{C}$, $\hat{\boldsymbol{E}}$ and $\{\mathcal{I}_i\}_{i \in \mathbb{N}_\beta}$ need not be unique. Furthermore, each set $\mathcal{I}_i$, $i \in \mathbb{N}_\beta$, can alternatively be represented as a correctable erasure pattern $\bar{\boldsymbol{e}}_i = (\bar{e}_{i,1}, \ldots, \bar{e}_{i,n})$, where $\bar{e}_{i,l} = 0$, $\forall\, l \in \mathcal{I}_i$. Also, the information sets $\{\mathcal{I}_i\}_{i \in \mathbb{N}_\beta}$ can alternatively be defined by a matrix $\bar{\boldsymbol{E}}$ of size $\beta \times n$ as

$$\bar{\boldsymbol{E}} = \begin{pmatrix} \bar{\boldsymbol{e}}_1 \\ \vdots \\ \bar{\boldsymbol{e}}_\beta \end{pmatrix}.$$

The two matrices $\hat{\boldsymbol{E}}$ and $\bar{\boldsymbol{E}}$ can be stacked into the matrix $\boldsymbol{E} = (e_{i,l})$ as

$$\boldsymbol{E} = \begin{pmatrix} \hat{\boldsymbol{E}} \\ \bar{\boldsymbol{E}} \end{pmatrix}. \tag{III.19}$$

To meet the conditions above, the only requirement is that for each $l \in \mathbb{N}_n$, $w_{\mathsf{H}}(\boldsymbol{t}_l) = \beta - w_{\mathsf{H}}(\boldsymbol{w}_l)$, where $\boldsymbol{t}_l$ and $\boldsymbol{w}_l$ are columns of $\hat{\boldsymbol{E}}$ and $\bar{\boldsymbol{E}}$, respectively. This is equivalent to finding a $(\beta + d) \times n$ β-column regular matrix $\boldsymbol{E}$ in which

each row is a correctable erasure pattern. Hence, we conclude that the requirements for $\boldsymbol{E}$ are equivalent to finding a PIR achievable rate matrix

$$\boldsymbol{\Lambda}_{d,\beta+d}(\mathcal{C}) = \mathbf{1}_{(\beta+d)\times n} - \boldsymbol{E}_{(\beta+d)\times n}. \tag{III.20}$$

In the following lemma, we prove that our construction of the queries ensures that the privacy condition (III.5a) is satisfied.

Lemma 4. *Consider a DSS that uses an $[n,k]$ code with subpacketization α to store f files, each divided into β stripes. Then, the queries $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_n$, designed as in* (III.16) *satisfy $\mathsf{H}\big(m|\boldsymbol{Q}^{(l)}\big) = \mathsf{H}(m)$, where $l \in \mathbb{N}_n$ represents the spy node.*

Proof. The queries $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_n$, are a sum of a random matrix $\boldsymbol{U}$ and a deterministic matrix $\boldsymbol{V}^{(l)}$. The resulting queries have elements that are independently and uniformly distributed at random from $\mathsf{GF}(q)$. Hence, any $\boldsymbol{Q}^{(l)}$ obtained by the spy node is statistically independent of m. This ensures that $\mathsf{H}\big(m|\boldsymbol{Q}^{(l)}\big) = \mathsf{H}(m)$. □

The following theorem shows that Protocol 2 achieves perfect information-theoretic PIR, and it gives its achievable PIR rate, $\mathsf{R}(\mathcal{C})$. Note that to prove perfect information-theoretic PIR it remains to be shown that from the responses $\boldsymbol{r}_l$ in (III.4) sent by the nodes back to the user, one can recover the requested file, i.e., that the constructed PIR protocol satisfies the recovery condition in (III.5b).

Theorem 2. *Consider a DSS that uses an $[n,k]$ code with subpacketization α to store f files, each divided into β stripes. If there exists a Γ-regular matrix $\boldsymbol{E}$ satisfying conditions* C1*,* C2*, and* C3*, then $\mathsf{H}\big(\boldsymbol{X}^{(m)}|\boldsymbol{r}_1,\ldots,\boldsymbol{r}_n\big) = 0$ and the PIR rate*

$$\mathsf{R}(\mathcal{C}) = \frac{\Gamma}{n} \leq \frac{n-k}{n}$$

is achievable.

Proof. See Appendix D. □

Theorem 2 generalizes [12, Th. 1] to any linear code.

Corollary 3. *If for an $[n,k]$ code $\mathcal{C}$ there exists an $(n-k)$-regular matrix $\boldsymbol{E}$ satisfying conditions* C1*,* C2*, and* C3*, then Protocol 2 achieves the asymptotic MDS-PIR capacity C_∞ in* (III.8)*.*

Remark 2. *From* (III.20)*, if there exists an $(n-k)$-regular matrix $\boldsymbol{E}$ satisfying conditions* C1*,* C2*, and* C3*, a $\boldsymbol{\Lambda}_{\kappa,\nu}$ MDS-PIR capacity-achieving matrix with $\frac{\kappa}{\nu} = \frac{k}{n}$ exists. Thus, if a code achieves the asymptotic MDS-PIR capacity C_∞ with Protocol 2, it also achieves the finite MDS-PIR capacity C_f with Protocol 1.*

Note that parameters Γ, β, and d (which are not explicitly mentioned in the theorem) have to be carefully selected such that a Γ-row regular matrix $\hat{\boldsymbol{E}}$ (satisfying condition C3) actually exists with a valid collection of information sets $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\beta}$. In the following corollary, we provide a valid set of values.

Corollary 4. *Let $\mathcal{C}$ be an $[n, k, d^{\mathcal{C}}_{\min}]$ code. For $\Gamma = \min\left(k, d^{\mathcal{C}}_{\min} - 1\right)$, it holds that*

$$\mathsf{H}\left(\boldsymbol{X}^{(m)} | \boldsymbol{r}_1, \ldots, \boldsymbol{r}_n\right) = 0, \tag{III.21}$$

and the PIR rate $\mathsf{R}(\mathcal{C}) = \frac{\min\left(k, d^{\mathcal{C}}_{\min}-1\right)}{n}$ is achievable.

Proof. Let $d = k$ and $\beta = \Gamma$. Then, (III.21) follows directly from Theorem 2, since we have shown in Lemma 3 that the required matrix $\boldsymbol{\Lambda}_{k,k+\Gamma}(\mathcal{C})$ exists for $\mathcal{C}$, and the existence of $\boldsymbol{E}_{(k+\Gamma)\times n}$ follows from (III.20). □

The above corollary provides a lower bound on the value of Γ for any code. In other words, it allows us to design a PIR protocol with PIR rate greater than or equal to $\frac{\min\left(k, d^{\mathcal{C}}_{\min}-1\right)}{n}$. We remark that with a better designed $\hat{\boldsymbol{E}}$, it may be possible to achieve a higher PIR rate. For systematic codes with rate $R^{\mathcal{C}} > 1/2$, a better lower bound on the maximum achievable PIR rate compared to that of Corollary 4 is given below.

Corollary 5. *Let $\mathcal{C}$ be an $[n, k]$ systematic code with $R^{\mathcal{C}} > 1/2$ and $\boldsymbol{H}^{\mathcal{C}} = (\boldsymbol{P} \mid \boldsymbol{I}_{n-k})$. Consider the $[n = k, k']$ code $\mathcal{C}'$ with parity-check matrix $\boldsymbol{H}^{\mathcal{C}'} = \boldsymbol{P}$. For $\Gamma = d^{\mathcal{C}'}_{\min} - 1$, it holds that*

$$\mathsf{H}\left(\boldsymbol{X}^{(m)} | \boldsymbol{r}_1, \ldots, \boldsymbol{r}_n\right) = 0, \tag{III.22}$$

and the PIR rate $\mathsf{R}(\mathcal{C}) = \frac{d^{\mathcal{C}'}_{\min}-1}{n}$ is achievable.

Proof. As for the proof of Corollary 4, let $d = k$ and $\beta = \Gamma$. Then, (III.22) follows directly from Theorem 2. Select k erasure patterns $\boldsymbol{e}'_i$, $i \in \mathbb{N}_k$, of length k and $w_{\mathsf{H}}\left(\boldsymbol{e}'_i\right) = d^{\mathcal{C}'}_{\min} - 1$. The patterns are all correctable by the code $\mathcal{C}'$. Thus, the erasure patterns

$$\boldsymbol{e}_i = (\boldsymbol{e}'_i, \underbrace{0, \cdots, 0}_{n-k})$$

are also correctable by $\mathcal{C}$. Choosing the information sets $\mathcal{I}_i = \mathbb{N}_k$, $i \in \mathbb{N}_\Gamma$, the required Γ-regular matrix $\boldsymbol{E}_{(\beta+d)\times n}$ can then be constructed from $\{\boldsymbol{e}_i\}_{i\in\mathbb{N}_k}$ and $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\Gamma}$. □

Observe that $R^{\mathcal{C}} > \frac{1}{2}$ implies $k > d^{\mathcal{C}}_{\min} - 1$. In [25], under the assumption that $k > d^{\mathcal{C}}_{\min} - 1$, a PIR protocol achieving a PIR rate of $\frac{d^{\mathcal{C}}_{\min}-1}{n}$ was given. Note that $d^{\mathcal{C}}_{\min} \leq d^{\mathcal{C}'}_{\min}$, and thus $\mathsf{R}(\mathcal{C}) \geq \frac{d^{\mathcal{C}}_{\min}-1}{n}$ for our construction.

Below we give two examples to elucidate Protocol 2. Example 4 illustrates the PIR protocol when the underlying code has rate $R^{\mathcal{C}} > 1/2$, with parameters $d = k$ and $\beta = \Gamma$. On the other hand, Example 5 uses an underlying code that has rate $R^{\mathcal{C}} < 1/2$, again with parameters $d = k$ and $\beta = \Gamma$.

Example 4. *Consider a DSS that uses the $[5, 3, 2]$ scalar ($\alpha = 1$) binary code $\mathcal{C}$ in Section 4.4 (with generator matrix given in (III.15)) to store a single file by dividing it into β stripes. Its parity-check matrix is given by*

$$\boldsymbol{H}^{\mathcal{C}} = (\boldsymbol{P} | \boldsymbol{I}_{n-k}) = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

To determine the value of the parameter β, we compute the minimum Hamming distance $d_{\min}^{\mathcal{C}'}$ of the $[n' = 3, k' = 1]$ code $\mathcal{C}'$ with parity-check matrix $\boldsymbol{H}^{\mathcal{C}'} = \boldsymbol{P}$. From $\boldsymbol{H}^{\mathcal{C}'}$ it follows that $d_{\min}^{\mathcal{C}'} = 3$. Hence, from Corollary 5, $\beta = 2$. Let the file to be stored be denoted by the 2×3 matrix $\boldsymbol{X} = (x_{i,j})$, where the message symbols $x_{i,j} \in \mathrm{GF}(2^\ell)$ for $\ell \in \mathbb{N}$. Then,

$$\boldsymbol{C} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,1} + x_{1,2} & x_{1,2} + x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,1} + x_{2,2} & x_{2,2} + x_{2,3} \end{pmatrix}.$$

The user wants to download the file $\boldsymbol{X}$ from the DSS and sends a query $\boldsymbol{Q}^{(l)}$, $l \in \mathbb{N}_5$, to the l-th storage node. The queries take the form shown in (III.16)*. For $l \in \mathbb{N}_5$, we construct the matrix $\boldsymbol{V}^{(l)} = \boldsymbol{\Delta}_l$ by choosing an appropriate matrix $\hat{\boldsymbol{E}}$. To do this, we carefully choose the information sets $\mathcal{I}_1 = \{1, 2, 3\}$ and $\mathcal{I}_2 = \{1, 2, 3\}$ (and hence $\boldsymbol{V}^{(4)} = \boldsymbol{V}^{(5)} = \boldsymbol{0}_{d \times \beta}$). This allows us to generate a column weight profile in $\hat{\boldsymbol{E}}$. More specifically, let $\boldsymbol{t}_l$ be the l-th column of $\hat{\boldsymbol{E}}$, $l \in \mathbb{N}_5$. We have $w_{\mathrm{H}}(\boldsymbol{t}_1) = w_{\mathrm{H}}(\boldsymbol{t}_2) = w_{\mathrm{H}}(\boldsymbol{t}_3) = 2$ and $w_{\mathrm{H}}(\boldsymbol{t}_4) = w_{\mathrm{H}}(\boldsymbol{t}_5) = 0$. A valid matrix $\hat{\boldsymbol{E}}$ is*

$$\hat{\boldsymbol{E}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

and we construct $\boldsymbol{\Delta}_1$ according to (III.17)*. Focusing on the first column of $\hat{\boldsymbol{E}}$, we can see that the first two rows have a one in the first position. Thus, we choose $j_1^{(1)} = s_1^{(1)}$, $j_2^{(1)} = s_2^{(1)}$, and $j_3^{(1)} = 0$, since $\hat{e}_{1,1} = 1$, $\hat{e}_{2,1} = 1$, and $\hat{e}_{3,1} = 0$. We take $s_1^{(1)}, s_2^{(1)} \in \mathbb{N}_2$. We arbitrarily choose $s_1^{(1)} = 1$ and $s_2^{(1)} = 2$ to get*

$$\boldsymbol{\Delta}_1 = \begin{pmatrix} \boldsymbol{\omega}_1 \\ \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Similarly, we construct

$$\boldsymbol{\Delta}_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{\Delta}_3 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

The queries $\boldsymbol{Q}^{(l)}$ are sent to the respective nodes and the responses

$$\boldsymbol{r}_1 = \begin{pmatrix} u_{1,1}x_{1,1} + u_{1,2}x_{2,1} + x_{1,1} \\ u_{2,1}x_{1,1} + u_{2,2}x_{2,1} + x_{2,1} \\ u_{3,1}x_{1,1} + u_{3,2}x_{2,1} \end{pmatrix} = \begin{pmatrix} I_1 + x_{1,1} \\ I_4 + x_{2,1} \\ I_7 \end{pmatrix},$$

$$\boldsymbol{r}_2 = \begin{pmatrix} u_{1,1}x_{1,2} + u_{1,2}x_{2,2} \\ u_{2,1}x_{1,2} + u_{2,2}x_{2,2} + x_{1,2} \\ u_{3,1}x_{1,2} + u_{3,2}x_{2,2} + x_{2,2} \end{pmatrix} = \begin{pmatrix} I_2 \\ I_5 + x_{1,2} \\ I_8 + x_{2,2} \end{pmatrix},$$

$$\boldsymbol{r}_3 = \begin{pmatrix} u_{1,1}x_{1,3} + u_{1,2}x_{2,3} + x_{2,3} \\ u_{2,1}x_{1,3} + u_{2,2}x_{2,3} \\ u_{3,1}x_{1,3} + u_{3,2}x_{2,3} + x_{1,3} \end{pmatrix} = \begin{pmatrix} I_3 + x_{2,3} \\ I_6 \\ I_9 + x_{1,3} \end{pmatrix},$$

$$\boldsymbol{r}_4 = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \\ u_{3,1} & u_{3,2} \end{pmatrix} \begin{pmatrix} x_{1,1}+x_{1,2} \\ x_{2,1}+x_{2,2} \end{pmatrix} = \begin{pmatrix} I_1+I_2 \\ I_4+I_5 \\ I_7+I_8 \end{pmatrix},$$

$$\boldsymbol{r}_5 = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \\ u_{3,1} & u_{3,2} \end{pmatrix} \begin{pmatrix} x_{1,2}+x_{1,3} \\ x_{2,2}+x_{2,3} \end{pmatrix} = \begin{pmatrix} I_2+I_3 \\ I_5+I_6 \\ I_8+I_9 \end{pmatrix},$$

where $I_i = \sum_{j=1}^{2} u_{h,j} x_{j,h'}$ *and* $i = 3(h-1)+h'$*, with* $h, h' \in \mathbb{N}_3$*, are collected by the user. Notice that each storage node sends back* $k = 3$ *symbols. The user obtains the requested file as follows. Knowing* I_2*, the user obtains* I_1 *and* I_3 *from the first components of* $\boldsymbol{r}_4$ *and* $\boldsymbol{r}_5$*. This allows the user to obtain* $x_{1,1}$ *and* $x_{2,3}$*. In a similar fashion, knowing* I_6 *the user gets* I_5 *from the second component of* $\boldsymbol{r}_5$*, then uses this to obtain* I_4 *from the second component of* $\boldsymbol{r}_4$*. This allows the user to obtain* $x_{2,1}$ *and* $x_{1,2}$*. Similarly, knowing* I_7 *allows the user to get* I_8 *from the third component of* $\boldsymbol{r}_4$*. Knowing* I_8 *allows the user to obtain* I_9 *from the third component of* $\boldsymbol{r}_5$*, which then allows to recover the symbols* $x_{2,2}$ *and* $x_{1,3}$*. In this way, the user recovers all symbols of the file and hence recovers* $\boldsymbol{X}$*. Note that* $\mathrm{R} = \frac{2\cdot 3}{5\cdot 3} = \frac{2}{5}$*, which is equal to the asymptotic MDS-PIR capacity* C_∞ *in* (III.8)*.*

Example 5. *Consider a DSS consisting of* $n = 7$ *storage nodes that store a single file* $\boldsymbol{X}$*. The DSS uses a* $[7, 3, 4]$ *scalar binary code* $\mathcal{C}$*. The parity-check matrix of the code is*

$$\boldsymbol{H}^{\mathcal{C}} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We take $\beta = \Gamma = n-k = 4$*. File* $\boldsymbol{X}$ *is of size* $\beta \times k$ *and hence consists of* βk *symbols in* $\mathrm{GF}(2^\ell)$*. Accordingly, the code array is*

$$\boldsymbol{C} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,2}+x_{1,3} & x_{1,1}+x_{1,3} & x_{1,1}+x_{1,2} & x_{1,1}+x_{1,2}+x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,2}+x_{2,3} & x_{2,1}+x_{2,3} & x_{2,1}+x_{2,2} & x_{2,1}+x_{2,2}+x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,2}+x_{3,3} & x_{3,1}+x_{3,3} & x_{3,1}+x_{3,2} & x_{3,1}+x_{3,2}+x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,2}+x_{4,3} & x_{4,1}+x_{4,3} & x_{4,1}+x_{4,2} & x_{4,1}+x_{4,2}+x_{4,3} \end{pmatrix}.$$

The queries sent to each node, each consisting of $d = k = 3$ *subqueries take the form in* (III.16)*. The aim of each subquery is to recover* Γ *code symbols using the PIR protocol. In order to do so, we construct the information sets* $\{\mathcal{I}_i\}_{i\in\mathbb{N}_4}$*. With careful consideration, we choose* $\mathcal{I}_1 = \{3, 4, 6\}$*,* $\mathcal{I}_2 = \{2, 6, 7\}$*,* $\mathcal{I}_3 = \{1, 3, 4\}$*, and* $\mathcal{I}_4 = \{1, 5, 6\}$*. The column weight profile of* $\hat{\boldsymbol{E}}$ *is* $(2, 1, 2, 2, 1, 3, 1)$*. A valid matrix* $\hat{\boldsymbol{E}}$ *is*

$$\hat{\boldsymbol{E}} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Note that each erasure pattern in $\hat{\boldsymbol{E}}$ (each row) is correctable by the code $\mathcal{C}$. As in Example 4, we map the columns of $\hat{\boldsymbol{E}}$ and $\{\mathcal{I}_i\}_{i\in\mathbb{N}_4}$ to the matrix $\boldsymbol{V}^{(l)}$ and obtain

$$\boldsymbol{\Delta}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \boldsymbol{\Delta}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$
$$\boldsymbol{\Delta}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \boldsymbol{\Delta}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$
$$\boldsymbol{\Delta}_5 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \boldsymbol{\Delta}_6 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \boldsymbol{\Delta}_7 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

As an example, we next show the reconstruction of symbols from the first pair of subqueries and subresponses. We have

$$r_{1,1} = I_1,\, r_{2,1} = I_2,\, r_{3,1} = I_3 + x_{1,3},$$
$$r_{4,1} = I_2 + I_3 + x_{1,2} + x_{1,3},\, r_{5,1} = I_1 + I_3 + x_{4,1} + x_{4,3},$$
$$r_{6,1} = I_1 + I_2 + x_{1,1} + x_{1,2},\, r_{7,1} = I_1 + I_2 + I_3,$$

where $r_{l,1}$ denotes the first subresponse from the l-th node, $I_i = \sum_{j=1}^{4} u_{h,j} x_{j,h'}$ and $i = 3(h-1) + h'$, $h, h' \in \mathbb{N}_3$. Clearly, the subresponses $r_{1,1}$, $r_{2,1}$, and $r_{7,1}$ allow the user to obtain the three interference symbols I_1, I_2, and I_3. This is solely because the first row of $\hat{\boldsymbol{E}}$ (pertaining to the first subqueries) is an erasure pattern correctable by $\mathcal{C}$. Having this knowledge, the user obtains the symbols $x_{1,3}$, $x_{1,2} + x_{1,3}$, $x_{4,1} + x_{4,3}$, and $x_{1,1} + x_{1,2}$ from the remaining subresponses. From the obtained code symbols the user can decode $x_{1,3}$, $x_{1,2}$, and $x_{1,1}$, hence obtaining the message symbols in the first row of $\boldsymbol{C}$. The code symbol $x_{4,1} + x_{4,3}$ is used to decode $x_{4,1}$, $x_{4,2}$, and $x_{4,3}$ from the code symbols that are further obtained from the third subresponse. In the same way, the remaining two subresponses allow the recovery of $\beta k = 12$ message symbols.

The PIR rate is $\mathrm{R} = \frac{4\cdot 3}{7\cdot 3} = \frac{4}{7}$, *which is equal to the asymptotic MDS-PIR capacity* C_∞ *in* (III.8).

6 MDS-PIR Capacity-Achieving Codes

For given values of n and k, whether an $[n, k]$ code is MDS-PIR capacity-achieving or not is of great interest. In this section, we provide a necessary condition for an arbitrary linear code to achieve the MDS-PIR capacities C_f and C_∞. Furthermore, we prove that certain important families of codes, namely cyclic codes, RM codes, and a class of distance-optimal reconstruction codes are MDS-PIR capacity-achieving. For Protocol 2, the MDS-PIR capacity-achieving proofs for these classes of codes assume $\beta = n - k$ and $d = k$, which are not necessary the minimum values given in (III.6). However, in the numerical results section (see Tables III.2 and III.3) we show examples for which Protocol 2 also achieves the MDS-PIR capacity for β and d in (III.6).

As shown in the previous sections, the only requirement for a code $\mathcal{C}$ to achieve capacity is that there exists an MDS-PIR capacity-achieving matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ (or a $(\Gamma = n-k)$-regular matrix $\boldsymbol{E}$ of size $(\beta+d)\times n$). In other words, the code $\mathcal{C}$ should be able to correct $\beta+d$ erasure patterns of $n-k$ erasures that satisfy the regularity condition of $\boldsymbol{E}$.

Let us first consider a fact for any information set of an $[n,k]$ code.

Proposition 4 ([34, Th. 1.6.2])**.** *If $\mathcal{I}$ is an information set of an $[n,k]$ code $\mathcal{C}$, then $\mathbb{N}_n \setminus \mathcal{I}$ is an information set for its $[n,n-k]$ dual code $\mathcal{C}^\perp$.*

Based on Proposition 4, the subsequent result follows.

Corollary 6. *The dual of an $[n,k]$ MDS-PIR capacity-achieving code is an $[n,n-k]$ MDS-PIR capacity-achieving code.*

To check the MDS-PIR capacity-achievability of a linear code, sometimes it might be easier to verify the MDS-PIR capacity-achieving condition for its dual code.

Next, we derive a useful result that gives the relation between an information set and a subcode of dimension s.

Lemma 5. *Given an $[n,k]$ code $\mathcal{C}$, for any information set $\mathcal{I}$ and an s-dimensional subcode $\mathcal{D}\subseteq\mathcal{C}$, we have*

$$\left|\mathcal{I}\cap\chi(\mathcal{D})\right| \geq s.$$

Proof. See Appendix E. □

Now, we are able to provide a necessary condition for a code to achieve the MDS-PIR capacity.

Theorem 3. *If an MDS-PIR capacity-achieving matrix exists for an $[n,k]$ code $\mathcal{C}$, then*

$$d_s^{\mathcal{C}} \geq \frac{n}{k}s, \quad \forall\, s\in\mathbb{N}_k. \tag{III.23}$$

Proof. By definition there exists a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ with $\frac{\kappa}{\nu}=\frac{k}{n}$. This means that there exist information sets $\mathcal{I}_i$, $i\in\mathbb{N}_\nu$, such that in $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\nu}$ each coordinate j of $\mathcal{C}$, $j\in\mathbb{N}_n$, appears exactly κ times. Let $\mathcal{D}$ be any subcode of dimension s of the $[n,k]$ code $\mathcal{C}$. This implies that

$$\kappa\left|\chi(\mathcal{D})\right| = \sum_{i=1}^{\nu}\left|\mathcal{I}_i\cap\chi(\mathcal{D})\right| \overset{(a)}{\geq} \nu s,$$

where (a) follows from Lemma 5. Based on the definition of $d_s^{\mathcal{C}}$, $s\in\mathbb{N}_k$, there exists a rank-s subcode $\mathcal{D}^*$ that achieves $d_s^{\mathcal{C}}$. We then have

$$d_s^{\mathcal{C}} \geq \frac{\nu}{\kappa}s = \frac{n}{k}s, \quad \forall\, s\in\mathbb{N}_k.$$

□

Based on the necessary condition, it can be shown that the code $\mathcal{C}$ in Example 2 is not MDS-PIR capacity-achieving with Protocol 1, since $d_2^{\mathcal{C}} = 3 < \frac{5}{3} \cdot 2$, i.e., it is impossible to find an MDS-PIR capacity-achieving matrix $\boldsymbol{\Lambda}_{\kappa,\nu}$ for this code.

We would like to emphasize that it seems that the necessary condition for MDS-PIR capacity-achieving matrices in Theorem 3 is also a sufficient condition. We have performed an exhaustive search for codes with parameters $k \in \mathbb{N}_n$ and $n \in \mathbb{N}_{11}$ (except for $[n,k] = [10,5]$ and $[n,k] = [11, 4 \leq k \leq 7]$) and seen that for codes that satisfy the necessary condition, there always exists an MDS-PIR capacity-achieving matrix. Therefore, we conjecture that (III.23) in Theorem 3 is an if and only if condition for the existence of an MDS-PIR capacity-achieving matrix.

Conjecture 1. *An MDS-PIR capacity-achieving matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ exists for an $[n,k]$ code $\mathcal{C}$ if and only if*

$$d_s^{\mathcal{C}} \geq \frac{n}{k} s, \quad \forall\, s \in \mathbb{N}_k.$$

In the following, we provide a sufficient condition for MDS-PIR capacity-achieving codes by using the code automorphisms of an $[n,k]$ code[27, Ch. 8].

Theorem 4. *Given an $[n,k]$ code $\mathcal{C}$, if there exist n distinct automorphisms $\pi_1, \dots, \pi_n$ of $\mathcal{C}$ such that for every code coordinate $j \in \mathbb{N}_n$, $\{\pi_1(j), \dots, \pi_n(j)\} = \mathbb{N}_n$, then the code $\mathcal{C}$ is an MDS-PIR capacity-achieving code.*

Proof. Since any $[n,k]$ code $\mathcal{C}$ contains at least one information set $\mathcal{I}$, the automorphisms $\{\pi_i\}_{i \in \mathbb{N}_n}$ guarantee that

$$\mathcal{I}_i \triangleq \{\pi_i(j) \colon j \in \mathcal{I}\}, \quad i \in \mathbb{N}_n,$$

are all information sets of $\mathcal{C}$. By assumption, for a given $j \in \mathcal{I}$, we have $\{\pi_1(j), \dots, \pi_n(j)\} = \mathbb{N}_n$. Since there are in total k coordinates in $\mathcal{I}$, every coordinate appears exactly k times in $\{\mathcal{I}_i\}_{i \in \mathbb{N}_n}$, and hence an MDS-PIR capacity-achieving matrix $\boldsymbol{\Lambda}_{k,n}(\mathcal{C})$ satisfying Definition 13 exists. □

Using their known code automorphisms and Theorem 4, it is easy to prove that the families of cyclic codes and RM codes achieve the MDS-PIR capacity.

6.1 Cyclic Codes

Corollary 7. *Cyclic codes are MDS-PIR capacity-achieving codes.*

Proof. The result follows from the fact that for an $[n,k]$ cyclic code $\mathcal{C}$ any set of k consecutive coordinates forms an information set. Hence, there exist n valid distinct automorphisms of $\mathcal{C}$ satisfying Theorem 4. □

6.2 Reed-Muller Codes

Corollary 8. *RM codes are MDS-PIR capacity-achieving codes.*

Proof. Consider an arbitrary RM code $\mathcal{R}(v,m)$ with $v \in \{0\} \cup \mathbb{N}_m$ for some $m \in \mathbb{N}$. We will show that there exists a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,v}$ for $\mathcal{R}(v,m)$ for $(\kappa,v) = (k,n)$. Let us consider the automorphisms $g(\boldsymbol{\mu}) \triangleq \boldsymbol{\mu}+\boldsymbol{\sigma}$ for all 2^m-tuples of $\boldsymbol{\sigma} \in \mathrm{GF}(2)^{m\times 1}$, and order them as $g_1, \ldots, g_n, n = 2^m$ (see Section 2.1). Since any $[n,k]$ RM code contains at least one information set $\mathcal{I}$, Proposition 2 guarantees that

$$\mathcal{I}_i \triangleq \{g_i(\boldsymbol{\mu}) \colon \boldsymbol{\mu} \in \mathcal{I}\}, \quad i \in \mathbb{N}_n,$$

are all information sets for the RM code. This implies that for a given $\boldsymbol{\mu} \in \mathcal{I}$, among $\{\mathcal{I}_i\}_{i\in\mathbb{N}_n}$, the set

$$\{\boldsymbol{\mu}+\boldsymbol{\sigma}_1, \boldsymbol{\mu}+\boldsymbol{\sigma}_2, \ldots, \boldsymbol{\mu}+\boldsymbol{\sigma}_n\}$$

forms the vector space $\mathrm{GF}(2)^{m\times 1}$. There are in total k coordinates $\boldsymbol{\mu}$ of $\mathcal{I}$, hence every m-tuple represented coordinate appears exactly k times, and we are able to find n distinct automorphisms satisfying Theorem 4. □

We remark here that because of the property of invertible and affine automorphisms for binary RM codes, it is not too hard to see that the MDS-PIR capacity-achievability of RM codes can be extended to nonbinary generalized RM codes [36]. The detailed discussion is omitted. Furthermore, note that in the independent work [28] it was also shown that RM codes can achieve the asymptotic MDS-PIR capacity, albeit with a protocol that requires a much larger β and d.

Besides cyclic codes and RM codes, there exist other families of codes satisfying Theorem 4, for instance, the class of low-density parity-check (LDPC) codes constructed from array codes [37, 38]. We further emphasize that the proof of Theorem 4 indicates that the automorphisms of an $[n,k]$ code are very important to design an MDS-PIR capacity-achieving matrix.

6.3 Local Reconstruction Codes

In this subsection, we prove that a certain family of LRCs achieve the MDS-PIR capacity by directly showing the existence of $(n-k)$-regular $n \times n$ matrix $\boldsymbol{E}$.

Consider an $[n,k]$ distance-optimal (r,δ) information locality code (see Definition 7) for which the $(n'-k)\times n'$ matrix

$$\left(\begin{array}{cccc|c} \boldsymbol{P}_1 & \boldsymbol{P}_2 & \cdots & \boldsymbol{P}_{L_\mathrm{c}} & \\ \boldsymbol{M}_1 & \boldsymbol{M}_2 & \cdots & \boldsymbol{M}_{L_\mathrm{c}} & \boldsymbol{I}_{n'-k} \end{array}\right) \triangleq \boldsymbol{H}^{\mathrm{MDS}} \tag{III.24}$$

is the parity-check matrix of an $[n',k]$ MDS code over $\mathrm{GF}(q)$, where $n' = n - (L_\mathrm{c}-1)(\delta-1)$.[4] For such a class of codes, we give an explicit construction of the matrix $\boldsymbol{E}$ in order to design the PIR protocol.

Recall that $L = \left\lfloor \frac{n}{n_\mathrm{c}} \right\rfloor$, $n_\mathrm{c} = r+\delta-1$, and let $\bar{r} \triangleq n \bmod n_\mathrm{c}$. We consider

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{E}_{1,1} & \boldsymbol{E}_{1,2} & \cdots & \boldsymbol{E}_{1,L+1} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{E}_{L+1,1} & \boldsymbol{E}_{L+1,2} & \cdots & \boldsymbol{E}_{L+1,L+1} \end{pmatrix}$$

[4] Examples of codes that satisfy (III.24) are Pyramid codes, the LRCs in [21], and codes from the parity-splitting construction of [22].

having $(L+1)^2$ submatrices $\boldsymbol{E}_{l,h}$, $l,h \in \mathbb{N}_{L+1}$. For any $l,h \in \mathbb{N}_L$, the submatrices $\boldsymbol{E}_{l,h}$ have dimensions $n_c \times n_c$, $\boldsymbol{E}_{l,L+1}$ has dimensions $n_c \times \bar{r}$, $\boldsymbol{E}_{L+1,h}$ has dimensions $\bar{r} \times n_c$, and $\boldsymbol{E}_{L+1,L+1}$ has dimensions $\bar{r} \times \bar{r}$. We denote by $\boldsymbol{e}_i^{(l)}$, $l \in \mathbb{N}_{L+1}$, the i-th row of $\left(\boldsymbol{E}_{l,1} | \ldots | \boldsymbol{E}_{l,L+1}\right)$. The coordinates of $\boldsymbol{e}_i^{(l)}$ represent the coordinates of the code $\mathcal{C}$ defined by its parity-check matrix in (III.2). Furthermore, each row vector is subdivided into $L+1$ subvectors $\boldsymbol{e}_{i,j}^{(l)}$, $j \in \mathbb{N}_{L+1}$, as

$$\boldsymbol{e}_i^{(l)} = (e_{i,1}^{(l)}, \ldots, e_{i,n}^{(l)}) = (\boldsymbol{e}_{i,1}^{(l)}, \ldots, \boldsymbol{e}_{i,L}^{(l)}, \boldsymbol{e}_{i,L+1}^{(l)}).$$

The subvectors $\boldsymbol{e}_{i,1}^{(l)}, \ldots, \boldsymbol{e}_{i,L}^{(l)}$ are of length n_c, while $\boldsymbol{e}_{i,L+1}^{(l)}$ is of length $\bar{r}$. Correspondingly, we can think about $\boldsymbol{E}$ as partitioned into $L+1$ column partitions, where the first L_c partitions correspond to the L_c local codes and the remaining $L+1-L_c$ partitions correspond to global parities (see also (III.3)). We can write $\boldsymbol{E}$ as

$$\boldsymbol{E} \triangleq \begin{pmatrix} \boldsymbol{e}_1^{(1)} \\ \vdots \\ \boldsymbol{e}_{n_c}^{(1)} \\ \vdots \\ \boldsymbol{e}_{n_c}^{(L)} \\ \boldsymbol{e}_1^{(L+1)} \\ \vdots \\ \boldsymbol{e}_{\bar{r}}^{(L+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{e}_{1,1}^{(1)} & \boldsymbol{e}_{1,2}^{(1)} & \cdots & \boldsymbol{e}_{1,L}^{(1)} & \boldsymbol{e}_{1,L+1}^{(1)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \boldsymbol{e}_{n_c,1}^{(1)} & \boldsymbol{e}_{n_c,2}^{(1)} & \cdots & \boldsymbol{e}_{n_c,L}^{(1)} & \boldsymbol{e}_{n_c,L+1}^{(1)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \boldsymbol{e}_{n_c,1}^{(L)} & \boldsymbol{e}_{n_c,2}^{(L)} & \cdots & \boldsymbol{e}_{n_c,L}^{(L)} & \boldsymbol{e}_{n_c,L+1}^{(L)} \\ \boldsymbol{e}_{1,1}^{(L+1)} & \boldsymbol{e}_{1,2}^{(L+1)} & \cdots & \boldsymbol{e}_{1,L}^{(L+1)} & \boldsymbol{e}_{1,L+1}^{(L+1)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \boldsymbol{e}_{\bar{r},1}^{(L+1)} & \boldsymbol{e}_{\bar{r},2}^{(L+1)} & \cdots & \boldsymbol{e}_{\bar{r},L}^{(L+1)} & \boldsymbol{e}_{\bar{r},L+1}^{(L+1)} \end{pmatrix}.$$

We refer to the set of rows $\boldsymbol{e}_1^{(l)}, \ldots, \boldsymbol{e}_{n_c}^{(l)}$ as the l-th row partition of $\boldsymbol{E}$.

For convenience, we divide $\boldsymbol{E}$ into four submatrices $\tilde{\boldsymbol{E}}$, $\boldsymbol{W}$, $\boldsymbol{Z}$, and $\boldsymbol{O}$ defined as

$$\tilde{\boldsymbol{E}} \triangleq \begin{pmatrix} \boldsymbol{e}_{1,1}^{(1)} & \boldsymbol{e}_{1,2}^{(1)} & \cdots & \boldsymbol{e}_{1,L}^{(1)} \\ \boldsymbol{e}_{2,1}^{(1)} & \boldsymbol{e}_{2,2}^{(1)} & \cdots & \boldsymbol{e}_{2,L}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{e}_{n_c,1}^{(L)} & \boldsymbol{e}_{n_c,2}^{(L)} & \cdots & \boldsymbol{e}_{n_c,L}^{(L)} \end{pmatrix}, \boldsymbol{Z} \triangleq \begin{pmatrix} \boldsymbol{e}_{1,L+1}^{(1)} \\ \boldsymbol{e}_{2,L+1}^{(1)} \\ \vdots \\ \boldsymbol{e}_{n_c,L+1}^{(L)} \end{pmatrix},$$

$$\boldsymbol{W} \triangleq \begin{pmatrix} \boldsymbol{e}_{1,1}^{(L+1)} & \boldsymbol{e}_{1,2}^{(L+1)} & \cdots & \boldsymbol{e}_{1,L}^{(L+1)} \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{e}_{\bar{r},1}^{(L+1)} & \boldsymbol{e}_{\bar{r},2}^{(L+1)} & \cdots & \boldsymbol{e}_{\bar{r},L}^{(L+1)} \end{pmatrix}, \boldsymbol{O} \triangleq \begin{pmatrix} \boldsymbol{e}_{1,L+1}^{(L+1)} \\ \vdots \\ \boldsymbol{e}_{\bar{r},L+1}^{(L+1)} \end{pmatrix}, \tag{III.25}$$

where $\tilde{\boldsymbol{E}}$ is an $n_c L \times n_c L$ matrix, having L^2 submatrices $\boldsymbol{E}_{l,h}$, $l,h \in \mathbb{N}_L$.

In the following, we give a systematic construction of $\boldsymbol{E}$ such that it is $(n-k)$-regular. The construction involves two steps.

a) **Initialize matrices $\tilde{\boldsymbol{E}}$, $\boldsymbol{W}$, $\boldsymbol{Z}$, and $\boldsymbol{O}$.** Matrix $\boldsymbol{Z}$ is initialized to the all-zero matrix of dimensions $n_c L \times \bar{r}$. Matrices $\boldsymbol{W}$ and $\boldsymbol{O}$ are initialized by setting $e_{i,j}^{(L+1)} = 1$, $i \in \mathbb{N}_{\bar{r}}$, $j \in \mathcal{P} = \bigcup_{j'=1}^{L+1} \mathcal{P}_{j'}$, where $\mathcal{P}$ corresponds to the parity coordinates of $\mathcal{C}$ and the sets $\mathcal{P}_{j'}$ are defined in Section 2.2 (see (III.3)). Let $m = \left\lfloor \frac{n-k}{L} \right\rfloor$, $m_1 = m+1$, $\rho_1 = \cdots = \rho_t = m_1$, and $\rho_{t+1} = \cdots = \rho_L = m$, where

$t = (n-k) \bmod L$. Matrix $\tilde{\boldsymbol{E}}$ is initialized with the structure

$$\tilde{\boldsymbol{E}} = \begin{pmatrix} \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \cdots & \boldsymbol{\pi}_L \\ \boldsymbol{\pi}_L & \boldsymbol{\pi}_1 & \cdots & \boldsymbol{\pi}_{L-1} \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{\pi}_2 & \boldsymbol{\pi}_3 & \cdots & \boldsymbol{\pi}_1 \end{pmatrix}, \tag{III.26}$$

where each matrix entry $\boldsymbol{\pi}_l$, $l \in \mathbb{N}_L$, is a ρ_l-regular square matrix of order n_c. Notice that due to the structure in (III.26), $\tilde{\boldsymbol{E}}$ has row and column weight equal to $n-k$, and subsequently each row of $\boldsymbol{E}$ has weight $n-k$. Note also that the columns of $\boldsymbol{E}$ with coordinates in $\mathcal{P}_j$, $j \in \mathbb{N}_L$, have column weight $n-k+\bar{r}$, while the columns with coordinates in $\mathcal{P}_{L+1}$ have weight $\bar{r}$.

b) **Swapping elements between $\tilde{\boldsymbol{E}}$ and $\boldsymbol{Z}$.** In the i-th row partition, we consider a set of row coordinates $\mathcal{R}^{(i)}$ from which $s_j^{(i)} \in \{0\} \cup \mathbb{N}$ ones from columns of the j-th column partition with coordinates in $\mathcal{P}_j$, $j \in \mathbb{N}_L$, are swapped with zeroes in the corresponding rows of $\boldsymbol{Z}$. Furthermore, the condition $\sum_{j=1}^{L} s_j^{(i)} \leq \bar{r}$ must be satisfied. For convenience, we define $\boldsymbol{s}^{(i)} = (s_1^{(i)}, \ldots, s_L^{(i)})$. The swapping of elements is performed iteratively with $\bar{r}$ iterations. We describe the procedure for iteration $j' \in \mathbb{N}_{\bar{r}}$. For the first row partititon, consider that $\boldsymbol{s}^{(1)}$ with $s_j^{(1)} = 1$ and $s_z^{(1)} = 0$, $\forall\, z \in \mathbb{N}_L \backslash \{j\}$, for some $j \in \mathbb{N}_L$, such that the resulting erasure patterns (see description below) are correctable by $\mathcal{C}$, exists. In other words, for all $i' \in \mathcal{R}^{(1)}$ and $p \in \mathcal{P}_j$ (where index j is such that $s_j^{(i)} = 1$) the one at coordinate (i', p) of $\tilde{\boldsymbol{E}}$ is swapped with a zero at coordinate (i', j') of $\boldsymbol{Z}$ (this corresponds to coordinate $(i', n_c L + j')$ of $\boldsymbol{E}$). Then, for the remaining row partitions $i = 2, \ldots, L$, consider $\boldsymbol{s}^{(i)}$ to be the $(i-1)$-th right cyclic shift of $\boldsymbol{s}^{(1)}$ and repeat the swapping procedure for the first row partition, i.e., swap the one at coordinate (i', p) with the zero at coordinate (i', j') for all $i' \in \mathcal{R}^{(i)}$ and $p \in \mathcal{P}_j$. Note that we have performed $\sum_{j=1}^{L} |P_j| = n-k-\bar{r}$ swaps from the columns of $\tilde{\boldsymbol{E}}$ with coordinates in the set $\cup_{j=1}^{L} \mathcal{P}_j$ to the j'-th column of $\boldsymbol{Z}$. Thus, each column in $\cup_{j=1}^{L} \mathcal{P}_j$ has column weight $n-k+\bar{r}-1$ and the $(n_c L + j')$-th column has column weight $n-k-\bar{r}+\bar{r} = n-k$. Letting $j' = j'+1$ and repeating the above procedure $\bar{r}$ times ensures $\boldsymbol{E}$ to be $(n-k)$-regular.

This completes the construction of $\boldsymbol{E}$, which has row and column weight $n-k$. In the following theorem, we show that each row of $\boldsymbol{E}$ (considered as an erasure pattern) can be corrected by any code from the class of distance-optimal (r, δ) information locality codes whose parity-check matrices are as in (III.2) and are compliant with (III.24). Thus, this class of codes is MDS-PIR capacity-achieving.

Theorem 5. *An $[n, k]$ distance-optimal (r, δ) information locality code $\mathcal{C}$ with parity-check matrix as in* (III.2) *and satisfying* (III.24) *is an MDS-PIR capacity-achieving code.*

Proof. See Appendix F. □

In the following, we present an example to illustrate the construction of the matrix $\boldsymbol{E}$. The existence of such a matrix ensures that the PIR protocols presented in Sections 4 and 5 achieve the finite MDS-PIR capacity C_f in (III.7) and the asymptotic MDS-PIR capacity C_∞ in (III.8), respectively.

Example 6. *Consider an* $[n = 7, k = 4]$ *Pyramid code* $\mathcal{C}$ *that is constructed from an* $[n' = 6, 4]$ *RS code over* $\mathrm{GF}(2^3)$ *with parity-check matrices*

$$\boldsymbol{H}^{\mathcal{C}} = \begin{pmatrix} z^3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z^3 & z & 1 & 0 \\ z^4 & 1 & 0 & z^5 & z^5 & 0 & 1 \end{pmatrix}$$

and

$$\boldsymbol{H}^{\mathrm{MDS}} = \begin{pmatrix} z^3 & 1 & z^3 & z & 1 & 0 \\ z^4 & 1 & z^5 & z^5 & 0 & 1 \end{pmatrix},$$

respectively, where z denotes a primitive element of $\mathrm{GF}(2^3)$. *It is easy to see that* $\mathcal{C}$ *is a distance-optimal* $(r = 2, \delta = 2)$ *information locality code. We have* $n_{\mathrm{c}} = 3$, $L = 2$, *and* $\bar{r} \triangleq n \bmod n_{\mathrm{c}} = 1$. *Since* $\rho_1 = 2$ *and* $\rho_2 = 1$, *we get*

$$\tilde{\boldsymbol{E}} = \begin{pmatrix} \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 \\ \boldsymbol{\pi}_2 & \boldsymbol{\pi}_1 \end{pmatrix} = \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right), \quad \boldsymbol{Z} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

where $\boldsymbol{\pi}_1$ *is a* 2*-regular* 3×3 *matrix and* $\boldsymbol{\pi}_2$ *is picked as the identity matrix. The set of parity coordinates is* $\mathcal{P} = \{3, 6, 7\}$, *and we set* $e_{1,3}^{(3)} = e_{1,6}^{(3)} = e_{1,7}^{(3)} = 1$. *As such, we get*

$$\boldsymbol{W} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{O} = \begin{pmatrix} 1 \end{pmatrix}.$$

This completes Step a) of the construction above. Note that each row of $\boldsymbol{E}$ *has now weight* 3. *The second step of the procedure (Step b)) is as follows. Consider the first iteration,* $j' = 1$. *In the first row partition we choose* $\boldsymbol{s}^{(1)} = (s_1^{(1)} = 1, s_2^{(1)} = 0)$. *Furthermore,* $s_1^{(1)} + s_2^{(1)} = \bar{r} = 1$. *Taking* $\mathcal{R}^{(1)} = \{2\}$, *we do the swap between the coordinates* $(i' = 2, p = 3 \in \mathcal{P}_1)$ *and* (i', j'). *For the second row partition we have* $\boldsymbol{s}^{(2)} = (0, 1)$ *which is a right cyclic shift of* $\boldsymbol{s}^{(1)}$. *Taking* $\mathcal{R}^{(2)} = \{6\}$, *we do the swap between the coordinates* $(i' = 6, p = 6 \in \mathcal{P}_2)$ *and* (i', j'). *Thus, we have*

$$e_{2,3}^{(1)} = 0,\ e_{2,7}^{(1)} = 1,$$
$$e_{3,6}^{(2)} = 0,\ e_{3,7}^{(2)} = 1.$$

Algorithm 1: Optimizing the PIR rate

Input: Distributed storage code $\mathcal{C}$ of length n
Output: Optimized matrix $\boldsymbol{E}_{\text{opt}}$ and largest possible Γ

$\Gamma \leftarrow \min\left(k, d_{\min}^{\mathcal{C}} - 1\right)$
$\boldsymbol{E}_{\text{opt}} \leftarrow \emptyset$, $\Gamma_{\text{opt}} \leftarrow \Gamma$
$\mathcal{L}_{n-k} \leftarrow$ `ComputeErasurePatternList`$(\mathcal{C}, n-k)$
while $\Gamma \leq n-k$ **do**
  $\mathcal{L}_\Gamma \leftarrow$ `ComputeErasurePatternList`$(\mathcal{C}, \Gamma)$
  if $\mathcal{L}_\Gamma \neq \emptyset$ **then**
    $\boldsymbol{E} \leftarrow$ `ComputeMatrix`$(\mathcal{L}_\Gamma, \mathcal{L}_{n-k})$
    if $\boldsymbol{E} \neq \emptyset$ **then**
      $\boldsymbol{E}_{\text{opt}} \leftarrow \boldsymbol{E}$, $\Gamma_{\text{opt}} \leftarrow \Gamma$
    else
      return $(\boldsymbol{E}_{\text{opt}}, \Gamma_{\text{opt}})$
    end
  end
  $\Gamma \leftarrow \Gamma + 1$
end
return $(\boldsymbol{E}_{\text{opt}}, \Gamma_{\text{opt}})$

Since $\bar{r} = 1$, this completes Step b), which results in

$$\boldsymbol{E} = \left(\begin{array}{ccc|ccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right).$$

The entries in red indicate the swapped values within each row. It can easily be verified that each row of $\boldsymbol{E}$ is an erasure pattern that is correctable by code $\mathcal{C}$.

7 Optimizing the PIR Rate for the Noncolluding Case

For codes for which we are not able to prove that they achieve the MDS-PIR capacity, in this section we provide an algorithm to optimize Protocols 1 and 2 in order to achieve the highest possible PIR rate R for a given code by taking the structure of the underlying code into consideration. The algorithm is given in Algorithm 1 and is based on Theorem 2. In particular, we need to find a $(d+\beta) \times n$ matrix $\boldsymbol{E}$ (III.19) in which $\hat{\boldsymbol{E}}$ consists of erasure patterns of weight Γ that are all correctable by $\mathcal{C}$, and in which $\bar{\boldsymbol{E}}$ corresponds to information sets of $\mathcal{C}$ (the support of each row is the complement of an information set). In addition, it is required that each column weight of $\hat{\boldsymbol{E}}$ is equal to the corresponding column weight of $\mathbf{1} - \bar{\boldsymbol{E}}$ (see Section 5). Note that from the resulting matrix $\boldsymbol{E}$ we can find a PIR achievable rate matrix by taking its binary complement as in (III.20) (see Section 5), thus optimizing Protocols 1 and 2 in Sections 4 and 5, respectively.

The main issues that need to be addressed are the efficient enumeration of the set of erasure patterns of a given weight Γ ($\hat{\boldsymbol{E}}$) and also of weight $n-k$ ($\bar{\boldsymbol{E}}$, corresponding to information sets) that can be corrected by $\mathcal{C}$, and the efficient computation of the matrix $\boldsymbol{E}$. These issues are addressed by the subprocedures `ComputeErasurePatternList`$(\mathcal{C},\cdot)$ and `ComputeMatrix`$(\mathcal{L}_\Gamma,\mathcal{L}_{n-k})$, in Lines 3, 5 and 7 of Algorithm 1, and discussed below in Sections 7.1 and 7.2, respectively. Here, $\mathcal{L}_\Gamma$ and $\mathcal{L}_{n-k}$ correspond to erasure patterns for $\hat{\boldsymbol{E}}$ and $\bar{\boldsymbol{E}}$, respectively. We remark that the algorithm will always return a valid $\boldsymbol{E} \neq \emptyset$, since initially $\Gamma = \min\left(k, d^{\mathcal{C}}_{\min} - 1\right)$. This follows directly from the fact that we can construct an arbitrary $\hat{\boldsymbol{E}}$ with row weights Γ such that its column weights match the corresponding weights in $\mathbf{1} - \bar{\boldsymbol{E}}$. Each row in $\hat{\boldsymbol{E}}$ is an erasure pattern that is correctable by $\mathcal{C}$.

Let $d = k$ and $\beta = n - k$. In the particular case of $\mathcal{C}$ being a rate $R^{\mathcal{C}} > 1/2$ systematic MDS code, $d^{\mathcal{C}}_{\min} = n - k + 1$, and the algorithm will do exactly one iteration of the main loop. This follows directly from the construction of $\boldsymbol{E}$: the matrix $\boldsymbol{E}$ can be constructed by taking the support of an arbitrary information set of $\mathcal{C}$ and cyclically shifting it n times to construct an $n \times n$ PIR achievable rate matrix, after which the resulting matrix is complemented as (III.20) to get $\boldsymbol{E}$. In this case, the overall PIR scheme reduces to the one described in [12, Sec. IV] for systematic MDS codes of rate $R^{\mathcal{C}} > 1/2$. Clearly, for general MDS codes (including nonsystematic codes) of rate $R^{\mathcal{C}} > 1/2$, the same construction of $\boldsymbol{E}$ works, and the algorithm will perform exactly one iteration of the main loop also for nonsystematic MDS codes. In the case of $\mathcal{C}$ being a rate $R^{\mathcal{C}} \leq 1/2$ general MDS code, the initial value of Γ becomes k (since $\Gamma = \min\left(k, d^{\mathcal{C}}_{\min} - 1\right) = \min(k, n-k) = k$), but the algorithm will also find a valid matrix $\boldsymbol{E}$ for $\Gamma = n - k \geq k$. Again, the existence of $\boldsymbol{E}$ follows from the same argument of cyclically shifting an existing information set n times. In the general case of $d \neq k$ and $\beta \neq n - k$, a similar argument to the one above can be made.

7.1 `ComputeErasurePatternList()`

Computing a list of erasure patterns that are correctable for a given short code can be done using any maximum likelihood (ML) decoding algorithm. For small codes, all length-n binary vectors of weight Γ (or $n-k$) correspond to erasure patterns that are correctable can be found by exhaustive search, while for longer codes a random search can be performed, in the sense of picking length-n binary vectors of weight Γ (or $n-k$) at random, and then verifying whether the corresponding erasure patterns are correctable or not. Alternatively, one can apply a random permutation π to the columns of $\boldsymbol{H}^{\mathcal{C}}$, apply the Gauss-Jordan algorithm to the resulting matrix to transform it into *row echelon form*, collect a subset of size Γ of the column indices of *leading-one-columns*,[5] and finally apply the inverse permutation π^{-1} to this subset of column indices. The resulting set corresponds to erased coordinates in $\mathcal{C}$ that can be recovered by the code. Finally, one can check whether all cyclic shifts of the added erasure pattern are correctable or not

[5]The leading-one-columns are the columns containing a *leading one*, where the first nonzero entry in each matrix row of a matrix in row echelon form is called a leading one.

and add the correctable cyclic shifts to $\mathcal{L}_\Gamma$ (or $\mathcal{L}_{n-k}$).

7.2 `ComputeMatrix()`

Given the lists $\mathcal{L}_\Gamma$ and $\mathcal{L}_{n-k}$ of erasure patterns of weight Γ and $n-k$, respectively, that are correctable for $\mathcal{C}$, we construct a $\left(|\mathcal{L}_\Gamma| + |\mathcal{L}_{n-k}|\right) \times n$ matrix, denoted by $\boldsymbol{\Psi} = (\psi_{i,j})$, in which each row $i \in \mathbb{N}_{|\mathcal{L}_\Gamma|}$ is one of the erasure patterns from $\mathcal{L}_\Gamma$ and each row $i \in \mathbb{N}_{|\mathcal{L}_\Gamma|+1:|\mathcal{L}_\Gamma|+|\mathcal{L}_{n-k}|}$ is one of the erasure patterns from $\mathcal{L}_{n-k}$. The problem is now to find a $d \times n$ submatrix $\hat{\boldsymbol{\Psi}}$ of the upper part of $\boldsymbol{\Psi}$ (rows 1 to $|\mathcal{L}_\Gamma|$) and a $\beta \times n$ submatrix $\bar{\boldsymbol{\Psi}}$ of the lower part of $\boldsymbol{\Psi}$ (rows $|\mathcal{L}_\Gamma| + 1$ to $|\mathcal{L}_\Gamma| + |\mathcal{L}_{n-k}|$) such that the column weight of each of the n columns is the same for $\hat{\boldsymbol{\Psi}}$ and the binary complement of $\bar{\boldsymbol{\Psi}}$. This can be formulated as an integer program (in the integer variables $\eta_1, \ldots, \eta_{|\mathcal{L}_\Gamma|+|\mathcal{L}_{n-k}|}$) in the following way,

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{|\mathcal{L}_\Gamma|+|\mathcal{L}_{n-k}|} \eta_i \\ \text{s. t.} \quad & \sum_{i=1}^{|\mathcal{L}_\Gamma|} \eta_i \psi_{i,j} = \sum_{i=|\mathcal{L}_\Gamma|+1}^{|\mathcal{L}_\Gamma|+|\mathcal{L}_{n-k}|} \eta_i (1-\psi_{i,j}), \ \forall j \in \mathbb{N}_n, \\ & \eta_i \in \{0,1\}, \ \forall i \in \mathbb{N}_{|\mathcal{L}_\Gamma|+|\mathcal{L}_{n-k}|}, \\ & \sum_{i=1}^{|\mathcal{L}_\Gamma|} \eta_i = d, \text{ and } \sum_{i=|\mathcal{L}_\Gamma|+1}^{|\mathcal{L}_\Gamma|+|\mathcal{L}_{n-k}|} \eta_i = \beta. \end{aligned} \tag{III.27}$$

A valid $(d+\beta)\times n$ matrix $\boldsymbol{E}$ is constructed from the rows of $\boldsymbol{\Psi}$ with η_i-values equal to one in any feasible solution of (III.27). When $|\mathcal{L}_\Gamma| + |\mathcal{L}_{n-k}|$ is large, solving (III.27) may become impractical (solving a general integer program is known to be NP-hard), in which case one can take several random subsets (of some size) of the lists $\mathcal{L}_\Gamma$ and $\mathcal{L}_{n-k}$, construct the corresponding matrices $\boldsymbol{\Psi}$, and try to solve the program in (III.27).

8 Multiple Colluding Nodes

In this section, we consider the scenario where $T > 1$ nodes act as spies and have the ability to collude. In particular, we propose a protocol for this scenario that improves upon the PIR protocol in [25]. We refer to the protocol in [25] as the $(\mathcal{C}, \bar{\mathcal{C}})$-*retrieval protocol (or scheme)*, since it is based on two linear codes: an $[n,k]$ code $\mathcal{C}$ and an $[n,\bar{k}]$ code $\bar{\mathcal{C}}$, where $\mathcal{C}$ is the underlying storage code of the DSS and $\bar{\mathcal{C}}$ defines the queries. Furthermore, the retrieval process is defined by an $[n,\tilde{k}]$ code $\tilde{\mathcal{C}}$ that is the Hadamard product of $\mathcal{C}$ and $\bar{\mathcal{C}}$, $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$. The protocol yields privacy against at most $T = d_{\mathsf{min}}^{\bar{\mathcal{C}}^\perp} - 1$ colluding nodes under the assumption that the code $\bar{\mathcal{C}}$ with $d_{\mathsf{min}}^{\bar{\mathcal{C}}^\perp} = T+1$ exists for the given T (existing in the sense that the Hadamard product of $\mathcal{C}$ and $\bar{\mathcal{C}}$ has rate strictly smaller than 1).

Originally, the protocol was designed to work with GRS codes, a class of MDS codes, i.e., both codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ are GRS codes. In this case $\bar{\mathcal{C}}$ has parameters $[n, \bar{k} = T]$, the retrieval code $\tilde{\mathcal{C}}$ has parameters $[n, \tilde{k} = k + T - 1]$, and the PIR rate is

$$\mathrm{R}_{\mathrm{GRS}} = \frac{n - (k + T - 1)}{n}.$$

For non-MDS codes, the protocol achieves a PIR rate

$$\mathrm{R}(\mathcal{C}, \bar{\mathcal{C}}) = \frac{d_{\min}^{\tilde{\mathcal{C}}} - 1}{n},$$

which is lower than $\mathrm{R}_{\mathrm{GRS}}$. In general, when the underlying codes are arbitrary codes, it can be shown that the PIR rate of the $(\mathcal{C}, \bar{\mathcal{C}})$-retrieval protocol is upper-bounded by

$$\mathrm{R}_{\mathrm{UB}} \triangleq \frac{n - \tilde{k}}{n}. \tag{III.28}$$

In particular, the $(\mathcal{C}, \bar{\mathcal{C}})$-retrieval protocol in [25] achieves a PIR rate $\mathrm{R}(\mathcal{C}, \bar{\mathcal{C}}) < \mathrm{R}_{\mathrm{UB}}$ for non-MDS codes. Furthermore, it was shown in [26] that if $\mathcal{C}$ is either a GRS code or an RM code, then $\bar{\mathcal{C}}$ always exists for any $T \leq n-k$. In this section, we look at this protocol from the perspective of arbitrary linear codes $\mathcal{C}$ and propose an improved protocol, referred to as Protocol 3, that achieves a higher PIR rate R_{P3}, where $\mathrm{R}(\mathcal{C}, \bar{\mathcal{C}}) \leq \mathrm{R}_{\mathrm{P3}} \leq \mathrm{R}_{\mathrm{UB}} \leq \mathrm{R}_{\mathrm{GRS}}$. In particular, we show that the upper bound R_{UB} can be achieved for some non-MDS codes. Also, for a given T we present a code family for $\mathcal{C}$ for which $\bar{\mathcal{C}}$ exists.

8.1 Protocol 3: The Multiple Colluding Case

The protocol presented here, referred to as Protocol 3, can be seen as an extension of Protocol 2 in Section 5. We assume that each file $\boldsymbol{X}^{(m)} = \left(x_{i,j}^{(m)}\right)$, $m \in \mathbb{N}_f$, of size $\beta \times k$ is stored using an $[n, k]$ code $\mathcal{C}$ over $\mathrm{GF}(q)$, where $x_{i,j}^{(m)} \in \mathrm{GF}(q^\ell)$ for some $\ell \in \mathbb{N}$. Let $\bar{\mathcal{C}}$ be an $[n, \bar{k}]$ code over $\mathrm{GF}(q)$. The code $\bar{\mathcal{C}}$ is used to design the query matrix $\boldsymbol{Q}^{(l)}$, of dimensions $d \times \beta f$, where $\boldsymbol{q}_i^{(l)}$ is the i-th subquery of $\boldsymbol{Q}^{(l)}$ (see Section 3.1). Furthermore, $\bar{\mathcal{C}}$ characterizes T, i.e., the maximum number of colluding nodes the PIR protocol can handle whilst maintaining information-theoretic privacy. As for Protocol 2, β and d are taken as small as possible according to (III.6). The response vector corresponding to the i-th subquery $\boldsymbol{q}_i^{(l)}$, denoted by $\boldsymbol{\rho}^{(i)} = (r_{1,i}, \ldots, r_{n,i})^\mathsf{T}$, is a collection of the n response symbols $r_{l,i}$ from the n storage nodes and is related to the codewords of an $[n, \tilde{k}]$ code $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$. Furthermore, $\tilde{\mathcal{C}}$ characterizes the PIR rate of the protocol.

Query Construction

The protocol requires that the user constructs queries by choosing βf codewords $\bar{\boldsymbol{c}}_i^{(m)} = (\bar{c}_{i,1}^{(m)}, \ldots, \bar{c}_{i,n}^{(m)})$, $i \in \mathbb{N}_\beta$ and $m \in \mathbb{N}_f$, drawn independently and uniformly

at random from the code $\bar{\mathcal{C}}$. It then constructs the vector

$$\mathring{\boldsymbol{c}}_l = (\mathring{\boldsymbol{c}}_l^{(1)}, \dots, \mathring{\boldsymbol{c}}_l^{(f)}), \quad l \in \mathbb{N}_n,$$

where $\mathring{\boldsymbol{c}}_l^{(m)} = (\bar{c}_{1,l}^{(m)}, \dots, \bar{c}_{\beta,l}^{(m)})$. Thus, the vector $\mathring{\boldsymbol{c}}_l$ is of length βf. The vector $\mathring{\boldsymbol{c}}_l^{(m)}$ is a collection of the entries of the l-th coordinates of the codewords $\bar{\boldsymbol{c}}_1^{(m)}, \dots, \bar{\boldsymbol{c}}_\beta^{(m)}$ that pertain to the m-th file. We denote by $\mathcal{J}_i \subseteq \mathbb{N}_n$, $i \in \mathbb{N}_d$, $|\mathcal{J}_i| = \Gamma$, the set of nodes from which the protocol obtains code symbols pertaining to the m-th file from the i-th subresponses.

Similar to Protocol 2 presented in Section 5 for the case of noncolluding nodes, we need to construct a matrix $\hat{\boldsymbol{E}}$ and β information sets $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\beta}$. The matrix $\hat{\boldsymbol{E}}$ is a $d \times n$ binary matrix where each row represents an erasure pattern of weight Γ correctable by $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$. The column weight profile of $\hat{\boldsymbol{E}}$ is determined from $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\beta}$ as in Section 5. Note that $\mathcal{J}_i$ is the support of the i-th row vector of $\hat{\boldsymbol{E}}$. Let m denote the index of the requested file. Then, the i-th subquery to node l is constructed as

$$\boldsymbol{q}_i^{(l)} = \mathring{\boldsymbol{c}}_l + \boldsymbol{\delta}_i^{(l)}, \tag{III.29}$$

where

$$\boldsymbol{\delta}_i^{(l)} = \begin{cases} \boldsymbol{\omega}_{\beta(m-1)+s_i^{(l)}} & \text{if } l \in \mathcal{J}_i, \\ \boldsymbol{\omega}_0 & \text{otherwise,} \end{cases} \tag{III.30}$$

for $l \in \mathbb{N}_n$, where $\boldsymbol{\omega}_t$, $t \in \mathbb{N}_{\beta f}$, is the t-th (βf)-dimensional unit vector and $\boldsymbol{\omega}_0 = \mathbf{0}_{1\times\beta f}$. The index $s_i^{(l)}$ is defined as

$$s_i^{(l)} \in \mathcal{F}_l = \{t \in \mathbb{N}_\beta \colon l \in \mathcal{I}_t\} \tag{III.31}$$

and $s_i^{(l)} \neq s_{i'}^{(l)}$ for $i \neq i'$, $i, i' \in \mathbb{N}_d$. The index $s_i^{(l)}$ denotes the symbol downloaded from the $s_i^{(l)}$-th row of the chunk pertaining to $\boldsymbol{X}^{(m)}$ of the l-th node in response to the i-th subquery. Clearly, we see that the symbols downloaded from all nodes form β information sets as $\sum_{i=1}^{d} |\mathcal{J}_i| = \sum_{i=1}^{\beta} |\mathcal{I}_i| = \beta k$.

Note that in (III.29), the vector $\mathring{\boldsymbol{c}}_l$ introduces randomness such that privacy is ensured, while the vector $\boldsymbol{\omega}$ is deterministic and is properly designed such that the requested file can be recovered by the user.

Response Construction

For the i-th subquery, the response symbol from the l-th node is constructed as

$$r_{l,i} = \langle \boldsymbol{q}_i^{(l)}, (c_{1,l}^{(1)}, \dots, c_{\beta,l}^{(f)}) \rangle. \tag{III.32}$$

The response symbol in (III.32) is the dot product between the subquery vector to the l-th node and the content of that node. The user obtains a response vector

$\boldsymbol{\rho}^{(i)}$, consisting of response symbols from n nodes as

$$\boldsymbol{\rho}^{(i)} = \begin{pmatrix} r_{1,i} \\ r_{2,i} \\ \vdots \\ r_{n,i} \end{pmatrix} = \underbrace{\sum_{i'=1}^{\beta} \sum_{m'=1}^{f} \begin{pmatrix} \tilde{c}_{i',1}^{(m')} c_{i',1}^{(m')} \\ \tilde{c}_{i',2}^{(m')} c_{i',2}^{(m')} \\ \vdots \\ \tilde{c}_{i',n}^{(m')} c_{i',n}^{(m')} \end{pmatrix}}_{\in \left\{ \boldsymbol{x} \in (\mathrm{GF}(q^\ell))^n \colon \boldsymbol{H}^{\tilde{\mathcal{C}}} \boldsymbol{x} = \boldsymbol{0} \right\}} + \begin{pmatrix} o_1^{(i)} \\ o_2^{(i)} \\ \vdots \\ o_n^{(i)} \end{pmatrix}, \tag{III.33}$$

where $\boldsymbol{H}^{\tilde{\mathcal{C}}}$ is the parity-check matrix of the code $\tilde{\mathcal{C}}$,[6] the symbol $o_l^{(i)}$ denotes the symbol obtained from the l-th node corresponding to the i-th subquery, and

$$o_l^{(i)} = \begin{cases} c_{i',l}^{(m)} & \text{if } l \in \mathcal{J}_i, \\ 0 & \text{otherwise,} \end{cases}$$

where $i' = s_i^{(l)}$. These symbols are obtained by post-processing (III.33) as follows,

$$\boldsymbol{H}^{\tilde{\mathcal{C}}} \boldsymbol{\rho}^{(i)} = \boldsymbol{H}^{\tilde{\mathcal{C}}} \begin{pmatrix} o_1^{(i)} \\ o_2^{(i)} \\ \vdots \\ o_n^{(i)} \end{pmatrix}. \tag{III.34}$$

This completes the construction of the PIR protocol. In the following, we prove that this protocol satisfies the PIR conditions (III.5a) and (III.5b) in Definition 8.

Lemma 6. *Consider a DSS that uses an $[n, k]$ code with subpacketization α to store f files, each divided into β stripes, and assume the privacy model of Section 3.1 with a set $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\} \subset \mathbb{N}_n$ of $|\mathcal{T}| \leq T \leq d_{\mathsf{min}}^{\tilde{\mathcal{C}}^\perp} - 1$ colluding nodes. Then, the subqueries $\boldsymbol{q}_i^{(l)}$, $l \in \mathbb{N}_n$, $i \in \mathbb{N}_d$, designed as in* (III.30) *satisfy* $\mathsf{H}\big(m | \boldsymbol{Q}^{(t_1)}, \ldots, \boldsymbol{Q}^{(t_{|\mathcal{T}|})}\big) = \mathsf{H}(m)$.

Proof. The addition of a deterministic vector in(III.30) does not change the probability distribution of the vectors $\boldsymbol{q}_i^{(t_1)}, \ldots, \boldsymbol{q}_i^{(t_{|\mathcal{T}|})}$. The same can be said about their joint distribution. Furthermore, in each query matrix $\boldsymbol{Q}^{(l)}$, $l \in \{t_1, \ldots, t_{|\mathcal{T}|}\}$, the subqueries $\boldsymbol{q}_i^{(l)}$ are independent of each other. Thus, the proof follows the same lines as the proof of [25, Th. 8]. □

Theorem 6. *Consider a DSS that uses an $[n, k]$ code $\mathcal{C}$ with subpacketization α to store f files, each divided into β stripes. Let $\bar{\mathcal{C}}$ be an $[n, \bar{k}]$ code such that there exists an $[n, \tilde{k}]$ code $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ of rate $R^{\tilde{\mathcal{C}}} < 1$. If there exists a Γ-row regular $d \times n$ binary matrix $\hat{\boldsymbol{E}}$ in which each row is a correctable erasure pattern for $\tilde{\mathcal{C}}$ and satisfying condition* C3*, then $\mathsf{H}(\boldsymbol{X}^{(m)} | \boldsymbol{\rho}^{(1)}, \ldots, \boldsymbol{\rho}^{(d)}) = 0$ and the PIR rate*

$$\mathsf{R}_{\mathsf{P3}} = \frac{\Gamma}{n} \leq \mathsf{R}_{\mathsf{UB}} \tag{III.35}$$

is achievable.

[6] Note that the upload cost of the PIR scheme in [25, 26] grows linearly with f. However, the requested upload cost can actually be ignored by extending the field size of the stored files.

Proof. By assumption there exists a matrix $\hat{\boldsymbol{E}}$ of size $d \times n$ having row weight Γ. Furthermore, again by assumption, each row of $\hat{\boldsymbol{E}}$ is an erasure pattern that is correctable by $\tilde{\mathcal{C}}$. From (III.30), (III.34) results in

$$\boldsymbol{H}^{\tilde{\mathcal{C}}}\boldsymbol{\rho}^{(i)} = \boldsymbol{H}^{\tilde{\mathcal{C}}}|_{\chi(\hat{\boldsymbol{e}}_i)}\boldsymbol{\rho}^{(i)},$$

where $\hat{\boldsymbol{e}}_i$ is the i-th row of $\hat{\boldsymbol{E}}$. The above linear system of equations is full rank as $\boldsymbol{H}^{\tilde{\mathcal{C}}}|_{\chi(\hat{\boldsymbol{e}}_i)}$ is full rank. This is because $\hat{\boldsymbol{e}}_i$ is a correctable erasure pattern for $\tilde{\mathcal{C}}$. As such, the Γ symbols $\{o_j^{(i)}\}_{j\in\mathcal{J}_i}$ are obtained. From all responses, the user obtains $\Gamma d = \beta k$ code symbols of the code $\mathcal{C}$. Furthermore, from (III.31), these Γd symbols are part of the β information sets $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\beta}$ of $\mathcal{C}$. Thus, $\mathsf{H}\big(\boldsymbol{X}^{(m)}|\boldsymbol{\rho}^{(1)},\dots,\boldsymbol{\rho}^{(d)}\big) = 0$. □

Unlike [25], where the authors consider sets $\mathcal{J}_i$ with a fixed structure, we generalize the sets to match arbitrary codes $\mathcal{C}$, $\bar{\mathcal{C}}$, and $\tilde{\mathcal{C}}$. In particular, the sets in [25] were constructed targeting MDS codes, in which case the PIR rate of the $(\mathcal{C},\bar{\mathcal{C}})$-retrieval protocol is upperbounded by R_{UB} in (III.28), as mentioned earlier. However, the use of these sets for arbitrary codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ does not allow to obtain the requested file $\boldsymbol{X}^{(m)}$. Thus, Theorem 6 can be seen as a generalization of [25, Th. 7] where the PIR rate for non-MDS codes was shown to be $\mathsf{R}(\mathcal{C},\bar{\mathcal{C}}) = (d_{\min}^{\bar{\mathcal{C}}} - 1)/n < \mathsf{R}_{\mathsf{UB}}$. Our proposed protocol can achieve higher rates as illustrated in the following corollary. In particular, we will show that the upper bound R_{UB} is achievable for some classes of non-MDS codes.

Corollary 9. *If for an $[n,k]$ code $\mathcal{C}$ and an $[n,\bar{k}]$ code $\bar{\mathcal{C}}$ there exists an $[n,\tilde{k}]$ code $\tilde{\mathcal{C}} = \mathcal{C}\circ\bar{\mathcal{C}}$ of rate $R^{\tilde{\mathcal{C}}} < 1$ and an $(n-\tilde{k})$-row regular $d \times n$ binary matrix $\hat{\boldsymbol{E}}$ in which each row is a correctable erasure pattern by $\tilde{\mathcal{C}}$ and satisfying condition* C3*, then Protocol 3 achieves the upper bound* R_{UB}.

As for Protocol 2, the parameters Γ, β, and d (which are not explicitly mentioned in Theorem 6) have to be carefully selected such that a Γ-row regular matrix $\hat{\boldsymbol{E}}$ (satisfying condition C3) actually exists with a valid collection of information sets $\{\mathcal{I}_i\}_{i\in\mathbb{N}_\beta}$ for $\mathcal{C}$.

8.2 Example

Lemma 6 proves that the proposed protocol provides privacy up to $T = d_{\min}^{\tilde{\mathcal{C}}^\perp} - 1$ colluding nodes. This is illustrated in the example below.

Consider a DSS with $n = 12$ nodes that stores a single file $\boldsymbol{X}^{(1)}$ of size 1×4. $\boldsymbol{X}^{(1)}$ is encoded using the $[12,4,6]$ binary code $\mathcal{C}$ with parity-check matrix

$$\boldsymbol{H}^{\mathcal{C}} = \begin{pmatrix} 0&1&1&0&1&0&0&0&0&0&0&0\\ 1&0&1&0&0&1&0&0&0&0&0&0\\ 1&1&1&0&0&0&1&0&0&0&0&0\\ 1&1&0&1&0&0&0&1&0&0&0&0\\ 1&0&1&1&0&0&0&0&1&0&0&0\\ 0&1&1&1&0&0&0&0&0&1&0&0\\ 1&1&1&1&0&0&0&0&0&0&1&0\\ 0&0&1&1&0&0&0&0&0&0&0&1 \end{pmatrix}.$$

Let $\bar{\mathcal{C}} = \mathcal{C}$, and the code $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ has parity-check matrix

$$\boldsymbol{H}^{\tilde{\mathcal{C}}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Note that the dual code $\bar{\mathcal{C}}^{\perp}$ has minimum Hamming distance $d_{\min}^{\bar{\mathcal{C}}^{\perp}} = 3$, thus this protocol protects against $T = d_{\min}^{\bar{\mathcal{C}}^{\perp}} - 1 = 2$ colluding nodes. Choosing $\Gamma = d_{\min}^{\tilde{\mathcal{C}}} - 1 = 1$, one can use the PIR protocol as presented in [25] to get a PIR rate of $\mathrm{R}(\mathcal{C}, \bar{\mathcal{C}}) = \frac{1}{12}$. However, we can set $\Gamma = 2$ and use Protocol 3 to achieve a higher PIR rate. Note that the value of Γ cannot be greater than 2 as the number of redundant symbols in $\tilde{\mathcal{C}}$ is 2. We choose

$$\hat{\boldsymbol{E}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and $\mathcal{I}_1 = \{2, 3, 9, 12\}$. Thus, $\beta = 1$ and $d = 2$. Note that each row of $\hat{\boldsymbol{E}}$ is an erasure pattern that is correctable by the $[12, 10, 2]$ code $\tilde{\mathcal{C}}$, and that $\mathcal{I}_1$ is an information set of $\mathcal{C}$. In order to form all the queries (each query consists of $d = 2$ subqueries), we need to choose $s_1^{(9)}$, $s_1^{(12)}$, $s_2^{(2)}$, and $s_2^{(3)}$. From (III.31), we have

$$s_1^{(9)} = 1, s_1^{(12)} = 1, s_2^{(2)} = 1, \text{ and } s_2^{(3)} = 1.$$

Now, consider the first subqueries. The query vectors $\boldsymbol{q}_1^{(9)}$ and $\boldsymbol{q}_1^{(12)}$ are

$$\boldsymbol{q}_1^{(9)} = \mathring{\boldsymbol{c}}_9 + \boldsymbol{\omega}_1 = \mathring{\boldsymbol{c}}_9 + 1,$$
$$\boldsymbol{q}_1^{(12)} = \mathring{\boldsymbol{c}}_{12} + \boldsymbol{\omega}_1 = \mathring{\boldsymbol{c}}_{12} + 1,$$

and $\boldsymbol{q}_1^{(l)} = \mathring{\boldsymbol{c}}_l$, $\forall\, l \in \mathbb{N}_{12} \backslash \{9, 12\}$. The corresponding response vector is

$$\boldsymbol{\rho}^{(1)} = \underbrace{\sum_{i=1}^{1} \sum_{m=1}^{1} \begin{pmatrix} \bar{c}_{i,1}^{(m)} c_{i,1}^{(m)} \\ \bar{c}_{i,2}^{(m)} c_{i,2}^{(m)} \\ \vdots \\ \bar{c}_{i,12}^{(m)} c_{i,12}^{(m)} \end{pmatrix}}_{\in \left\{ \boldsymbol{x} \in (\mathrm{GF}(q^{\ell}))^n \colon \boldsymbol{H}^{\tilde{\mathcal{C}}} \boldsymbol{x} = \boldsymbol{0} \right\}} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ c_{1,9}^{(1)} \\ 0 \\ 0 \\ c_{1,12}^{(1)} \end{pmatrix}.$$

Finally, the user computes

$$\boldsymbol{H}^{\tilde{\mathcal{C}}} \boldsymbol{\rho}^{(1)} = \boldsymbol{H}^{\tilde{\mathcal{C}}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ c_{1,9}^{(1)} \\ 0 \\ 0 \\ c_{1,12}^{(1)} \end{pmatrix} = \begin{pmatrix} c_{1,9}^{(1)} \\ c_{1,12}^{(1)} \end{pmatrix}.$$

In a similar manner, from $\boldsymbol{\rho}^{(2)}$ the user obtains $c_{1,2}^{(1)}$ and $c_{1,3}^{(1)}$. Clearly, the indices of the symbols downloaded by the user form the information set $\mathcal{I}_1$, from which we can obtain the requested file $\boldsymbol{X}^{(1)}$. The PIR rate of the scheme is $\mathrm{R}(\mathcal{C}, \bar{\mathcal{C}}) = \frac{4}{24} = \frac{1}{6}$, i.e., double of the PIR rate of the protocol in [25]. Furthermore, it achieves the upper bound in (III.35).

A limiting factor for Protocol 3 is that the upper bound on the PIR rate R_{UB} in (III.35) depends on the dimension of $\tilde{\mathcal{C}}$. Furthermore, in order to achieve the PIR property with large T, one requires $\bar{\mathcal{C}}$ to be of large dimension such that $\bar{\mathcal{C}}^{\perp}$ has large minimum Hamming distance. Therefore, for arbitrary $\mathcal{C}$ and $\bar{\mathcal{C}}$ the chances that $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ has rate $R^{\tilde{\mathcal{C}}} = 1$ (the code does not even correct a single erasure) are quite high for large values of T, since the Hadamard product is highly nonlinear. In other words, the probability that $\mathrm{R}_{\mathsf{UB}} = 0$ is high. Below, we present code constructions for which $\mathrm{R}_{\mathsf{UB}} > 0$.

8.3 Codes for Protocol 3

As seen in the preceding sections, the codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ must be chosen such that the code $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ has rate $R^{\tilde{\mathcal{C}}} < 1$ for the PIR protocol to work. In this section, we provide a family of codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ that satisfy $R^{\tilde{\mathcal{C}}} < 1$ for a given T.

In particular, we show that a special class of UUV codes can be used for the codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ to obtain a valid $\tilde{\mathcal{C}}$. Let $\mathcal{U}$ be an $[n_1, k_1]$ code and $\mathcal{V}$ an $[n_1, 1]$ repetition code, both over GF(2). We construct the $[n = 2n_1, k = k_1 + 1]$ code $\mathcal{C} = (\mathcal{U} \mid \mathcal{U} + \mathcal{V})$ with generator matrix

$$\boldsymbol{G}^{\mathcal{C}} = \begin{pmatrix} \boldsymbol{G}^{\mathcal{U}} & \boldsymbol{G}^{\mathcal{U}} \\ \boldsymbol{0}_{1\times n_1} & \boldsymbol{1}_{1\times n_1} \end{pmatrix}. \tag{III.36}$$

Theorem 7. *Let $\mathcal{U}$ be an $[n_1, k_1]$ binary code where $n_1 \geq k_1 + 2$ and $\mathcal{V}$ an $[n_1, 1]$ binary repetition code. Then, the $[n = 2n_1, k = k_1 + 1]$ codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ constructed using* (III.36) *ensure that the vector space $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ of length n is a linear code of dimension strictly less than n, i.e., $R^{\tilde{\mathcal{C}}} < 1$.*

Proof. See Appendix G. □

Theorem 7 proves that for an arbitrary linear code $\mathcal{U}$, the UUV code ensures that $\tilde{k} < n$ and thus $\tilde{\mathcal{C}}$ is a valid code. The fact that any code $\mathcal{U}$ can be used in the protocol makes the UUV construction attractive. Also, the UUV construction may produce d_{min}-optimal binary linear codes. For instance, the codes $\mathcal{C}$ and $\bar{\mathcal{C}}$ in Section 8.2 are optimal binary linear codes. One drawback of the UUV construction, however, is that the constructed codes are in general low rate codes.

In [26], the authors showed that choosing $\mathcal{C}$ and $\bar{\mathcal{C}}$ to be RM codes with carefully selected parameters ensures that $\tilde{\mathcal{C}}$ is also an RM code of dimension $\tilde{k} < n$. However, the PIR rate is very low [26, Th. 15]. In the following subsection, we show that RM codes can indeed achieve a higher PIR rate of $\mathrm{R} = (n - \tilde{k})/n = \mathrm{R}_{\mathsf{UB}}$.

8.4 Codes Achieving the Maximum PIR Rate of Protocol 3

In order to consider the codes achieving the maximum possible PIR rate for Protocol 3, we give a definition similar to Definition 10 in Section 4.

Definition 14. *Let $\mathcal{C}$ be an $[n,k]$ code and $\bar{\mathcal{C}}$ an $[n,\bar{k}]$ code. Denote by $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ the $\tilde{k}$-dimensional code generated by the Hadamard product of $\mathcal{C}$ and $\bar{\mathcal{C}}$. A $(k+n-\tilde{k})\times n$ binary matrix $\tilde{\boldsymbol{\Lambda}}_{k,k+n-\tilde{k}}$ is called a PIR maximum rate matrix for Protocol 3 if the following conditions are satisfied.*

1. *$\tilde{\boldsymbol{\Lambda}}_{k,k+n-\tilde{k}}$ is a k-column regular matrix, and*
2. *there are exactly k rows $\{\boldsymbol{\lambda}_i\}_{i\in\mathbb{N}_k}$ and $n-\tilde{k}$ rows $\{\boldsymbol{\lambda}_{i+k}\}_{i\in\mathbb{N}_{n-\tilde{k}}}$ of $\tilde{\boldsymbol{\Lambda}}_{k,k+n-\tilde{k}}$, such that $\forall\, i \in \mathbb{N}_k$, $\chi(\boldsymbol{\lambda}_i)$ is an information set for $\tilde{\mathcal{C}}$ and $\forall\, i \in \mathbb{N}_{n-\tilde{k}}$, $\chi(\boldsymbol{\lambda}_{i+k})$ is an information set for $\mathcal{C}$.*

Similar to the case of noncolluding nodes in Section 5, it is not difficult to show that the existence of a $k \times n$ matrix $\hat{\boldsymbol{E}}$ for the code $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ and an $(n-\tilde{k}) \times n$ matrix $\bar{\boldsymbol{E}}$ for the code $\mathcal{C}$ is equivalent to the existence of $\boldsymbol{\Lambda}_{k,k+n-\tilde{k}}$.

The following corollary follows immediately from a similar reasoning as for Theorem 3.

Corollary 10. *If a PIR maximum rate matrix $\tilde{\boldsymbol{\Lambda}}_{k,k+n-\tilde{k}}$ exists for Protocol 3, then*

$$d_s^{\mathcal{C}} \geq \frac{n-\tilde{k}}{k}s, \quad \forall\, s \in \mathbb{N}_k,$$
$$d_s^{\tilde{\mathcal{C}}} \geq s, \quad \forall\, s \in \mathbb{N}_{\tilde{k}}.$$

Proof. Using an argumentation similar to the proof of Theorem 3, the existence of a PIR maximum rate matrix for Protocol 3 implies that there exist k information sets $\{\tilde{\mathcal{I}}_i\}_{i\in\mathbb{N}_k}$ of $\tilde{\mathcal{C}}$ and $n-\tilde{k}$ information sets $\{\mathcal{I}_{i'}\}_{i'\in\mathbb{N}_{n-\tilde{k}}}$ of $\mathcal{C}$ such that each coordinate j of $\mathcal{C}$ appears exactly k times in $\{\tilde{\mathcal{I}}_i\}_{i\in\mathbb{N}_k} \cup \{\mathcal{I}_{i'}\}_{i'\in\mathbb{N}_{n-\tilde{k}}}$, $j \in \mathbb{N}_n$. Hence, we obtain

$$k\left|\chi(\mathcal{D})\right| = \underbrace{\sum_{i=1}^{k}\left|\tilde{\mathcal{I}}_i \cap \chi(\mathcal{D})\right|}_{\geq 0} + \sum_{i'=1}^{n-\tilde{k}}\left|\mathcal{I}_{i'} \cap \chi(\mathcal{D})\right|$$
$$\geq \sum_{i'=1}^{n-\tilde{k}}\left|\mathcal{I}_{i'} \cap \chi(\mathcal{D})\right| \geq (n-\tilde{k})s;$$
$$k\left|\chi(\tilde{\mathcal{D}})\right| = \sum_{i=1}^{k}\left|\tilde{\mathcal{I}}_i \cap \chi(\tilde{\mathcal{D}})\right| + \underbrace{\sum_{i'=1}^{n-\tilde{k}}\left|\mathcal{I}_{i'} \cap \chi(\tilde{\mathcal{D}})\right|}_{\geq 0}$$
$$\geq \sum_{i=1}^{k}\left|\tilde{\mathcal{I}}_i \cap \chi(\tilde{\mathcal{D}})\right| \geq ks,$$

where $\mathcal{D}$ is an $[n, s]$ subcode of $\mathcal{C}$, $s \in \mathbb{N}_k$, and $\tilde{\mathcal{D}}$ is an $[n, s]$ subcode of $\tilde{\mathcal{C}}$, $s \in \mathbb{N}_{\tilde{k}}$. □

It can be seen from the proof above that we can only have $|\tilde{\mathcal{I}} \cap \chi(\mathcal{D})| \geq 0$ for an information set $\tilde{\mathcal{I}}$ of $\tilde{\mathcal{C}}$ and a subcode $\mathcal{D} \subseteq \mathcal{C}$ (or $|\mathcal{I} \cap \chi(\tilde{\mathcal{D}})| \geq 0$ for an information set $\mathcal{I}$ of $\mathcal{C}$ and a subcode $\tilde{\mathcal{D}} \subseteq \tilde{\mathcal{C}}$). Hence, unlike in Conjecture 1, we do not conjecture this necessary condition to be sufficient.

Similar to Theorem 4 for the noncolluding case, we provide a sufficient condition for codes to achieve the maximum possible PIR rate of Protocol 3 by using code automorphisms of $\mathcal{C}$ and $\tilde{\mathcal{C}}$.

Theorem 8. *Let $\mathcal{C}$ be an $[n, k]$ code, $\bar{\mathcal{C}}$ an $[n, \bar{k}]$ code, and $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$. If there exist k information sets $\tilde{\mathcal{I}}_1, \ldots, \tilde{\mathcal{I}}_k$ of $\tilde{\mathcal{C}}$, an information set $\mathcal{I}$ of $\mathcal{C}$, and $n - \tilde{k}$ distinct automorphisms of $\mathcal{C}$ such that for every code coordinate $j_i \in \mathcal{I}$, $i \in \mathbb{N}_k$,*

$$\tilde{\mathcal{I}}_i \cup \{\pi_1(j_i), \ldots, \pi_{n-\tilde{k}}(j_i)\} = \{1, 2, \ldots, n\},$$

then the codes $\mathcal{C}$ and $\tilde{\mathcal{C}}$ achieve the maximum possible PIR rate of Protocol 3, i.e., R_{UB}.

Proof. Since there exist $n - \tilde{k}$ distinct automorphisms of $\mathcal{C}$ such that $\mathcal{I}_j \triangleq \{\pi_j(j_i) \colon j_i \in \mathcal{I}\}$, $j \in \mathbb{N}_{n-\tilde{k}}$, are information sets of $\mathcal{C}$, and for every code coordinate $j_i \in \mathcal{I}$, $i \in \mathbb{N}_k$,

$$\tilde{\mathcal{I}}_i \cup \{\pi_1(j_i), \ldots, \pi_{n-\tilde{k}}(j_i)\} = \{1, 2, \ldots, n\},$$

each code coordinate $h \in \mathbb{N}_n$ appears exactly k times in $\{\tilde{\mathcal{I}}_i\}_{i \in \mathbb{N}_k} \cup \{\mathcal{I}_j\}_{j \in \mathbb{N}_{n-\tilde{k}}}$, which shows the existence of a PIR maximum rate matrix $\tilde{\boldsymbol{\Lambda}}_{k,k+n-\tilde{k}}$ for Protocol 3. □

We now show that RM codes achieve the maximum PIR rate of Protocol 3.

Corollary 11. *Let $\mathcal{C}$ be an $[n, k]$ RM code $\mathcal{R}(v, m)$, $\bar{\mathcal{C}}$ an $[n, \bar{k}]$ RM code $\mathcal{R}(\bar{v}, m)$, and $\tilde{k} = k + \bar{k}$, where $n = 2^m$, $k = \sum_{i=0}^{v} \binom{m}{i}$, and $\bar{k} = \sum_{i=0}^{\bar{v}} \binom{m}{i}$. Then, a PIR maximum rate matrix $\tilde{\boldsymbol{\Lambda}}_{k,k+n-\tilde{k}}$ exists for Protocol 3, and its PIR rate is*

$$\mathrm{R}_{\mathrm{P3}} = \frac{n - \tilde{k}}{n} = \mathrm{R}_{\mathrm{UB}}.$$

Proof. It can be easily shown that $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ is an RM code $\mathcal{R}(\tilde{v}, m)$ with $\tilde{v} = v + \bar{v}$. Consider two information sets $\mathcal{I}$ and $\tilde{\mathcal{I}}$ of $\mathcal{C}$ and $\tilde{\mathcal{C}}$, respectively. (Lemma 1 gives one way to construct these two information sets.) We construct the $k + n - \tilde{k}$ information sets

$$\begin{aligned} \tilde{\mathcal{I}}_i &\triangleq \{\boldsymbol{\sigma} + \boldsymbol{\mu}_i \colon \boldsymbol{\sigma} \in \tilde{\mathcal{I}}\}, \quad i \in \mathbb{N}_k, \\ \mathcal{I}_j &\triangleq \{\boldsymbol{\mu} + \bar{\boldsymbol{\sigma}}_j \colon \boldsymbol{\mu} \in \mathcal{I}\}, \quad j \in \mathbb{N}_{n-\tilde{k}}, \end{aligned}$$

where $\{\boldsymbol{\mu}_i\}_{i \in \mathbb{N}_k}$ and $\{\bar{\boldsymbol{\sigma}}_j\}_{j \in \mathbb{N}_{n-\tilde{k}}}$ are the numbered binary m-tuples in $\mathcal{I}$ and $\mathrm{GF}(2)^{m \times 1} \setminus \tilde{\mathcal{I}}$, respectively. From Proposition 2, $\{\tilde{\mathcal{I}}_i\}_{i \in \mathbb{N}_k}$ and $\{\mathcal{I}_j\}_{j \in \mathbb{N}_{n-\tilde{k}}}$ are information sets for $\tilde{\mathcal{C}}$ and $\mathcal{C}$, respectively.

To prove the existence of a PIR maximum rate matrix for Protocol 3 with $\mathcal{C} = \mathcal{R}(v, m)$ and $\tilde{\mathcal{C}} = \mathcal{R}(\bar{v}, m)$, it is sufficient to show that each m-tuple representing a coordinate of the RM code appears exactly k times in the $k + n - \tilde{k}$ constructed information sets above. Without loss of generality, the i-th information set $\tilde{\mathcal{I}}_i$, $i \in \mathbb{N}_k$, can be written as

$$\tilde{\mathcal{I}}_i = \{\boldsymbol{\mu}_i + \boldsymbol{\sigma}_1, \dots, \boldsymbol{\mu}_i + \boldsymbol{\sigma}_{\tilde{k}}\},$$

where $\boldsymbol{\sigma}_{j'} \in \tilde{\mathcal{I}}$, $j' \in \mathbb{N}_{\tilde{k}}$. Furthermore, consider the i-th elements across all sets $\mathcal{I}_j$, $j \in \mathbb{N}_{n-\tilde{k}}$. They have the form

$$\boldsymbol{\mu}_i + \bar{\boldsymbol{\sigma}}_j,$$

where $\boldsymbol{\mu}_i \in \mathcal{I}$. Since $\bar{\boldsymbol{\sigma}}_j \in \mathrm{GF}(2)^{m\times 1} \setminus \tilde{\mathcal{I}}$ and $\boldsymbol{\sigma}_i \in \tilde{\mathcal{I}}$, the set

$$\{\boldsymbol{\mu}_i + \boldsymbol{\sigma}_1, \dots, \boldsymbol{\mu}_i + \boldsymbol{\sigma}_{\tilde{k}}\} \cup \{\boldsymbol{\mu}_i + \bar{\boldsymbol{\sigma}}_1, \dots, \boldsymbol{\mu}_i + \bar{\boldsymbol{\sigma}}_{n-\tilde{k}}\}$$

with cardinality $n = 2^m$ is equal to $\mathrm{GF}(2)^{m\times 1}$, i.e., the set containing the elements of the i-th information set $\tilde{\mathcal{I}}_i$ and the i-th elements $\boldsymbol{\mu}_i + \bar{\boldsymbol{\sigma}}_j$ in all sets $\mathcal{I}_j$ is equal to the set of all binary $n = 2^m$ tuples. Therefore, we are able to find k information sets $\{\tilde{\mathcal{I}}_i\}_{i\in\mathbb{N}_k}$ of $\tilde{\mathcal{C}}$, an information set $\mathcal{I}$ of $\mathcal{C}$, and $n - \tilde{k}$ distinct automorphisms $\pi_j(\boldsymbol{\mu}) = \boldsymbol{\mu} + \bar{\boldsymbol{\sigma}}_j$ of $\mathcal{C}$, $j \in \mathbb{N}_{n-\tilde{k}}$, satisfying Theorem 8. This completes the proof. □

We remark again that Corollary 11 can be extended to nonbinary generalized RM codes. Finally, note that in the independent work [28] it was also shown that RM codes can achieve the maximum possible PIR rate of the $(\mathcal{C}, \tilde{\mathcal{C}})$-retrieval protocol, i.e., R_{UB}, for transitive codes. However, it is important to highlight that our Protocol 3 requires a much smaller β (number of stripes) and a significant smaller d (number of subqueries). Indeed, the protocol in [28] requires very large β and d (in the order of 10000 for the example provided), and thus our protocol is more practical.

8.5 Optimizing the PIR rate

For those codes for which we do not have a proof that R_{UB} is achieved, we now provide an algorithm to optimize the PIR rate R_{P3} such that it comes closer to the upper bound R_{UB}. The algorithm is identical to Algorithm 1 for the case of noncolluding nodes with some key differences which we highlight below.

- In Line 1, Γ in initialized to 1.
- The while loop in Line 4 runs up to $n - \tilde{k}$.
- The first argument to the subprocedure `ComputeErasurePatternList`$(\cdot, \Gamma)$ is changed from $\mathcal{C}$ to $\mathcal{C} \circ \tilde{\mathcal{C}}$ in Line 5.

With these minor modifications, Algorithm 1 can be used to optimize the PIR rate in the case of T colluding nodes. Numerical results are presented below in Section 9. Note that Γ is initialized to 1 as opposed to $\min\left(k, d^{\mathcal{C}}_{\mathrm{min}} - 1\right)$ in the

Code	$d_{\min}^{\mathcal{C}}$	$d_{\min}^{\mathcal{C}'}$	$\mathrm{R}_{\text{non-opt}}$	R_{opt}	C_{∞}
$\mathcal{C}_1$: [5, 3] (Example 4)	2	3	0.4	0.4	0.4
$\mathcal{C}_2$: [11, 6]	4	4	0.2727	0.4545	0.4545
$\mathcal{C}_3$: [12, 8] Pyramid	4	4	0.25	0.3333	0.3333
$\mathcal{C}_4$: [18, 12] Pyramid	5	5	0.2222	0.3333	0.3333
$\mathcal{C}_5$: [16, 10] LRC	5	5	0.25	0.3750	0.3750
$\mathcal{C}_6$: [154, 121] LRC	4	6	0.0325	0.2013	0.2143
$\mathcal{C}_7$: [187, 121] LRC	7	16	0.0802	0.3262	0.3529

Table III.2: Optimized values for the PIR rate for different codes having code rates strictly larger than 1/2 for the case of noncolluding nodes.

Code	$d_{\min}^{\mathcal{C}}$	$\mathrm{R}_{\text{non-opt}}$	R_{opt}	C_{∞}
$\mathcal{C}_8$: [7, 3] (Example 5)	4	0.4286	0.5714	0.5714
$\mathcal{C}_9$: [9, 4] LRC ([23, Ex. 1])	5	0.4444	0.5555	0.5555
$\mathcal{C}_{10}$: [12, 6] LRC ([23, Ex. 2])	6	0.4167	0.5	0.5

Table III.3: Optimized values for the PIR rate for different codes having code rates at most 1/2 for the case of noncolluding nodes.

case of noncolluding nodes. This is because $\hat{\boldsymbol{E}}$ and $\bar{\boldsymbol{E}}$ of $\boldsymbol{E}$ are based on different codes. This also guarantees that the algorithm always returns $\boldsymbol{E}_{\text{opt}} \neq \emptyset$ (assuming $d_{\min}^{\tilde{\mathcal{C}}} \geq 2$), since in this case all weight-1 erasure patterns are correctable by $\tilde{\mathcal{C}}$, and a valid matrix $\boldsymbol{E}$ can be trivially constructed.

9 Numerical Results

In this section, we present maximized PIR rates for the PIR protocols described in Sections 4, 5 and 8. Unless specified otherwise, these protocols are optimized using Algorithm 1 with minimum possible values for the parameters β and d as given in (III.6). In contrast to Sections 6 to 6.3, where different classes of codes were proved to be MDS-PIR capacity-achieving, we consider here other codes (with two exceptions as detailed below) and their highest possible PIR rates. The results are tabulated in Tables III.2 and III.3 for the case of noncolluding nodes, and in Table III.4 for the colluding case. Results in Table III.2 are for code rates strictly larger than 1/2, while codes of rate at most 1/2 are tabulated in Table III.3.

In Tables III.2 and III.3, C_{∞} (see (III.8)) is the asymptotic MDS-PIR capacity and R_{opt} is the optimized PIR rate computed from Algorithm 1. In Table III.2, $\mathrm{R}_{\text{non-opt}} = \frac{d_{\min}^{\mathcal{C}'}-1}{n}$, while in Table III.3, $\mathrm{R}_{\text{non-opt}} = \frac{d_{\min}^{\mathcal{C}}-1}{n}$. In Table III.4, $\mathrm{C}_{\mathrm{LB},\infty} \triangleq \frac{n-(k+T-1)}{n}$ is a lower bound (taken from [25]) on the asymptotic MDS-PIR capacity in the case of at most T colluding nodes, while R_{opt} is the optimized PIR rate computed from Algorithm 1 and $\mathrm{R}_{\text{non-opt}} = \frac{d_{\min}^{\tilde{\mathcal{C}}}-1}{n}$.

The code $\mathcal{C}_1$ in Table III.2 is from Example 4, $\mathcal{C}_2$ is an [11, 6] binary linear code with optimum minimum Hamming distance, while codes $\mathcal{C}_3$ and $\mathcal{C}_4$ are Pyramid

Code $\mathcal{C}$	$\bar{\mathcal{C}}$	$\tilde{\mathcal{C}}$	$d_{\min}^{\mathcal{C}}$	T	$\mathrm{R}_{\text{non-opt}}$	R_{opt}	R_{UB}	$\mathrm{C}_{\text{LB},\infty}$
$\mathcal{C}_9$: $[9,4]$ LRC ([23, Ex. 1])	RS$[9,2]$	RS$(9,6)$	5	2	0.3333	0.3333	0.3333	0.4444
$\mathcal{C}_{10}$: $[12,6]$ LRC ([23, Ex. 2])	RS$[12,2]$	RS$(12,8)$	6	2	0.3333	0.3333	0.3333	0.4167
$\mathcal{C}_{11}$: $[12,4]$ (Section 8.2)	$\mathcal{C}$	$[12,10,2]$	6	2	0.0833	0.1667	0.1667	0.5833
$\mathcal{C}_{12}$: $[12,4]$ LRC ([23, Ex. 5])	RS$[12,2]$	$[12,7,5]$	6	2	0.3333	0.4167	0.4167	0.5833
$\mathcal{C}_{13}$: $[26,9]$ $(\mathcal{U} \mid \mathcal{U}+\mathcal{V})$	$\mathcal{C}$	$[26,22,1]$	8	3	0	0.1538	0.1538	0.5769
$\mathcal{C}_{14}$: $[32,6]$ $(\mathcal{U} \mid \mathcal{U}+\mathcal{V})$	$\mathcal{C}$	$[32,16,8]$	16	3	0.2188	0.5	0.5	0.75

Table III.4: Optimized values for the PIR rate for different codes for the colluding case with $T = 2$ and $T = 3$.

codes, taken from [19], of locality 4 and 6, respectively, $\mathcal{C}_5$ is an LRC of locality 5 borrowed from [20]. In [39], a construction of optimal (in terms of minimum Hamming distance) binary LRCs with multiple repair groups was given. In particular, in [39, Constr. 3], a construction based on array LDPC codes was provided. The minimum Hamming distance of array LDPC codes is known for certain sets of parameters (see, e.g., [40] and references therein). Codes $\mathcal{C}_6$ and $\mathcal{C}_7$ in Table III.2 are *optimal* LRCs based on array LDPC codes constructed using [39, Constr. 3] and having information locality 11. The protocols for these two underlying codes have $\beta = \Gamma$ and $d = k$.

Code $\mathcal{C}_8$ in Table III.3 is the dual code of the $[7,4,3]$ Hamming code and is taken from Example 5, while the codes $\mathcal{C}_9$ and $\mathcal{C}_{10}$ are $d_{\min}$-optimal LRCs over GF(13) of all-symbol locality 2 and 3, respectively, taken from [23] (see Examples 1 and 2, respectively, in [23]). These two codes are also tabulated in Table III.4. The corresponding $\bar{\mathcal{C}}$ codes are RS codes and their parameters are given in Table III.4 (an RS code of length n and dimension k is denoted by RS$[n,k]$). Code $\mathcal{C}_{11}$ (from Table III.4) is taken from Section 8.2, while code $\mathcal{C}_{12}$ (also from Table III.4) is taken from [23] (see Example 5 in [23]). Note that $\mathcal{C}_{12}$ is an LRC of length 12 over GF(13) with two disjoint recovering sets of sizes 2 and 3, respectively, for every symbol of the code (all-symbol locality). Code $\mathcal{C}_{13}$ (from Table III.4) is a $[26,9,8]$ binary UUV code that is close to an optimal binary linear code (the best known code for these parameters has a minimum Hamming distance of 9), while the code $\mathcal{C}_{14}$ is a UUV code where $\mathcal{U}$ is a $[16,5,8]$ RM code (the code $\mathcal{R}(1,4)$). Note that $\mathcal{C}_{14}$ becomes the RM code $\mathcal{R}(1,5)$. Due to the high computational complexity of Algorithm 1 for $\mathcal{C}_{14}$, we are unable to compute the maximum rate of Protocol 3 for the minimum values of β and d. Instead, we take $\beta = \Gamma$ and $d = k$ and use Corollary 11 to obtain the maximum rate of the protocol.

It is observed in Tables III.2 and III.3 that in the case of noncolluding nodes, the optimized PIR rate R_{opt} is equal to the asymptotic capacity C_∞ for all tabulated codes except $\mathcal{C}_6$ and $\mathcal{C}_7$. Note that the codes $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_5$–$\mathcal{C}_{10}$ do not fall within the code families that we proved are MDS-PIR capacity-achieving (see Section 6). Thus, the results in Tables III.2 and III.3 show that, interestingly, other codes can achieve the asymptotic MDS-PIR capacity as well. On the other hand, $\mathcal{C}_3$ and $\mathcal{C}_4$ satisfy the conditions of Theorem 5. Thus, they are MDS-PIR capacity-achieving with $\beta = \Gamma$ and $d = k$. The results in the table show that they also achieve C_∞ for β and d as in (III.6). Also, note that by the nature of the optimization procedure (see Remark 2), MDS-PIR capacity-achieving matrices $\Lambda_{k,n}$ of all tabu-

lated codes except $\mathcal{C}_6$ and $\mathcal{C}_7$ are found. This implies that they are also MDS-PIR capacity-achieving codes for any finite number of files and must satisfy the necessary condition based on generalized Hamming weights in Theorem 3. Due to the high computational complexity of Algorithm 1, it is difficult to maximize the PIR rates of $\mathcal{C}_6$ and $\mathcal{C}_7$. Therefore, it is an open problem whether or not they are MDS-PIR capacity-achieving codes. For the colluding case (see Table III.4) the lower bound $\mathrm{C}_{\mathrm{LB},\infty}$ on the asymptotic MDS-PIR capacity is not achieved, even after optimization. To the best of our knowledge, GRS codes are the only known class of codes where this bound is actually achieved [25]. On the other hand, the upper bound R_{UB} (from (III.35)) is attained in all cases.

10 Conclusion

We presented three different PIR protocols, namely Protocol 1, Protocol 2, and Protocol 3, for DSSs where data is stored using an arbitrary linear code. We first considered the case when no nodes in the DSS collude. Under this scenario, Protocols 1 and 2 achieve the PIR property. We proved that, for certain non-MDS codes, Protocol 1 achieves the finite MDS-PIR capacity (and also the asymptotic MDS-PIR capacity) and Protocol 2, which is a much simpler protocol compared to Protocol 1, achieves the asymptotic MDS-PIR capacity. Thus, the MDS property is not necessary in order to achieve the MDS-PIR capacity (both finite and asymptotic). We also provided a necessary and a sufficient condition for codes to be MDS-PIR capacity-achieving. The necessary condition is based on generalized Hamming weights while the sufficient condition is obtained from automorphisms of linear codes. We proved that cyclic codes, RM codes, and distance-optimal information locality codes are MDS-PIR capacity-achieving codes. For other codes, we provided an optimization algorithm that optimizes Protocols 1 and 2 in order to maximize their PIR rates. We also considered the scenario where a subset of nodes in the DSS collude. For such a scenario, we proposed Protocol 3, which is an improvement of the PIR protocol by Freij-Hollanti *et al.* The improvement allows the protocol to achieve even higher PIR rates, and the PIR rates for non-MDS codes are no longer limited by their code rates. Subsequently, we presented an optimization algorithm to optimize the PIR rate of the protocol, and a family of codes based on the classical $(\mathcal{U} \mid \mathcal{U} + \mathcal{V})$ construction that can be used with this protocol. Furthermore, as for the noncolluding case, we provided a necessary and a sufficient condition to achieve the maximum possible PIR rate of Protocol 3. Moreover, we proved that RM codes satisfy the sufficient condition and can achieve much higher PIR rates than previously reported by Freij-Hollanti *et al.* Finally, we presented some numerical results on the PIR rates for several linear codes, including distance-optimal all-symbol locality LRCs constructed by Tamo and Barg.

A Proof of Lemma 1

We need to ensure that given a $k \times n$ generator matrix $\boldsymbol{G}$ of $\mathcal{R}(v,m)$ with $k = \sum_{i=0}^{v}\binom{m}{i}$ and $n = 2^m$, the $k \times k$ matrix $\boldsymbol{G}|_{\mathcal{I}}$ that comprises the columns of the generator matrix indexed by the coordinates of $\mathcal{I}$ is invertible. We are going to elaborate on this by considering all the monomials $z_1^{\mu_1}\cdots z_m^{\mu_m}$, $\mu_i \in \mathrm{GF}(2)$, in a so-called *graded lexicographic order*, where each vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)^{\mathsf{T}} \in \mathrm{GF}(2)^{m\times 1}$ defines a column of the generator matrix $\boldsymbol{G}$ according to (III.1). Formally speaking, denote $z_1^{\mu_1}\cdots z_m^{\mu_m}$ by $\boldsymbol{z}^{\boldsymbol{\mu}}$. We say $\boldsymbol{z}^{\boldsymbol{\mu}} \prec \boldsymbol{z}^{\boldsymbol{\mu}'}$ either if $w_{\mathsf{H}}(\boldsymbol{\mu}) < w_{\mathsf{H}}(\boldsymbol{\mu}')$ or if $w_{\mathsf{H}}(\boldsymbol{\mu}) = w_{\mathsf{H}}(\boldsymbol{\mu}')$ and the topmost nonzero entry of $\boldsymbol{\mu} - \boldsymbol{\mu}'$ (subtraction is over the reals) is positive. For instance, in graded lexicographical ordering we have $z_1 \prec z_2 \prec z_3 \prec z_1z_2 \prec z_1z_3 \prec z_2z_3 \prec z_1z_2z_3$ for $m = 3$.

Now we are ready for the proof. It is noted that a basis of $\mathcal{R}(v,m)$ can be viewed as $\mathcal{B} \triangleq \{1, z_1, z_2, z_3, \ldots\} = \{\boldsymbol{z}^{\boldsymbol{\mu}'} : w_{\mathsf{H}}(\boldsymbol{\mu}') \le v\}$. Let us list the monomials in $\mathcal{B}$ in graded lexicographical order, and let the ℓ-th monomial $f_\ell(\boldsymbol{z})$ of the ordered list represent the ℓ-th row of $\boldsymbol{G}$, $\ell \in \mathbb{N}_k$. According to the generator matrix construction of $\mathcal{R}(v,m)$, it is known that the $(\ell, \boldsymbol{\mu})$ entry of $\boldsymbol{G}$ is equal to the value of the ℓ-th monomial $f_\ell(\boldsymbol{z})$ at $\boldsymbol{z} = \boldsymbol{\mu}$ [27, Ch. 13]. Furthermore, given a column coordinate $\boldsymbol{\mu} \in \mathcal{I}$, for $\ell \in \mathbb{N}_k$, we have

$$f_\ell(\boldsymbol{\mu}) = \begin{cases} 1 & \text{if } \boldsymbol{z}^{\boldsymbol{\mu}} = f_\ell(\boldsymbol{z}), \\ 0 & \text{if } \boldsymbol{z}^{\boldsymbol{\mu}} \prec f_\ell(\boldsymbol{z}). \end{cases}$$

Thus, the $(\boldsymbol{z}^{\boldsymbol{\mu}}, \boldsymbol{\mu})$ entry can be seen as a pivot of $\boldsymbol{G}|_{\mathcal{I}}$ and $\boldsymbol{G}|_{\mathcal{I}}$ is obviously invertible.

B Proof of Theorem 1

The proof is completed by showing that the following statements are true.

File symmetry within each storage node. For all repetitions, we investigate file symmetry for every possible combination of files in each round within each storage node. In the first round ($\ell = 1$) of all κ repetitions, it follows from (III.9) that, for each $m' \in \mathbb{N}_{2:f}$, the downloaded number of undesired symbols $y_{s,j}^{(m')}$ is equal to $\kappa \mathsf{U}(1) = \kappa^{f-1}$, while for the desired symbols, from (III.10), it follows that the user requests κ^{f-1} code symbols for $y_{s,j}^{(1)}$. In the $(\ell+1)$-th round of all κ repetitions, $\ell \in \mathbb{N}_{f-1}$, arbitrarily choose a combination of files $\mathcal{M} \subseteq \mathbb{N}_{2:f}$, where $|\mathcal{M}| = \ell$. It follows from (III.11) that the total number of requested desired symbols for files pertaining to $\{1\} \cup \mathcal{M}$ is equal to

$$\begin{aligned}(v-\kappa)\big[\big(\mathsf{U}(\ell)-1\big) - \mathsf{U}(\ell-1) + 1\big] &= (v-\kappa)\kappa^{f-(\ell+1)}(v-\kappa)^{\ell-1} \\ &= \kappa^{f-(\ell+1)}(v-\kappa)^{\ell}.\end{aligned}$$

On the other hand, for the undesired symbols, it follows from (III.9) that in the $(\ell+1)$-th round the user requests

$$\kappa\big[\big(\mathsf{U}(\ell+1)-1\big) - \mathsf{U}(\ell) + 1\big] = \kappa\kappa^{f-(\ell+2)}(v-\kappa)^{\ell} = \kappa^{f-(\ell+1)}(v-\kappa)^{\ell}$$

linear sums for a combination of files $\mathcal{M} \subseteq \mathbb{N}_{2:f}$, $|\mathcal{M}| = \ell + 1$. Thus, in rounds $\mathbb{N}_{f-1}$, an equal number of linear sums for all combinations of files $\mathcal{M} \subseteq \mathbb{N}_f$ are downloaded. By construction, these are linear sums of unique code symbols pertaining to f files. Thus, symmetry in all $f-1$ rounds is ensured. In the f-th round, only desired symbols are downloaded. Since each desired symbol is a linear combination of code symbols from all f files, an equal number of linear sums is again downloaded from each file. Therefore, symmetry within each node and in each round is ensured.

The $\beta \times k$ file $\boldsymbol{X}^{(1)}$ can be reliably decoded. In the first round ($\ell = 1$) of all κ repetitions, $\forall\, s \in \mathbb{N}_{\kappa^{f-1}}$, the user has downloaded the matrix

$$\begin{pmatrix} y^{(1)}_{\kappa^{f-1}(a_{1,1}-1)+s,1} & \cdots & y^{(1)}_{\kappa^{f-1}(a_{1,n}-1)+s,n} \\ \vdots & \cdots & \vdots \\ y^{(1)}_{\kappa^{f-1}(a_{\kappa,1}-1)+s,1} & \cdots & y^{(1)}_{\kappa^{f-1}(a_{\kappa,n}-1)+s,n} \end{pmatrix}$$

of code symbols. Given an $a \in \mathbb{N}_v$, recalling Definitions 11 and 12, it follows that for each $s \in \mathbb{N}_{\kappa^{f-1}}$, the coordinate set $\mathcal{S}\big(\kappa^{f-1}(a-1)+s \big| \kappa^{f-1}(\boldsymbol{A}_{\kappa\times n}-\boldsymbol{1}_{\kappa\times n})+s\boldsymbol{1}_{\kappa\times n}\big)$ contains an information set. Hence, the $(\kappa^{f-1}(a-1)+1)$-th, ..., $(\kappa^{f-1}(a-1)+\kappa^{f-1})$-th stripes are recovered. Since $a_{i,j} \in \mathbb{N}_v$, we know until now that the user has obtained the 1-st, 2-nd, ..., $(\kappa^{f-1}(v-1)+\kappa^{f-1})$-th stripes. Note that $\kappa^{f-1}(v-1)+\kappa^{f-1} = \mathrm{D}(0)v$. Moreover, owing to (III.12), in the $(\ell' = \ell+1)$-th round of all κ repetitions with $\ell \in \mathbb{N}_{f-1}$, $\forall\, s \in \mathbb{N}_{\mathrm{D}(\ell-1):(\mathrm{D}(\ell)-1)}$ the matrices

$$\begin{pmatrix} y^{(1)}_{s\cdot v+a_{1,1},1} & \cdots & y^{(1)}_{s\cdot v+a_{1,n},n} \\ \vdots & \cdots & \vdots \\ y^{(1)}_{s\cdot v+a_{\kappa,1},1} & \cdots & y^{(1)}_{s\cdot v+a_{\kappa,n},n} \end{pmatrix}$$

of code symbols are downloaded. Similarly, fix an $s \in \mathbb{N}_{\mathrm{D}(\ell-1):(\mathrm{D}(\ell)-1)}$. Then, $\forall\, a \in \mathbb{N}_v$, the coordinate set $\mathcal{S}\big(sv+a \big| sv\boldsymbol{1}_{\kappa\times n}+\boldsymbol{A}_{\kappa\times n}\big)$ must contain an information set, and the user can recover the $(sv+1)$-th, ..., $(sv+v)$-th stripes. Observe that in the last $(\ell' = (f-1)+1)$-th round, the row index of the last recovered strip is equal to $(\mathrm{D}(f-1)-1)v+v$. Hence, the total number of stripes the user has recovered is

$$\begin{aligned} \big(\mathrm{D}(f-1)-1\big)v+v &= \left[\sum_{\ell=0}^{f-1}\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(v-\kappa)^{\ell}-1\right]v+v \\ &= (v^{f-1}-1)v+v = v^f. \end{aligned}$$

This indicates that the user has restored all v^f stripes for $\boldsymbol{X}^{(1)}$, and $\boldsymbol{X}^{(1)}$ is in fact reliably reconstructed.

The PIR achievable rate is expressed as (III.13)**.** According to (III.9), since there are $\binom{f-1}{\ell}$ combinations of files other than the first file with index $m = 1$, the user

has downloaded

$$\kappa\binom{f-1}{\ell}\big[\mathrm{U}(\ell)-1-\mathrm{U}(\ell-1)+1\big]=\kappa\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell-1}$$
$$=\binom{f-1}{\ell}\kappa^{f-\ell}(\nu-\kappa)^{\ell-1}$$

undesired symbols from each storage node in the ℓ-th round, $\ell\in\mathbb{N}_{f-1}$. Moreover, from (III.10) and (III.11), the user has downloaded κ^{f-1} desired symbols from each storage node in round $\ell=1$, and

$$\mathrm{D}(\ell)-1-\mathrm{D}(\ell-1)+1=\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell}$$

extra desired symbols from each storage node in the $(\ell+1)$-th round, $\ell\in\mathbb{N}_{f-1}$. In summary, the total download cost for Protocol 1 using $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ is equal to

$$\begin{aligned}
nd &= \text{total number of undesired symbols} + \text{ total number of desired symbols}\\
&=\kappa n\sum_{\ell=1}^{f-1}\binom{f-1}{\ell}\kappa^{f-\ell}(\nu-\kappa)^{\ell-1}+\kappa n\sum_{\ell=0}^{f-1}\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell}\\
&=\kappa n\left[\frac{\kappa}{\nu-\kappa}\sum_{\ell=1}^{f-1}\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell}+\sum_{\ell=0}^{f-1}\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell}\right]\\
&=\kappa n\left[\frac{\kappa}{\nu-\kappa}(\nu^{f-1}-\kappa^{f-1})+\nu^{f-1}\right]\\
&=\frac{\kappa n}{\nu-\kappa}\left[\kappa\nu^{f-1}-\kappa^{f}+\nu^{f}-\kappa\nu^{f-1}\right]\\
&=\frac{\kappa n}{\nu-\kappa}\left[\nu^{f}-\kappa^{f}\right].
\end{aligned}$$

Therefore, the PIR achievable rate $\mathrm{R}(\mathcal{C})$ is given by

$$\mathrm{R}(\mathcal{C})=\frac{\beta k}{nd}=\frac{\nu^{f}k}{\frac{\kappa n}{\nu-\kappa}\left[\nu^{f}-\kappa^{f}\right]}=\frac{\frac{(\nu-\kappa)k}{\kappa n}}{\left[1-\left(\frac{\kappa}{\nu}\right)^{f}\right]}.$$

C Proof of Lemma 3

By setting $\kappa=k$ and using Definition 11, we prove the existence of $\boldsymbol{\Lambda}_{k,\nu}$ with $\nu=k+\min\left(k,d^{\mathcal{C}}_{\min}-1\right)$. In fact, given an $[n,k,d^{\mathcal{C}}_{\min}]$ code $\mathcal{C}$, observe that for an interference matrix $\boldsymbol{A}_{k\times n}$ derived from a valid $\boldsymbol{\Lambda}_{k,\nu}$, $\mathcal{S}(a|\boldsymbol{A}_{k\times n})$ must contain an information set $\forall\, a\in\mathbb{N}_{\nu}$. We first choose $\Gamma=\min\left(k,d^{\mathcal{C}}_{\min}-1\right)$ information sets of $\mathcal{C}$. Note that since every code contains at least one information set, one can always arbitrarily choose Γ information sets even if some of them are repeatedly

chosen. Let us denote the selected information sets by $\mathcal{I}_i$, $i \in \mathbb{N}_\Gamma$, and start to construct the corresponding matrix $\boldsymbol{A}_{k\times n}$ with

$$a_{i,j} = k + i, \text{ if } j \in \mathcal{I}_i,\ i \in \mathbb{N}_\Gamma. \tag{III.37}$$

In this way, $k\Gamma$ entries of $\boldsymbol{A}_{k\times n}$ are constructed. Next, denote the remaining non-constructed entries in each column of $\boldsymbol{A}_{k\times n}$ by

$$\mathcal{A}_j \triangleq \left\{a_{i_1^{(j)},j}, \dots, a_{i_{s(j)}^{(j)},j}\right\}, \quad j \in \mathbb{N}_n,$$

where $s(j) \leq k$ is the total number of nonconstructed entries in each column. Hence, there are in total $kn - k\Gamma = k(n-\Gamma)$ nonconstructed entries as follows,

$$\left\{a_{i_1^{(1)},1}, \dots, a_{i_{s(1)}^{(1)},1}, \dots, a_{i_1^{(n)},n}, \dots, a_{i_{s(n)}^{(n)},n}\right\}. \tag{III.38}$$

If we consecutively assign $\{1, \dots, k\}$ to the entries of $\boldsymbol{A}_{k\times n}$ in (III.38) and repeat this process $n-\Gamma$ times, the remaining $k(n-\Gamma)$ entries of $\boldsymbol{A}_{k\times n}$ will certainly be constructed. Note that since we consecutively assign values of $\mathbb{N}_k$ and the largest number of empty entries of each column of $\boldsymbol{A}_{k\times n}$ is k, it is impossible to have repeated values of $\mathbb{N}_k$ in each column of the constructed $\boldsymbol{A}_{k\times n}$. From (III.37) and (III.38), it can be seen that each $a \in \mathbb{N}_k$ occurs in $n-\Gamma$ columns of $\boldsymbol{A}_{k\times n}$. From Proposition 3, we can then say that the set $\mathcal{S}(a|\boldsymbol{A}_{k\times n})$ of cardinality $n-\Gamma \geq n-(d_{\min}^{\mathcal{C}}-1)$ contains an information set. For the remaining $a \in \mathbb{N}_{k+1:k+\Gamma}$, (III.37) ensures that $\mathcal{S}(a|\boldsymbol{A}_{k\times n})$ contains an information set. Thus, this procedure will result in a valid PIR interference matrix $\boldsymbol{A}_{k\times n}$. The proof is then completed, since we can construct a PIR achievable rate matrix $\boldsymbol{\Lambda}_{k,k+\Gamma}$ from $\boldsymbol{A}_{k\times n}$.

D Proof of Theorem 2

Consider the i-th subresponse of each response $\boldsymbol{r}_l$. Out of the n subresponses generated from the n storage nodes, there are Γ subresponses originating from a subset of nodes $\mathcal{J} \subset \mathbb{N}_n$, $|\mathcal{J}| = \Gamma$, of the form

$$r_{l,i} = Y_l + c_{s,l}^{(m)}, \quad \forall\, l \in \mathcal{J},\ s \in \mathbb{N}_\beta. \tag{III.39}$$

Y_l is referred to as *code interference symbol*. Considering $\boldsymbol{G}^{\mathcal{C}} = (g_{i',l})$, for $l \in \mathbb{N}_n$, each code symbol and code interference symbol have the form

$$c_{s,l}^{(m)} = \sum_{i'=1}^{k} g_{i',l} x_{s,i'}^{(m)}, \tag{III.40}$$

$$Y_l = \sum_{i'=1}^{k} g_{i',l} I_{(i-1)k+i'}, \tag{III.41}$$

where $x_{s,i'}^{(m)}$ is an information symbol of $\mathcal{C}$, and

$$I_{(i-1)k+i'} = \sum_{m=1}^{f} \sum_{i''=(m-1)\beta+1}^{m\beta} u_{i,i''} x_{i''-(m-1)\beta,i'}^{(m)}$$

is an interference symbol. To obtain Γ code symbols from (III.39), the user requires the knowledge of the code interference symbols Y_l. This is obtained from the remaining $n - \Gamma$ subresponses of the nodes in $\overline{\mathcal{J}} \triangleq \mathbb{N}_n \setminus \mathcal{J}$, which are

$$r_{l,i} = Y_l, \quad \forall\, l \in \overline{\mathcal{J}}. \tag{III.42}$$

From (III.40) and (III.41) we can observe that the interference symbols Y_l have the same form as the code symbols of $\mathcal{C}$. Since there are Γ unknowns, solving (III.42) resembles ML decoding of the code $\mathcal{C}$. (III.42) is a full rank system in the unknowns $I_1, \ldots, I_k$ (from the third requirement of $\hat{\boldsymbol{E}}$ in Section 5) in $\mathrm{GF}(q^\ell)$. Hence, knowing the interference symbols allows the recovery of Γ *unique* (from the first requirement for $\hat{\boldsymbol{E}}$ in Section 5) code symbols from the i-th subquery as the user has the knowledge of Y_l, $l \in \mathcal{J}$. In a similar way, from all subqueries, the user obtains $d\Gamma = \beta k$ *unique* code symbols pertaining to file $\boldsymbol{X}^{(m)}$. These βk code symbols are part of β information sets (from the second requirement of $\hat{\boldsymbol{E}}$ in Section 5 and (III.18)). Furthermore, since each information set is implicitly linked to a unique stripe of the requested file and $s \in \mathbb{N}_\beta$ (see (III.39)) is selected (without repetition) from $\mathcal{F}_l$ (see (III.18)), k code symbols from each stripe are obtained, and the user can recover the whole file $\boldsymbol{X}^{(m)}$, from which it follows that $\mathsf{H}\big(\boldsymbol{X}^{(m)} | \boldsymbol{r}_1, \ldots, \boldsymbol{r}_n\big) = 0$.

E Proof of Lemma 5

We prove the inequality by using the well-known Sylvester's rank inequality:[7] If $\boldsymbol{U}$ is an $s \times k$ matrix and $\boldsymbol{G}$ is a matrix of size $k \times n$, then

$$\operatorname{rank}(\boldsymbol{U}\boldsymbol{G}) \geq \operatorname{rank}(\boldsymbol{U}) + \operatorname{rank}(\boldsymbol{G}) - k.$$

Let $\mathcal{C}$ be an $[n, k]$ code with generator matrix $\boldsymbol{G}$. Given an arbitrary information set $\mathcal{I}$, $\boldsymbol{G}|_{\mathcal{I}}$ is by definition invertible (see Definition 1). We next choose an arbitrary subcode $\mathcal{D} \subseteq \mathcal{C}$ of rank s that can be generated by $\boldsymbol{U}\boldsymbol{G}$ for some $s \times k$ matrix $\boldsymbol{U}$ of rank s.

Applying Sylvester's rank inequality, we have

$$\operatorname{rank}(\boldsymbol{U}(\boldsymbol{G}|_{\mathcal{I}})) \geq s + k - k = s.$$

Because each basis vector of the space $\boldsymbol{U}(\boldsymbol{G}|_{\mathcal{I}})$ must at least contain one nonzero component, this leads to

$$\big|\mathcal{I} \cap \chi(\mathcal{D})\big| = \big|\chi(\mathcal{D}|_{\mathcal{I}})\big| = \big|\chi(\boldsymbol{U}(\boldsymbol{G}|_{\mathcal{I}}))\big| \geq s,$$

where $\chi(\mathcal{D})$ is the support of $\mathcal{D}$ (see Definition 2).

F Proof of Theorem 5

The proof is a two-step procedure. First, we prove that all rows in $\boldsymbol{E}$ after Step a) are correctable by $\mathcal{C}$. Secondly, we prove that the swaps in certain rows in Step

[7] The proof of this inequality is available in the literature on linear algebra, so here we omit the proof.

b) ensure that the resulting rows are correctable erasure patterns. We start by proving two key lemmas (Lemmas 7 and 8 below), which will form the basis of the overall proof of the theorem.

Lemma 7. *Let $\mathcal{C}$ be an $[n, k]$ distance-optimal (r, δ) information locality code consisting of L_c local codes and with parity-check matrix as in* (III.2)*. Additionally, it adheres to the condition in* (III.24)*. Then, $\mathcal{C}$ can simultaneously correct $\delta - 1 + \nu_j$ erasures, $\nu_j \geq 0$, in each local code $\mathcal{C}|_{\mathcal{S}_j}$ provided that the number of global parities available is at least $\nu_1 + \cdots + \nu_{L_c}$.*

Proof. We begin by defining $\boldsymbol{H}^{\mathcal{C}}|_{\mathcal{J}}^{\mathcal{I}}$ as the submatrix of $\boldsymbol{H}^{\mathcal{C}}$ restricted in columns by the set $\mathcal{J}$ and in rows by the set $\mathcal{I}$. For $j \in \mathbb{N}_{L_c}$, consider the j-th local code. Let $\mathcal{E}_j$ denote the set of coordinates that are erased in the j-th local code, where $|\mathcal{E}_j| = \delta - 1 + \nu_j$. Let

$$\mathcal{R}_j = \{(\delta - 1)(j - 1) + 1, \ldots, (\delta - 1)j\} \cup \mathcal{A}_j$$

be a set of rows of $\boldsymbol{H}^{\mathcal{C}}$ of cardinality $|\mathcal{R}_j| = |\mathcal{E}_j|$, where $\mathcal{A}_j \subset \mathbb{N}_{L_c(\delta-1)+1:(n-k)}$, $\left|\mathcal{A}_j\right| = \nu_j$, is a set of rows of $\boldsymbol{H}^{\mathcal{C}}$ (which correspond to parity-check equations of the available global parities). In order to prove the lemma one needs to prove that

$$\operatorname{rank}\left(\boldsymbol{H}^{\mathcal{C}}\big|_{\cup_j \mathcal{E}_j}^{\cup_j \mathcal{R}_j}\right) = \sum_{j=1}^{L_c} (\delta - 1 + \nu_j). \tag{III.43}$$

For each $j \in \mathbb{N}_{L_c}$ and $j' \neq j$, assume that there exists a set $\mathcal{A}_{j'} \subset \mathbb{N}_{L_c(\delta-1)+1:(n-k)}$ such that $\mathcal{A}_j \cap \mathcal{A}_{j'} = \emptyset$. Then, it follows that $\mathcal{R}_j \cap \mathcal{R}_{j'} = \emptyset$, and since $\mathcal{E}_j \cap \mathcal{E}_{j'} = \emptyset$,

$$\operatorname{rank}\left(\boldsymbol{H}^{\mathcal{C}}\big|_{\cup_j \mathcal{E}_j}^{\cup_i \mathcal{R}_j}\right) = \sum_{j=1}^{L_c} \operatorname{rank}\left(\boldsymbol{H}^{\mathcal{C}}|_{\mathcal{E}_j}^{\mathcal{R}_j}\right).$$

Thus, to show (III.43) it is sufficient to show that

$$\operatorname{rank}\left(\boldsymbol{H}^{\mathcal{C}}|_{\mathcal{E}_j}^{\mathcal{R}_j}\right) = \delta - 1 + \nu_j \tag{III.44}$$

for all $j \in \mathbb{N}_{L_c}$.

To show this, consider now the $[n', k]$ MDS code $\mathcal{C}'$ whose parity-check matrix is given by $\boldsymbol{H}^{\mathsf{MDS}}$ in(III.24). Let $\mathcal{S}'_j \subset \mathbb{N}_{n'}$ denote a set of coordinates of $\mathcal{C}'$ of cardinality $\left|\mathcal{S}'_j\right| = k + \delta - 1 + \nu_j$. More specifically,

$$\mathcal{S}'_j = \{1, \ldots, k\} \cup \{k + 1, \ldots, k + \delta - 1\} \cup \mathcal{B}_j,$$

where $\mathcal{B}_j = \{a - L_c(\delta - 1) + (\delta - 1) + k \colon a \in \mathcal{A}_j\} \subset \mathbb{N}_{\delta+k:n'}$. In other words, the set $\mathcal{S}'_j$ consists of k systematic coordinates and $\delta - 1 + \nu_j$ parity coordinates of $\mathcal{C}'$. The punctured code $\mathcal{C}'_j = \mathcal{C}'|_{\mathcal{S}'_j}$ is defined by a parity-check matrix $\boldsymbol{H}^{\mathcal{C}'_j}$ of dimensions $(\delta - 1 + \nu_j) \times (k + \delta - 1 + \nu_j)$ that is a submatrix of $\boldsymbol{H}^{\mathsf{MDS}}$. Since the punctured code of an MDS code is also an MDS code [41], $\mathcal{C}'_j$ has minimum

Hamming distance $d_{\min}^{\mathcal{C}'_j} = \delta + \nu_j = \delta + |\mathcal{A}_j|$. Note that for some column index set $\mathcal{J} \subset \mathbb{N}_{k+\delta-1+\nu_j}$, $|\mathcal{J}| = |\mathcal{E}_j|$, one can build $\boldsymbol{H}^{\mathcal{C}}|_{\mathcal{E}_j}^{\mathcal{R}_j} = \boldsymbol{H}^{\mathcal{C}'_j}|_{\mathcal{J}}$. From the MDS property, it follows that

$$\operatorname{rank}\left(\boldsymbol{H}^{\mathcal{C}}|_{\mathcal{E}_j}^{\mathcal{R}_j}\right) = \operatorname{rank}\left(\boldsymbol{H}^{\mathcal{C}'_j}|_{\mathcal{J}}\right) = \delta - 1 + \nu_j.$$

Finally, if the total number of global parities is at least $\sum_{j=1}^{L_c} \nu_j$, we can assign to the set $\mathcal{A}_j$, $j \in \mathbb{N}_{L_c}$, a set of ν_j rows of $\boldsymbol{H}^{\mathcal{C}}$ corresponding to global parity-checks such that the sets $\mathcal{A}_j$ are all disjoint, hence (III.44) holds for all $j \in \mathbb{N}_{L_c}$, and (III.43) follows, which completes the proof. □

Lemma 8. *Consider an erasure pattern $\boldsymbol{e}$ of length n of the form*

$$\boldsymbol{e} = (e_1, \ldots, e_n) = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_L, \boldsymbol{e}_{L+1}),$$

where the subvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_L$ are all of length $n_c = r + \delta - 1$ and $\boldsymbol{e}_{L+1}$ is of length $\bar{r} = n \bmod n_c$. Let $\chi(\boldsymbol{e}_j)$, $j \in \mathbb{N}_{L+1}$, be the support of $\boldsymbol{e}_j$ and $t = (n-k) \bmod L$. If $|\chi(\boldsymbol{e}_1)| = \cdots = |\chi(\boldsymbol{e}_t)| = m_1$, $|\chi(\boldsymbol{e}_{t+1})| = \cdots = |\chi(\boldsymbol{e}_L)| = m$, and $|\chi(\boldsymbol{e}_{L+1})| = 0$, where $m = \left\lfloor \frac{n-k}{L} \right\rfloor$ and $m_1 = m + 1$, then $\boldsymbol{e}$ is correctable by $\mathcal{C}$.

Proof. The erasure pattern $\boldsymbol{e}$ is divided into $L+1$ partitions represented by $\boldsymbol{e}_j = (e_{n_c(j-1)+1}, \ldots, e_{n_c j})$, $j \in \mathbb{N}_{L+1}$, where $\boldsymbol{e}_j$, $j \in \mathbb{N}_{L_c}$, corresponds to the coordinates of the j-th local code, and $\boldsymbol{e}_{L_c+1}, \ldots, \boldsymbol{e}_{L+1}$ correspond to the coordinates of the global parities of $\mathcal{C}$.

The set $\chi(\boldsymbol{e}_j)$, $j \in \mathbb{N}_{L+1}$, is the set of coordinates erased from the j-th partition, and we construct the erasure patterns $\boldsymbol{e}_j$, $j \in \mathbb{N}_L$, such that $|\chi(\boldsymbol{e}_j)| = \delta - 1 + \nu_j$ with

$$\nu_j = \begin{cases} m_1 - (\delta - 1) & \text{if } j \in \mathbb{N}_t, \\ m - (\delta - 1) & \text{if } j \in \mathbb{N}_{t+1:L}, \end{cases}$$

where $t = (n-k) \bmod L$, and let $\chi(\boldsymbol{e}_{L+1}) = \emptyset$. In other words, we construct the erasure patterns such that the erasures are distributed as equally as possible across the first L partitions.

From Definition 7, it follows that $n-k \geq (\delta-1)L_c + (L-L_c)(r+\delta-1) \geq L(\delta-1)$ (where the last inequality follows from $L \geq L_c$), hence $\delta - 1$ is an integer satisfying the inequality $L(\delta-1) \leq n-k$, and subsequently $\delta - 1 \leq \frac{n-k}{L}$. The integer m is the largest integer such that $m \leq \frac{n-k}{L}$. Therefore, $\delta - 1 \leq m$. To show that $\boldsymbol{e}$ is correctable it is enough to show that the erasures in the L_c local codes can be corrected, since in this case we have a nonerased information set for $\mathcal{C}$, which allows to correct the remaining erasures in $\boldsymbol{e}$.

From Lemma 7, to correct $\delta - 1 + \nu_j$ erasures in the j-th local code for all $j \in \mathbb{N}_{L_c}$, the number of global parities available, $\gamma_{\text{tot}} + \bar{r}$, must be

$$\gamma_{\text{tot}} + \bar{r} \geq \sum_{j=1}^{L_c} \nu_j = \begin{cases} m_1 t + m(L_c - t) - L_c(\delta - 1) & \text{if } t \leq L_c, \\ m_1 L_c - L_c(\delta - 1) & \text{if } t > L_c, \end{cases} \tag{III.45}$$

where γ_{tot} is the number of global parities available in the $(L_{\text{c}}+1)$-th, ..., L-th partitions and $\bar{r} = n - n_{\text{c}}L$ is the number of global parities in the $(L+1)$-th partition. By counting the number of global parities not erased in $L-L_{\text{c}}$ partitions, we get

$$\gamma_{\text{tot}} = \begin{cases} (n_{\text{c}} - m)(L - L_{\text{c}}) & \text{if } t \leq L_{\text{c}}, \\ (n_{\text{c}} - m_1)(t - L_{\text{c}}) + (n_{\text{c}} - m)(L - t) & \text{if } t > L_{\text{c}}. \end{cases} \tag{III.46}$$

By substituting (III.46) into (III.45), we get (after performing some simple arithmetic) the condition

$$n - k - mL \geq t,$$

which is valid for both cases of t ($t \leq L_{\text{c}}$ and $t > L_{\text{c}}$). By definition of t and m, the above inequality is met with equality, and it follows that $\boldsymbol{e}$ is a correctable erasure pattern. □

F.1 Proof of Step a)

Let $\tilde{\boldsymbol{E}}$, $\boldsymbol{W}$, $\boldsymbol{Z}$, and $\boldsymbol{O}$ be submatrices of $\boldsymbol{E}$ as shown in (III.25). We begin the proof by proving that each of the $n_{\text{c}}L$ rows of the matrix $\left(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z}\right)$ are correctable erasure patterns, where $\tilde{\boldsymbol{E}}$ is defined in (III.26). This is proved by induction on the row partitions of $\left(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z}\right)$.

Base Case. Consider the first row partition of $\left(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z}\right)$, given by

$$\left(\boldsymbol{\pi}_1 \quad \boldsymbol{\pi}_2 \quad \cdots \quad \boldsymbol{\pi}_L \quad \mathbf{0}_{n_{\text{c}} \times \bar{r}}\right).$$

For each row vector $\boldsymbol{e}_i^{(1)}$, $i \in \mathbb{N}_{n_{\text{c}}}$, in this row partition, where the subscript i indicates the row index and the superscript the row partition, consider the subvectors $\boldsymbol{e}_{i,1}^{(1)}, \ldots, \boldsymbol{e}_{i,L}^{(1)}$. From Step a) in Section 6.3, for all $i \in \mathbb{N}_{n_{\text{c}}}$, the j-th subvectors $\boldsymbol{e}_{i,j}^{(1)}$ have support of cardinality $|\chi(\boldsymbol{e}_{i,j}^{(1)})| = m_1$ for all $j \in \mathbb{N}_t$, where $t = (n-k) \bmod L$, $|\chi(\boldsymbol{e}_{i,j}^{(1)})| = m$ for $j \in \mathbb{N}_{t+1:L}$, and $|\chi(\boldsymbol{e}_{i,L+1}^{(1)})| = 0$. Thus, the vectors $\boldsymbol{e}_i^{(1)}$ in the first row partition of $\left(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z}\right)$ have the same structure as the erasure pattern $\boldsymbol{e}$ from Lemma 8 and are therefore erasure patterns that are correctable by $\mathcal{C}$. Note that the number of global parities available in the $(L_{\text{c}}+1)$-th, ..., L-th subvectors of vector $\boldsymbol{e}_i^{(1)}$, $\gamma_{\text{tot}}^{(1)}$, is $\gamma_{\text{tot}}^{(1)} = \gamma_{\text{tot}}$, hence $\gamma_{\text{tot}}^{(1)} + \bar{r} = \gamma_{\text{tot}} + \bar{r} \geq \sum_{j=1}^{L_{\text{c}}} \nu_j$ and from the proof of Lemma 8 the error pattern $\boldsymbol{e}_i^{(1)}$ is correctable.

Inductive Step. Assume that the vectors $\boldsymbol{e}_i^{(l)}$, $i \in \mathbb{N}_{n_{\text{c}}}$, in the l-th row partition of $\left(\tilde{\boldsymbol{E}} | \boldsymbol{Z}\right)$ are correctable by $\mathcal{C}$ and that each local code $\mathcal{C}|_{\mathcal{S}_j}$ can correct $\delta - 1 + \nu_j^{(l)}$ erasures, $j \in \mathbb{N}_{L_{\text{c}}}$. The row vectors are taken from the matrix

$$\left(\boldsymbol{\pi}_{\sigma^{l-1}(1)} \quad \boldsymbol{\pi}_{\sigma^{l-1}(2)} \quad \cdots \quad \boldsymbol{\pi}_{\sigma^{l-1}(L)} \quad \mathbf{0}_{n_{\text{c}} \times \bar{r}}\right),$$

where $\sigma \triangleq (L\,(L-1) \cdots 1)$ denotes a *cycle* whose mapping is $L \mapsto (L-1) \mapsto \cdots \mapsto 1 \mapsto L$. The $(L+1)$-th subvectors satisfy $|\chi(\boldsymbol{e}_{i,L+1}^{(l)})| = 0$. From Lemma 7, the

underlying characteristic of the vectors $\boldsymbol{e}_i^{(l)}$ is that they are correctable erasure patterns if the number of global parities not erased in $\boldsymbol{e}_i^{(l)}$, $\gamma_{\text{tot}}^{(l)} + \bar{r}$, is larger than or equal to $\sum_{j=1}^{L_c} \nu_j^{(l)}$.

In the $(l+1)$-th row partition of $\left(\tilde{\boldsymbol{E}}|\boldsymbol{Z}\right)$, the n_c rows have the form

$$\left(\boldsymbol{\pi}_{\sigma^l(1)} \quad \boldsymbol{\pi}_{\sigma^l(2)} \quad \cdots \quad \boldsymbol{\pi}_{\sigma^l(L)} \quad \mathbf{0}_{n_c \times \bar{r}}\right).$$

Due to the cyclic shifts, for $j \in \mathbb{N}_L$, all the j-th subvectors of the vectors $\boldsymbol{e}_i^{(l+1)}$ in row partition $l+1$, $l \in \mathbb{N}_{L-1}$, have support size $|\chi(\boldsymbol{e}_{i,j}^{(l+1)})| = |\chi(\boldsymbol{e}_{i,\sigma^{-1}(j)}^{(l)})|$. Thus, there exist two indices $j', j'' \in \mathbb{N}_L$, $j' \neq j''$, such that

$$\begin{aligned} |\chi(\boldsymbol{e}_{i,j'}^{(l+1)})| - |\chi(\boldsymbol{e}_{i,j'}^{(l)})| &= |\chi(\boldsymbol{e}_{i,j''}^{(l)})| - |\chi(\boldsymbol{e}_{i,j''}^{(l+1)})|, \\ |\chi(\boldsymbol{e}_{i,j}^{(l+1)})| &= |\chi(\boldsymbol{e}_{i,j}^{(l)})|, \quad \forall j \in \mathbb{N}_L \setminus \{j', j''\}. \end{aligned} \tag{III.47}$$

One can see that there are at most 4 (depending on t and L_c) choices for the pair (j', j'') as follows.

Case 1. $j', j'' \in \mathbb{N}_{L_c}$: From (III.47), it follows that $\nu_{j'}^{(l+1)} - \nu_{j'}^{(l)} = \nu_{j''}^{(l)} - \nu_{j''}^{(l+1)}$, $\nu_j^{(l+1)} = \nu_j^{(l)}$, and $\gamma_{\text{tot}}^{(l)} = \gamma_{\text{tot}}^{(l+1)}$. Thus, we have $\sum_{j=1}^{L_c} \nu_j^{(l+1)} = \sum_{j=1}^{L_c} \nu_j^{(l)} = \gamma_{\text{tot}}^{(l)} + \bar{r} = \gamma_{\text{tot}}^{(l+1)} + \bar{r}$.

Case 2. $j', j'' \in \mathbb{N}_{L_c+1:L}$: From (III.47), it follows that $\gamma_{\text{tot}}^{(l+1)} = \gamma_{\text{tot}}^{(l)}$ and $\sum_{j=1}^{L_c} \nu_j^{(l+1)} = \sum_{j=1}^{L_c} \nu_j^{(l)}$. Therefore, $\sum_{j=1}^{L_c} \nu_j^{(l+1)} = \gamma_{\text{tot}}^{(l+1)} + \bar{r}$.

Case 3. $j' \in \mathbb{N}_{L_c}$, $j'' \in \mathbb{N}_{L_c+1:L}$: From (III.47), it follows that $\nu_{j'}^{(l+1)} - \nu_{j'}^{(l)} = \gamma_{\text{tot}}^{(l+1)} - \gamma_{\text{tot}}^{(l)}$. Moreover, it can be seen that $\sum_{j \neq j', j \in \mathbb{N}_{L_c}} \nu_j^{(l+1)} = \sum_{j \neq j', j \in \mathbb{N}_{L_c}} \nu_j^{(l)}$. Hence, we have

$$\begin{aligned} \sum_{j=1}^{L_c} \nu_j^{(l+1)} &= \sum_{j \neq j', j \in \mathbb{N}_{L_c}} \nu_j^{(l+1)} + \nu_{j'}^{(l+1)} \\ &= \sum_{j \neq j', j \in \mathbb{N}_{L_c}} \nu_j^{(l+1)} + (\nu_{j'}^{(l+1)} - \nu_{j'}^{(l)}) + \nu_{j'}^{(l)} \\ &= \sum_{j \neq j', j \in \mathbb{N}_{L_c}} \nu_j^{(l)} + (\gamma_{\text{tot}}^{(l+1)} - \gamma_{\text{tot}}^{(l)}) + \nu_{j'}^{(l)} \\ &= \sum_{j=1}^{L_c} \nu_j^{(l)} + (\gamma_{\text{tot}}^{(l+1)} - \gamma_{\text{tot}}^{(l)}) \\ &\overset{(b)}{=} \gamma_{\text{tot}}^{(l)} + \bar{r} + (\gamma_{\text{tot}}^{(l+1)} - \gamma_{\text{tot}}^{(l)}) \\ &= \gamma_{\text{tot}}^{(l+1)} + \bar{r}, \end{aligned}$$

where (b) holds since $\sum_{j=1}^{L_c} \nu_j^{(l)} = \gamma_{\text{tot}}^{(l)} + \bar{r}$.

Case 4. $j' \in \mathbb{N}_{L_c+1:L}$, $j'' \in \mathbb{N}_{L_c}$: Following an argumentation similar to Case 3, we have $\sum_{j=1}^{L_c} \nu_j^{(l+1)} = \gamma_{\text{tot}}^{(l+1)} + \bar{r}$.

In each of the above cases we see that the condition $\gamma_{\text{tot}} + \bar{r} \geq \sum_{j=1}^{L_c} \nu_j^{(l+1)}$ is satisfied (with equality). From the proof of Lemma 8, the n_c rows in the $(l+1)$-th row partition of $\left(\tilde{\boldsymbol{E}}|\boldsymbol{Z}\right)$ are correctable by $\mathcal{C}$, which completes the inductive step.

The rows of $(\boldsymbol{W} \mid \boldsymbol{O})$ as shown in Step a) in Section 6.3 have support corresponding to only the parity symbols of $\mathcal{C}$. Thus, these rows are all correctable by $\mathcal{C}$, and it follows from the above arguments that each row of $\boldsymbol{E}$ is an erasure pattern that is correctable by $\mathcal{C}$.

F.2 Proof of Step b)

We now address the second part of the proof. Note that the column coordinates in $\mathcal{P}_j$, $j \in \mathbb{N}_L$, have column weight $n - k + \bar{r}$ after Step a). Step b) involves the swapping of one entries from these coordinates with zero entries in the column coordinates of $\boldsymbol{Z}$. The swapping is done to ensure that the column weight of the columns in $\mathcal{P}_j$, $j \in \mathbb{N}_L$, is reduced to $n - k$, while those of the columns of $\boldsymbol{Z}$ are increased to $n - k - \bar{r}$. Since $\boldsymbol{O}$ is an all-one matrix, the columns of $\boldsymbol{E}$ in $\mathcal{P}_{L+1}$ have also weight $n - k$. It is possible to show that such a swapping always exists. Overall, the resulting matrix $\boldsymbol{E}$ is $(n-k)$-column regular. To ensure that the erasure patterns are correctable, we use Lemma 7. For each row, we need to satisfy

$$\sum_{j=1}^{L_c} \nu_j \leq \gamma_{\text{tot}} + \bar{r}. \tag{III.48}$$

Clearly, if for a certain row of $(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z})$ a one from a column from a column partition in $\mathbb{N}_{L_c+1:L}$ (corresponding to $\tilde{\boldsymbol{E}}$) is swapped with a zero in a column from partition $L+1$ (corresponding to $\boldsymbol{Z}$), then the resulting erasure pattern is still correctable by $\mathcal{C}$ as (III.48) is still valid. On the other hand, for $j \in \mathbb{N}_c$, if for a certain row of $(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z})$ a one from the j-th column partition is swapped with a zero in the $(L+1)$-th column partition, then such a row is still a correctable erasure pattern provided that $\nu_j > 0$ before the swap. This is easy to see as the swapping procedure reduces ν_j by one and reduces $\bar{r}$ by one. Thus, (III.48) is still satisfied. From the aforementioned arguments and the fact that each row of any row partition of $(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z})$ has at most $\bar{r}$ swaps of ones occurring from the set of $\mathbb{N}_L$ column partitions and zeroes from the $(L+1)$-th partition, valid swaps are possible if and only if

$$\sum_{j=1}^{L_c} \nu_j + \sum_{j=L_c+1}^{L} \rho_j \geq \bar{r}. \tag{III.49}$$

As $\nu_j = \rho_j - (\delta - 1)$, the left hand side of (III.49) can be lowerbounded as

$$\sum_{j=1}^{L_c} \nu_j + \sum_{j=L_c+1}^{L} \rho_j = \sum_{j=1}^{L} \rho_j - L_c(\delta - 1) \geq \sum_{j=1}^{L} \rho_j - L(\delta - 1) = n - k - L(\delta - 1). \tag{III.50}$$

From (III.50), $k = L_c r$, and $\bar{r} = n - L(r + \delta - 1)$, (III.49) reduces to $L \geq L_c$. By definition, this is always true. Thus, there exist $\bar{r}$ valid swaps from each row of each row partition of $(\tilde{\boldsymbol{E}} \mid \boldsymbol{Z})$ that ensure that the resulting erasure patterns are still correctable by $\mathcal{C}$.

Each of the $\bar{r}$ iterations of the systematic procedure in Step b) assumes that in the first row partition there exists a valid swap. Over all iterations, this means that there should exist at most $\bar{r}$ valid swaps in any row of the row partition. From the discussion above, this is clearly true. This means that all rows in this partition are correctable erasure patterns. For the remainder row partitions, the column indices from where ones are swapped correspond to cyclic shifts of the indices from the first row partition. From (III.26), it is seen that $\tilde{\boldsymbol{E}}$ has a cyclic structure. Therefore, the swaps in the i-th row partition ensure that each row is still a correctable erasure pattern. This completes the proof.

G Proof of Theorem 7

To prove the theorem we need the following lemma.

Lemma 9. *Let $\mathcal{C}$ be an $[n = 2n_1, k = k_1 + 1]$ binary code constructed from an $[n_1, k_1]$ code $\mathcal{U}$ through the $(\mathcal{U} \mid \mathcal{U} + \mathcal{V})$ construction, where $\mathcal{V}$ is an $[n, 1]$ binary repetition code. The generator matrix $\boldsymbol{G}^{\mathcal{C}}$ of $\mathcal{C}$ is given in* (III.36)*. Let $\bar{\mathcal{C}} = \mathcal{C}$ and $\boldsymbol{G}^{\bar{\mathcal{C}}} = \boldsymbol{G}^{\mathcal{C}}$. Then, the code $\tilde{\mathcal{C}} = \mathcal{C} \circ \bar{\mathcal{C}}$ is a vector space of dimension*

$$\dim(\tilde{\mathcal{C}}) \leq \begin{cases} k_1 + n_1 + 1 & \text{if } n_1 - k_1 \leq \binom{k_1}{2}, \\ 2k_1 + \binom{k_1}{2} + 1 & \text{otherwise.} \end{cases} \tag{III.51}$$

Proof. From Definition 4, we know that $\tilde{\boldsymbol{c}} \in \tilde{\mathcal{C}}$ has the form $\tilde{\boldsymbol{c}} = (c_1\bar{c}_1, \ldots, c_n\bar{c}_n)$, where $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ and $\bar{\boldsymbol{c}} = (\bar{c}_1, \ldots, \bar{c}_n) \in \bar{\mathcal{C}}$. Considering $\boldsymbol{G}^{\mathcal{C}} = (g^{c}_{i,j})$ and $\boldsymbol{G}^{\bar{\mathcal{C}}} = (g^{\bar{c}}_{i,j})$, the vector space $\tilde{\mathcal{C}}$ is spanned by the row space of

$$\boldsymbol{G}^{\tilde{\mathcal{C}}} = \begin{pmatrix} g^{c}_{1,1}\boldsymbol{g}^{\bar{c}}_1 & g^{c}_{1,2}\boldsymbol{g}^{\bar{c}}_2 & \cdots & g^{c}_{1,n}\boldsymbol{g}^{\bar{c}}_n \\ g^{c}_{2,1}\boldsymbol{g}^{\bar{c}}_1 & g^{c}_{2,2}\boldsymbol{g}^{\bar{c}}_2 & \cdots & g^{c}_{2,n}\boldsymbol{g}^{\bar{c}}_n \\ \vdots & \vdots & \cdots & \vdots \\ g^{c}_{k,1}\boldsymbol{g}^{\bar{c}}_1 & g^{c}_{k,2}\boldsymbol{g}^{\bar{c}}_2 & \cdots & g^{c}_{k,n}\boldsymbol{g}^{\bar{c}}_n \end{pmatrix}, \tag{III.52}$$

where the vector $\boldsymbol{g}^{\bar{c}}_j$, $j \in \mathbb{N}_n$, denotes the j-th column vector of $\boldsymbol{G}^{\bar{\mathcal{C}}}$. The matrix $\boldsymbol{G}^{\tilde{\mathcal{C}}}$ is a matrix consisting of k^2 row vectors (corresponding to codewords of $\tilde{\mathcal{C}}$) of length n. We divide $\boldsymbol{G}^{\tilde{\mathcal{C}}}$ into k submatrices $\boldsymbol{G}^{\tilde{\mathcal{C}}}_i$, where $\boldsymbol{G}^{\tilde{\mathcal{C}}}_i = (g^{c}_{i,1}\boldsymbol{g}^{\bar{c}}_1 | g^{c}_{i,2}\boldsymbol{g}^{\bar{c}}_2 | \ldots | g^{c}_{i,n}\boldsymbol{g}^{\bar{c}}_n)$, $i \in \mathbb{N}_k$ (see (III.52)). From (III.36) and since $\boldsymbol{G}^{\mathcal{C}} = \boldsymbol{G}^{\bar{\mathcal{C}}}$,

we have $g^{c}_{k,j} = g^{\tilde{c}}_{k,j} = 0$, $j \in \mathbb{N}_{n_1}$, and $g^{c}_{k,j} = g^{\tilde{c}}_{k,j} = 1$, $j \in \mathbb{N}_{n_1+1:n}$. Therefore, (III.52) can be expanded to

$$\boldsymbol{G}^{\tilde{\mathcal{C}}} = \begin{pmatrix} g^{c}_{1,1}\begin{pmatrix} g^{\tilde{c}}_{1,1} \\ g^{\tilde{c}}_{2,1} \\ \vdots \\ g^{\tilde{c}}_{k_1,1} \\ 0 \end{pmatrix} & \cdots & g^{c}_{1,n_1}\begin{pmatrix} g^{\tilde{c}}_{1,n_1} \\ g^{\tilde{c}}_{2,n_1} \\ \vdots \\ g^{\tilde{c}}_{k_1,n_1} \\ 0 \end{pmatrix} & g^{c}_{1,n_1+1}\begin{pmatrix} g^{\tilde{c}}_{1,n_1+1} \\ g^{\tilde{c}}_{2,n_1+1} \\ \vdots \\ g^{\tilde{c}}_{k_1,n_1+1} \\ 1 \end{pmatrix} & \cdots & g^{c}_{1,n}\begin{pmatrix} g^{\tilde{c}}_{1,n} \\ g^{\tilde{c}}_{2,n} \\ \vdots \\ g^{\tilde{c}}_{k_1,n} \\ 1 \end{pmatrix} \\ g^{c}_{2,1}\begin{pmatrix} g^{\tilde{c}}_{1,1} \\ g^{\tilde{c}}_{2,1} \\ \vdots \\ g^{\tilde{c}}_{k_1,1} \\ 0 \end{pmatrix} & \cdots & g^{c}_{2,n_1}\begin{pmatrix} g^{\tilde{c}}_{1,n_1} \\ g^{\tilde{c}}_{2,n_1} \\ \vdots \\ g^{\tilde{c}}_{k_1,n_1} \\ 0 \end{pmatrix} & g^{c}_{2,n_1+1}\begin{pmatrix} g^{\tilde{c}}_{1,n_1+1} \\ g^{\tilde{c}}_{2,n_1+1} \\ \vdots \\ g^{\tilde{c}}_{k_1,n_1+1} \\ 1 \end{pmatrix} & \cdots & g^{c}_{2,n}\begin{pmatrix} g^{\tilde{c}}_{1,n} \\ g^{\tilde{c}}_{2,n} \\ \vdots \\ g^{\tilde{c}}_{k_1,n} \\ 1 \end{pmatrix} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0\begin{pmatrix} g^{\tilde{c}}_{1,1} \\ g^{\tilde{c}}_{2,1} \\ \vdots \\ g^{\tilde{c}}_{k_1,1} \\ 0 \end{pmatrix} & \cdots & 0\begin{pmatrix} g^{\tilde{c}}_{1,n_1} \\ g^{\tilde{c}}_{2,n_1} \\ \vdots \\ g^{\tilde{c}}_{k_1,n_1} \\ 0 \end{pmatrix} & 1\begin{pmatrix} g^{\tilde{c}}_{1,n_1+1} \\ g^{\tilde{c}}_{2,n_1+1} \\ \vdots \\ g^{\tilde{c}}_{k_1,n_1+1} \\ 1 \end{pmatrix} & \cdots & 1\begin{pmatrix} g^{\tilde{c}}_{1,n} \\ g^{\tilde{c}}_{2,n} \\ \vdots \\ g^{\tilde{c}}_{k_1,n} \\ 1 \end{pmatrix} \end{pmatrix}. \tag{III.53}$$

Furthermore, let $\boldsymbol{G}^{\mathcal{U}} = (g^{u}_{i,j})$ be the generator matrix of $\mathcal{U}$. From (III.36), we have $g^{c}_{i,j} = g^{\tilde{c}}_{i,j} = g^{u}_{i,j}$ for $i \in \mathbb{N}_{k_1}$ and $j \in \mathbb{N}_{n_1}$. For $i, j \in \mathbb{N}_k$, we denote the i-th row of the j-th submatrix $\boldsymbol{G}^{\tilde{c}}_{j}$ as $\boldsymbol{w}^{(j)}_{i}$. For $i \in \mathbb{N}_{k-1}$, the i-th row of the i-th submatrix $\boldsymbol{G}^{\tilde{c}}_{i}$ is given as

$$\boldsymbol{w}^{(i)}_{i} = (g^{c}_{i,1}g^{\tilde{c}}_{i,1}, g^{c}_{i,2}g^{\tilde{c}}_{i,2}, \dots, g^{c}_{i,n}g^{\tilde{c}}_{i,n}). \tag{III.54}$$

Since $g^{\tilde{c}}_{i,j} = g^{c}_{i,j} \in \mathrm{GF}(2)$, (III.54) reduces to $\boldsymbol{w}^{(i)}_{i} = (g^{c}_{i,1}, g^{c}_{i,2}, \dots, g^{c}_{i,n})$. Furthermore, from (III.36) we see that $g^{c}_{i,j} = g^{c}_{i,n_1+j} = g^{u}_{i,j}$, $j \in \mathbb{N}_{n_1}$, $i \in \mathbb{N}_{k_1}$. Therefore, these $k_1 = k - 1$ rows form the k_1 basis vectors of the code space $(\mathcal{U}, \mathcal{U})$ and can be arranged in a matrix as

$$\begin{pmatrix} \boldsymbol{G}^{\mathcal{U}} & \boldsymbol{G}^{\mathcal{U}} \end{pmatrix}. \tag{III.55}$$

The k-th row of $\boldsymbol{G}^{\tilde{c}}_{i}$ can be written as

$$\begin{aligned} \boldsymbol{w}^{(i)}_{k} &= (\overbrace{0, 0, \dots, 0}^{n_1}, g^{c}_{i,n_1+1}, g^{c}_{i,n_1+2}, \dots, g^{c}_{i,n}) \\ &\overset{(c)}{=} (0, 0, \dots, 0, g^{u}_{i,1}, g^{u}_{i,2}, \dots, g^{u}_{i,n_1}), \end{aligned}$$

where (c) results from the structure of $\boldsymbol{G}^{\mathcal{C}}$ in (III.36). Stacking together the k-th row of all k_1 submatrices $\boldsymbol{G}^{\tilde{c}}_{i}$, $i \in \mathbb{N}_{k_1}$, results in the k_1 row vectors

$$\begin{pmatrix} \boldsymbol{0}_{k_1 \times n_1} & \boldsymbol{G}^{\mathcal{U}} \end{pmatrix}. \tag{III.56}$$

In a similar way, the rows $\boldsymbol{w}_i^{(k)}$, $i \in \mathbb{N}_k$, of the k-th submatrix $\boldsymbol{G}_k^{\bar{\mathcal{C}}}$ result in the matrix

$$\begin{pmatrix} \mathbf{0}_{k_1 \times n_1} & \boldsymbol{G}^{\mathcal{U}} \\ \mathbf{0}_{1 \times n_1} & \mathbf{1}_{1 \times n_1} \end{pmatrix}. \tag{III.57}$$

Of the remaining $(k-1)(k-2)$ rows in (III.53), since $\mathcal{C} = \bar{\mathcal{C}}$, there exist $\binom{k_1}{2}$ distinct rows as follows,

$$\boldsymbol{\Theta} = \begin{pmatrix} g_{1,1}^{\mathcal{C}} g_{2,1}^{\bar{\mathcal{C}}} & g_{1,2}^{\mathcal{C}} g_{2,2}^{\bar{\mathcal{C}}} & \cdots & g_{1,n}^{\mathcal{C}} g_{2,n}^{\bar{\mathcal{C}}} \\ g_{1,1}^{\mathcal{C}} g_{3,1}^{\bar{\mathcal{C}}} & g_{1,2}^{\mathcal{C}} g_{3,2}^{\bar{\mathcal{C}}} & \cdots & g_{1,n}^{\mathcal{C}} g_{3,n}^{\bar{\mathcal{C}}} \\ \vdots & \vdots & \cdots & \vdots \\ g_{1,1}^{\mathcal{C}} g_{k_1,1}^{\bar{\mathcal{C}}} & g_{1,2}^{\mathcal{C}} g_{k_1,2}^{\bar{\mathcal{C}}} & \cdots & g_{1,n}^{\mathcal{C}} g_{k_1,n}^{\bar{\mathcal{C}}} \\ g_{2,1}^{\mathcal{C}} g_{3,1}^{\bar{\mathcal{C}}} & g_{2,2}^{\mathcal{C}} g_{3,2}^{\bar{\mathcal{C}}} & \cdots & g_{2,n}^{\mathcal{C}} g_{3,n}^{\bar{\mathcal{C}}} \\ \vdots & \vdots & \cdots & \vdots \\ g_{2,1}^{\mathcal{C}} g_{k_1,1}^{\bar{\mathcal{C}}} & g_{2,2}^{\mathcal{C}} g_{k_1,2}^{\bar{\mathcal{C}}} & \cdots & g_{2,n}^{\mathcal{C}} g_{k_1,n}^{\bar{\mathcal{C}}} \\ \vdots & \vdots & \cdots & \vdots \\ g_{k_1-1,1}^{\mathcal{C}} g_{k_1,1}^{\bar{\mathcal{C}}} & g_{k_1-1,2}^{\mathcal{C}} g_{k_1,2}^{\bar{\mathcal{C}}} & \cdots & g_{k_1-1,n}^{\mathcal{C}} g_{k_1,n}^{\bar{\mathcal{C}}} \end{pmatrix}.$$

Furthermore, from the construction of $\mathcal{C}$ in (III.36), we have $(g_{i,1}^{\mathcal{C}}, \ldots, g_{i,n_1}^{\mathcal{C}}) = (g_{i,n_1+1}^{\mathcal{C}}, \ldots, g_{i,n}^{\mathcal{C}}) = (g_{i,1}^{\mathcal{U}}, \ldots, g_{i,n_1}^{\mathcal{U}})$, $i \in \mathbb{N}_{k_1}$ and because $\bar{\mathcal{C}} = \mathcal{C}$, we have $(g_{i,1}^{\bar{\mathcal{C}}}, \ldots, g_{i,n_1}^{\bar{\mathcal{C}}}) = (g_{i,n_1+1}^{\bar{\mathcal{C}}}, \ldots, g_{i,n}^{\bar{\mathcal{C}}})$. Therefore,

$$\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} & \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} \end{pmatrix}, \tag{III.58}$$

where $\boldsymbol{\theta}_{\binom{k_1}{2} \times n_1}$ is a binary matrix of size $\binom{k_1}{2} \times n_1$. From (III.55)–(III.58), $\boldsymbol{G}^{\bar{\mathcal{C}}}$ can be written as

$$\boldsymbol{G}^{\bar{\mathcal{C}}} = \begin{pmatrix} \boldsymbol{G}^{\mathcal{U}} & \boldsymbol{G}^{\mathcal{U}} \\ \mathbf{0}_{k_1 \times n_1} & \boldsymbol{G}^{\mathcal{U}} \\ \mathbf{0}_{k_1 \times n_1} & \boldsymbol{G}^{\mathcal{U}} \\ \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} & \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} \\ \mathbf{0}_{1 \times n_1} & \mathbf{1}_{1 \times n_1} \end{pmatrix}.$$

Using Gaussian elimination, $\boldsymbol{G}^{\bar{\mathcal{C}}}$ can be reduced to

$$\boldsymbol{G}^{\bar{\mathcal{C}}} = \begin{pmatrix} \boldsymbol{G}^{\mathcal{U}} & \mathbf{0}_{k_1 \times n_1} \\ \mathbf{0}_{k_1 \times n_1} & \boldsymbol{G}^{\mathcal{U}} \\ \mathbf{0}_{k_1 \times n_1} & \mathbf{0}_{k_1 \times n_1} \\ \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} & \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} \\ \mathbf{0}_{1 \times n_1} & \mathbf{1}_{1 \times n_1} \end{pmatrix}. \tag{III.59}$$

Let $\boldsymbol{G}^{\mathcal{U}} = \left(\boldsymbol{I}_{k_1} | \boldsymbol{P}_{k_1 \times (n_1 - k_1)}\right)$, where $\boldsymbol{P}_{k_1 \times (n_1 - k_1)}$ is the parity matrix of size $k_1 \times (n_1 - k_1)$. We now count the number of independent rows in the matrix

$$\begin{pmatrix} \boldsymbol{G}^{\mathcal{U}} \\ \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{k_1} & \boldsymbol{P}_{k_1 \times (n_1 - k_1)} \\ & \boldsymbol{\theta}_{\binom{k_1}{2} \times n_1} \end{pmatrix}.$$

Upon performing Gaussian elimination, we get

$$\begin{pmatrix} \boldsymbol{I}_{k_1} & \boldsymbol{P}_{k_1\times(n_1-k_1)} \\ \boldsymbol{0}_{\binom{k_1}{2}\times k_1} & \boldsymbol{\Delta}_{\binom{k_1}{2}\times(n_1-k_1)} \end{pmatrix},$$

where $\boldsymbol{\Delta}_{\binom{k_1}{2}\times(n_1-k_1)}$ is a matrix of dimensions $\binom{k_1}{2} \times (n_1 - k_1)$ with elements in GF(2). Hence, we have $\operatorname{rank}(\boldsymbol{\Delta}) \leq \min\left(\binom{k_1}{2}, (n_1 - k_1)\right)$. From this and (III.59), we can easily see that

$$\tilde{k} = \operatorname{rank}\left(\boldsymbol{G}^{\tilde{\mathcal{C}}}\right) = k_1 + k_1 + \operatorname{rank}(\boldsymbol{\Delta}) + 1 \leq \begin{cases} k_1 + n_1 + 1 & \text{if } n_1 - k_1 \leq \binom{k_1}{2}, \\ 2k_1 + \binom{k_1}{2} + 1 & \text{otherwise.} \end{cases}$$

□

Lemma 9 gives an upper bound on the dimension of $\tilde{\mathcal{C}}$. In order to prove $\dim(\tilde{\mathcal{C}}) < n$, we check when the upper bound in (III.51) is at most $n - 1$. For the first case in (III.51), we need to show

$$\tilde{k} \leq k_1 + n_1 + 1 \leq 2n_1 - 1.$$

Clearly, this is true since $n_1 \geq k_1 + 2$ by assumption. For the second case in (III.51) we have to show

$$\tilde{k} \leq 2k_1 + \binom{k_1}{2} + 1 \leq 2n_1 - 1.$$

Since $n_1 > \binom{k_1}{2} + k_1$, the above inequality reduces to

$$\binom{k_1}{2} > 2.$$

Clearly, this is true for $k_1 \in \mathbb{N}_{3:\infty}$. In the following, we argue for $k_1 \in \mathbb{N}_2$. Since $n_1 \geq k_1 + 2$ by assumption, we have

$$2n_1 - 1 \geq 2(k_1 + 2) - 1 = 2k_1 + 3 > 2k_1 + \binom{k_1}{2} + 1,$$

for $k_1 \in \mathbb{N}_2$. Therefore, $\dim(\tilde{\mathcal{C}}) < n$ for $n_1 \geq k_1 + 2$.

References

[1] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. Annual IEEE Symp. Foundations Comp. Sci. (FOCS)*, Milwaukee, WI, Oct. 1995, pp. 41–50.

[2] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, Nov. 1998.

[3] A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond, "Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval," in *Proc. Annual IEEE Symp. Foundations Comp. Sci. (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 261–270.

[4] S. Yekhanin, "Towards 3-query locally decodable codes of subexponential length," *Journal of the ACM*, vol. 55, no. 1, pp. 1–16, Feb. 2008.

[5] K. Efremenko, "3-query locally decodable codes of subexponential length," in *Proc. 41th Annual ACM Symp. Theory Comput. (STOC)*, Bethesda, MD, Jun. 2009, pp. 39–44.

[6] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 36th Annual ACM Symp. Theory Comput. (STOC)*, Chicago, IL, Jun. 2004, pp. 262–271.

[7] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, Jun./Jul. 2014, pp. 856–860.

[8] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, China, Jun. 2015, pp. 2842–2846.

[9] A. Fazeli, A. Vardy, and E. Yaakobi, "Codes for distributed PIR with low storage overhead," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, China, Jun. 2015, pp. 2852–2856.

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

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

[12] R. Tajeddine, O. W. Gnilke, and S. El Rouayheb, "Private information retrieval from MDS coded data in distributed storage systems," 2018, to app. in *IEEE Trans. Inf. Theory*.

[13] 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 Trans. Inf. Theory*, vol. 64, no. 2, pp. 1000–1022, Feb. 2018.

[14] ——, "The capacity of robust private information retrieval with colluding databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2361–2370, Apr. 2018.

[15] Q. Wang and M. Skoglund, "Linear symmetric private information retrieval for MDS coded distributed storage with colluding servers," Aug. 2017, arXiv:1708.05673v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1708.05673

[16] ——, "Secure symmetric private information retrieval from colluding databases with adversaries," Jul. 2017, arXiv:1707.02152v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1707.02152

[17] V. Guruswami and M. Wootters, "Repairing Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5684–5698, Sep. 2017.

[18] I. Tamo, M. Ye, and A. Barg, "Optimal repair of Reed-Solomon codes: Achieving the cut-set bound," May 2017, arXiv:cs/1706.00112v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1706.00112

[19] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. IEEE Int. Symp. Net. Comp. Appl. (NCA)*, Cambridge, MA, Jul. 2007, pp. 79–86.

[20] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," in *Proc. 39th Very Large Data Bases Endowment (VLDB)*, Trento, Italy, Aug. 2013, pp. 325–336.

[21] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in Windows Azure storage," in *Proc. USENIX Annual Tech. Conf.*, Boston, MA, Jun. 2012.

[22] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[23] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, Aug. 2014.

[24] S. Kumar, E. Rosnes, and A. Graell i Amat, "Private information retrieval in distributed storage systems using an arbitrary linear code," in *Proc IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 1421–1425.

[25] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, and D. A. Karpuk, "Private information retrieval from coded databases with colluding servers," *SIAM J. Appl. Algebra and Geom.*, vol. 1, no. 1, pp. 647–664, Nov. 2017.

[26] R. Freij-Hollanti, O. Gnilke, C. Hollanti, A.-L. Horlemann-Trautmann, D. Karpuk, and I. Kubjas, "Reed-Muller codes for private information reterival," in *Proc. 10th Int. Workshop Coding Cryptography (WCC)*, Saint-Petersburg, Russia, Sep. 2017.

[27] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[28] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, A.-L. Horlemann-Trautmann, D. Karpuk, and I. Kubjas, "t-private information retrieval schemes using transitive codes," Dec. 2017, arXiv:1712.02850v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1712.02850

[29] V. K. Wei, "Generalized Hamming weights for linear codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 5, pp. 1412–1418, Sep. 1991.

[30] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge University Press, 2013.

[31] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Trans. IRE Prof. Gro. Inf. Theory*, vol. 4, no. 4, pp. 38–49, Sep. 1954.

[32] J. D. Key, T. P. McDonough, and V. C. Mavron, "Information sets and partial permutation decoding for codes from finite geometries," *Finite Fields Th. App.*, vol. 12, no. 2, pp. 232–247, Jun. 2006.

[33] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[34] W. C. Huffman and V. Pless, Eds., *Fundamentals of Error-Correcting Codes*. Cambridge, UK: Cambridge University Press, 2010.

[35] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[36] P. Delsarte, J. M. Goethals, and F. J. MacWilliams, "On generalized Reed-Muller codes and their relatives," *Inf. Contr.*, vol. 16, no. 5, pp. 403–442, Jul. 1970.

[37] J. L. Fan, "Array codes as low-density parity-check codes," in *Proc. 2nd Int. Symp. Turbo Codes & Rel. Topics (ISTC)*, Brest, France, Sep. 2000, pp. 543–546.

[38] K. Yang and T. Helleseth, "On the minimum distance of array codes as LDPC codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3268–3271, Dec. 2003.

[39] J. Hao and S.-T. Xia, "Constructions of optimal binary locally repairable codes with multiple repair groups," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1060–1063, Jun. 2016.

[40] E. Rosnes, M. A. Ambroze, and M. Tomlinson, "On the minimum/stopping distance of array low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5204–5214, Sep. 2014.

[41] C. Feyling, "Punctured maximum distance separable codes," *Electron. Lett.*, vol. 29, no. 5, pp. 470–471, Mar. 1993.

Paper IV

Asymmetry Helps: Improved Private Information Retrieval Protocols for Distributed Storage

Hsuan-Yin Lin, Siddhartha Kumar, Eirik Rosnes, and Alexandre Graell i Amat

In *Proc. IEEE Information Theory Workshop (ITW)*, Guangzhou, China, November 2018.

The proofs of the theorems presented here were not included in the paper presented at ITW. We will include these proofs in the extended version of the paper, titled "On the Fundamental Limit of Private Information Retrieval for Coded Distributed Storage", for possible future publication in a journal.

The layout has been revised.

Abstract

We consider private information retrieval (PIR) for distributed storage systems (DSSs) with noncolluding nodes where data is stored using a non maximum distance separable (MDS) linear code. It was recently shown that if data is stored using a particular class of non-MDS linear codes, the *MDS-PIR capacity*, i.e., the maximum possible PIR rate for MDS-coded DSSs, can be achieved. For this class of codes, we prove that the PIR capacity is indeed equal to the MDS-PIR capacity, giving the first family of non-MDS codes for which the PIR capacity is known. For other codes, we provide asymmetric PIR protocols that achieve a strictly larger PIR rate compared to existing symmetric PIR protocols.

1 Introduction

The concept of private information retrieval (PIR) was first introduced by Chor *et al.* [1]. A PIR protocol allows a user to privately retrieve an arbitrary data item stored in multiple servers (referred to as nodes in the sequel) without disclosing any information of the requested item to the nodes. The efficiency of a PIR protocol is measured in terms of the total communication cost between the user and the nodes, which is equal to the sum of the upload and download costs. In distributed storage systems (DSSs), data is encoded by an $[n, k]$ linear code and then stored on n nodes in a distributed manner. Such DSSs are referred to as coded DSSs [2, 3].

One of the primary aims in PIR is the design of efficient PIR protocols from an information-theoretic perspective. Since the upload cost does not scale with the file size, the download cost dominates the total communication cost [3, 4]. Thus, the efficiency of a PIR protocol is commonly measured by the amount of information retrieved per downloaded symbol, referred to as the PIR rate. Recently, Sun and Jafar derived the maximum achievable PIR rate, the so-called *PIR capacity*, for the case of DSSs with replicated data [5, 6]. In the case where the data stored is encoded by an MDS storage code (the so-called *MDS-coded DSS*) and no nodes collude, a closed-form expression for the PIR capacity, referred to as the *MDS-PIR capacity*, was derived in [7].

In the earlier work [8–10], the authors focused on the properties of non-MDS storage codes in order to achieve the MDS-PIR capacity. In particular, in [9, 10] it was shown that the MDS-PIR capacity can be achieved for a special class of non-MDS linear codes, which, with some abuse of language, we refer to as *MDS-PIR capacity-achieving* codes (there might exist other codes outside of this class that achieve the MDS-PIR capacity). However, it is still unknown whether the MDS-PIR capacity is the best possible PIR rate that can be achieved for an arbitrarily coded DSS. In particular, an expression for the PIR capacity for coded DSSs with arbitrary linear storage codes is still missing.

In this paper, we first prove that the PIR capacity of coded DSSs that use the class of MDS-PIR capacity-achieving codes introduced in [9] is equal to the MDS-

PIR capacity. We then address the fundamental question of what is the maximum achievable PIR rate for an arbitrarily coded DSS. To this purpose, we mainly consider non-MDS-PIR capacity-achieving codes. Most of the earlier works focus on designing symmetric PIR protocols and it was shown in [5, 7, 11] that any PIR scheme can be made symmetric for MDS-coded DSSs. However, this is in general not the case for non-MDS codes. Specifically, we propose an *asymmetric* PIR protocol, Protocol A, that allows asymmetry in the responses from the storage nodes. For non-MDS-PIR capacity-achieving codes, Protocol A achieves improved PIR rates compared to the PIR rates of existing symmetric PIR protocols. Furthermore, we present an asymmetric PIR protocol, named Protocol B, that applies to non-MDS-PIR capacity-achieving codes that can be written as a direct sum of MDS-PIR capacity-achieving codes. Finally, we give an example showing that it is possible to construct an improved (compared to Protocol A) asymmetric PIR protocol. However, the protocol is code-dependent and strongly relies on finding *good* punctured MDS-PIR capacity-achieving subcodes of the non-MDS-PIR capacity-achieving code.

2 Preliminaries and System Model

2.1 Notation and Definitions

We denote by $\mathbb{N}$ the set of all positive integers and by $\mathbb{N}_a \triangleq \{1, 2, \dots, a\}$. Vectors are denoted by lower case bold letters, matrices by upper case bold letters, and sets by calligraphic upper case letters, e.g., $\boldsymbol{x}$, $\boldsymbol{X}$, and $\mathcal{X}$ denote a vector, a matrix, and a set, respectively. In addition, $\mathcal{X}^{\mathrm{c}}$ denotes the complement of a set $\mathcal{X}$ in a universe set. For a given index set $\mathcal{S}$, we also write $X^{\mathcal{S}}$ and $Y_{\mathcal{S}}$ to represent $\left\{X^{(m)} \colon m \in \mathcal{S}\right\}$ and $\left\{Y_l \colon l \in \mathcal{S}\right\}$, respectively. The fonts of random and deterministic quantities are not distinguished typographically since it should be clear from the context. We denote a submatrix of $\boldsymbol{X}$ that is restricted in columns by the set $\mathcal{I}$ by $\boldsymbol{X}|_{\mathcal{I}}$. The function $\mathsf{LCM}(n_1, n_2, \dots, n_a)$ computes the lowest common multiple of a positive integers $n_1, n_2, \dots, n_a$. The function $\mathsf{H}(\cdot)$ represents the entropy of its argument and $\mathsf{I}(\cdot;\cdot)$ denotes the mutual information of the first argument with respect to the second argument. $(\cdot)^{\mathsf{T}}$ denotes the transpose of its argument. We use the customary code parameters $[n, k]$ to denote a code $\mathcal{C}$ over the finite field $\mathrm{GF}(q)$ of blocklength n and dimension k. A generator matrix of $\mathcal{C}$ is denoted by $\boldsymbol{G}^{\mathcal{C}}$, while $\mathcal{C}^{\boldsymbol{G}}$ represents the corresponding code generated by $\boldsymbol{G}$. The function $\chi(\boldsymbol{x})$ denotes the support of a vector $\boldsymbol{x}$, while the support of a code $\mathcal{C}$ is defined as the set of coordinates where not all codewords are zero. A set of coordinates of $\mathcal{C}$, $\mathcal{I} \subseteq \mathbb{N}_n$, of size k is said to be an *information set* if and only if $\boldsymbol{G}^{\mathcal{C}}|_{\mathcal{I}}$ is invertible. The s-th generalized Hamming weight of an $[n, k]$ code $\mathcal{C}$, denoted by $d_s^{\mathcal{C}}$, $s \in \mathbb{N}_k$, is defined as the cardinality of the smallest support of an s-dimensional subcode of $\mathcal{C}$.

2.2 System Model

We consider a DSS that stores f files $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(f)}$, where each file $\boldsymbol{X}^{(m)} = (x_{i,l}^{(m)})$, $m \in \mathbb{N}_f$, can be seen as a random $\beta \times k$ matrix over $\mathrm{GF}(q)$ with $\beta, k \in \mathbb{N}$. Assume that each entry $x_{i,l}^{(m)}$ of $\boldsymbol{X}^{(m)}$ is chosen independently and uniformly at random from $\mathrm{GF}(q)$, $m \in \mathbb{N}_f$. Thus,

$$\begin{aligned} \mathsf{H}\left(\boldsymbol{X}^{(m)}\right) &= \mathsf{L}, \ \forall\, m \in \mathbb{N}_f, \\ \mathsf{H}\left(\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(f)}\right) &= f\mathsf{L} \quad \text{(in } q\text{-ary units)}, \end{aligned}$$

where $\mathsf{L} \triangleq \beta \cdot k$. Each file is encoded using a linear code as follows. Let $\boldsymbol{x}_i^{(m)} = \left(x_{i,1}^{(m)}, \ldots, x_{i,k}^{(m)}\right)$, $i \in \mathbb{N}_\beta$, be a message vector corresponding to the i-th row of $\boldsymbol{X}^{(m)}$. Each $\boldsymbol{x}_i^{(m)}$ is encoded by an $[n, k]$ code $\mathcal{C}$ over $\mathrm{GF}(q)$ into a length-n codeword $\boldsymbol{c}_i^{(m)} = \left(c_{i,1}^{(m)}, \ldots, c_{i,n}^{(m)}\right)$. The βf generated codewords $\boldsymbol{c}_i^{(m)}$ are then arranged in the array $\boldsymbol{C} = \left((\boldsymbol{C}^{(1)})^\top | \ldots | (\boldsymbol{C}^{(f)})^\top\right)^\top$ of dimensions $\beta f \times n$, where $\boldsymbol{C}^{(m)} = \left((\boldsymbol{c}_1^{(m)})^\top | \ldots | (\boldsymbol{c}_\beta^{(m)})^\top\right)^\top$. The code symbols $c_{1,l}^{(m)}, \ldots, c_{\beta,l}^{(m)}$, $m \in \mathbb{N}_f$, for all f files are stored on the l-th storage node, $l \in \mathbb{N}_n$.

2.3 Privacy Model

To retrieve file $\boldsymbol{X}^{(m)}$ from the DSS, the user sends a random query $Q_l^{(m)}$ to the l-th node for all $l \in \mathbb{N}_n$. In response to the received query, node l sends the response $A_l^{(m)}$ back to the user. $A_l^{(m)}$ is a deterministic function of $Q_l^{(m)}$ and the code symbols stored in the node.

Definition 1. *Consider a DSS with n noncolluding nodes storing f files. A user who wishes to retrieve the m-th file sends the queries $Q_l^{(m)}$, $l \in \mathbb{N}_n$, to the storage nodes, which return the responses $A_l^{(m)}$. This scheme achieves perfect information-theoretic PIR if and only if*

$$\textit{Privacy:} \quad \mathsf{I}\left(m; Q_l^{(m)}, A_l^{(m)}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(f)}\right) = 0, \ \forall\, l \in \mathbb{N}_n, \tag{IV.1}$$

$$\textit{Recovery:} \quad \mathsf{H}\left(\boldsymbol{X}^{(m)} \,\middle|\, A_1^{(m)}, \ldots, A_n^{(m)}, Q_1^{(m)}, \ldots, Q_n^{(m)}\right) = 0. \tag{IV.2}$$

2.4 PIR Rate and Capacity

Definition 2. *The PIR rate of a PIR protocol, denoted by R, is the amount of information retrieved per downloaded symbol, i.e., $\mathsf{R} \triangleq \frac{\beta k}{\mathsf{D}}$, where D is the total number of downloaded symbols for the retrieval of a single file.*

We will write $\mathsf{R}(\mathcal{C})$ to highlight that the PIR rate depends on the underlying storage code $\mathcal{C}$. It was shown in [7] that for the noncolluding case and for a given number of files f stored using an $[n, k]$ MDS code, the MDS-PIR capacity is

$$\mathsf{C}_f^{[n,k]} \triangleq \frac{n-k}{n}\left[1 - \left(\frac{k}{n}\right)^f\right]^{-1}, \tag{IV.3}$$

where superscript "$[n,k]$" indicates the code parameters of the underlying MDS storage code. When the number of files f tends to infinity, (IV.3) reduces to

$$\mathrm{C}_{\infty}^{[n,k]} \triangleq \lim_{f\to\infty} \mathrm{C}_{f}^{[n,k]} = \frac{n-k}{n},$$

which we refer to as the asymptotic MDS-PIR capacity. Note that for the case of non-MDS linear codes, the PIR capacity is unknown.

2.5 MDS-PIR Capacity-Achieving Codes

In [9], two symmetric PIR protocols for coded DSSs, named Protocol 1 and Protocol 2, were proposed and shown to achieve the MDS-PIR capacity for certain important classes of non-MDS codes. Their PIR rates depend on the following property of the underlying storage code $\mathcal{C}$.

Definition 3. *Let $\mathcal{C}$ be an arbitrary $[n,k]$ code. A $\nu \times n$ binary matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ is said to be a* PIR achievable rate matrix *for $\mathcal{C}$ if the following conditions are satisfied.*

1. *The Hamming weight of each column of $\boldsymbol{\Lambda}_{\kappa,\nu}$ is κ, and*
2. *for each matrix row $\boldsymbol{\lambda}_i$, $i \in \mathbb{N}_{\nu}$, $\chi(\boldsymbol{\lambda}_i)$ always contains an information set.*

The following theorem gives the achievable PIR rate of Protocol 1 from [9, Thm. 1].

Theorem 1. *Consider a DSS that uses an $[n,k]$ code $\mathcal{C}$ to store f files. If a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ exists, then the PIR rate*

$$\mathrm{R}_{f,\mathrm{S}}(\mathcal{C}) \triangleq \frac{(\nu-\kappa)k}{\kappa n}\left[1-\left(\frac{\kappa}{\nu}\right)^{f}\right]^{-1} \tag{IV.4}$$

is achievable.

In (IV.4), we use subscript S to indicate that this PIR rate is achievable by the symmetric Protocol 1 in [9]. Define $\mathrm{R}_{\infty,\mathrm{S}}(\mathcal{C})$ as the limit of $\mathrm{R}_{f,\mathrm{S}}(\mathcal{C})$ as the number of files f tends to infinity, i.e., $\mathrm{R}_{\infty,\mathrm{S}}(\mathcal{C}) \triangleq \lim_{f\to\infty} \mathrm{R}_{f,\mathrm{S}}(\mathcal{C}) = \frac{(\nu-\kappa)k}{\kappa n}$. The asymptotic PIR rate $\mathrm{R}_{\infty,\mathrm{S}}(\mathcal{C})$ is also achieved by the file-independent Protocol 2 from [9].

Corollary 12. *If a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ exists for an $[n,k]$ code $\mathcal{C}$, then the MDS-PIR capacity* (IV.3) *is achievable.*

Definition 4. *A PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ for an $[n,k]$ code $\mathcal{C}$ is called an* MDS-PIR capacity-achieving *matrix, and $\mathcal{C}$ is referred to as an* MDS-PIR capacity-achieving *code.*

In the following, we briefly state a main result for Protocol 1 and Protocol 2 from [9] and compare the required number of stripes and download cost of these protocols.

Theorem 2. *If an MDS-PIR capacity-achieving matrix exists for an* $[n, k]$ *code* $\mathcal{C}$ *with* $\frac{\kappa}{v} = \frac{k}{n}$*, then the PIR rates* $\mathrm{C}_f^{[n,k]}$ *and* $\mathrm{C}_\infty^{[n,k]}$ *are achievable by Protocol 1 and Protocol 2 from [9], respectively, using the corresponding required* β *and* D*. From Definition 2, we have*

$$\frac{n\mathrm{D}}{\beta} = \begin{cases} k\left(\mathrm{C}_f^{[n,k]}\right)^{-1} & \text{for Protocol 1,} \\ k\left(\mathrm{C}_\infty^{[n,k]}\right)^{-1} & \text{for Protocol 2.} \end{cases} \tag{IV.5}$$

Furthermore, the smallest number of stripes β *of Protocol 1 and Protocol 2 is equal to* v^f *and* $\frac{\mathrm{LCM}(k,n-k)}{k}$*, respectively.*

The following theorem from [9, Thm. 3] provides a necessary condition for the existence of an MDS-PIR capacity-achieving matrix.

Theorem 3. *If an MDS-PIR capacity-achieving matrix exists for an* $[n, k]$ *code* $\mathcal{C}$*, then* $d_s^{\mathcal{C}} \geq \frac{n}{k}s$*,* $\forall\, s \in \mathbb{N}_k$*.*

3 PIR Capacity for MDS-PIR Capacity-Achieving Codes

In this section, we prove that the PIR capacity of MDS-PIR capacity-achieving codes is equal to the MDS-PIR capacity.

Theorem 4. *Consider a DSS that uses an* $[n, k]$ *MDS-PIR capacity-achieving code* $\mathcal{C}$ *to store* f *files. Then, the maximum achievable PIR rate over all possible PIR protocols, i.e., the PIR capacity, is equal to the MDS-PIR capacity* $\mathrm{C}_f^{[n,k]}$ *in* (IV.3)*.*

Proof. See Appendix A. □

Theorem 4 provides an expression for the PIR capacity for the family of MDS-PIR capacity-achieving codes (i.e., (IV.3)). Moreover, for any finite number of files f and in the asymptotic case where f tends to infinity, the PIR capacity can be achieved using Protocols 1 and 2 from [9], respectively.

4 Asymmetry Helps: Improved PIR Protocols

In this section, we present three asymmetric PIR protocols for non-MDS-PIR capacity-achieving codes, illustrating that asymmetry helps to improve the PIR rate. By asymmetry we simply mean that the number of symbols downloaded from the different nodes is not the same, i.e., for any fixed $m \in \mathbb{N}_f$, the entropies $\mathsf{H}(A_l^{(m)})$, $l \in \mathbb{N}_n$, may be different. This is in contrast to the case of MDS codes, where any asymmetric protocol can be made symmetric while preserving its PIR rate [5, 7, 11]. We start with a simple motivating example showing that the PIR rate of Protocol 1 from [9] can be improved for some underlying storage codes.

		Node 1	Node 2	Node 3	Node 4	Node 5
repetition 1	round 1	$y^{(1)}_{2(2-1)+1,1}$ $y^{(1)}_{2(2-1)+2,1}$	$y^{(1)}_{2(1-1)+1,2}$ $y^{(1)}_{2(1-1)+2,2}$	$y^{(1)}_{2(1-1)+1,3}$ $y^{(1)}_{2(1-1)+2,3}$	$y^{(1)}_{2(1-1)+1,4}$ $y^{(1)}_{2(1-1)+2,4}$	$y^{(1)}_{2(1-1)+1,5}$ $y^{(1)}_{2(1-1)+2,5}$
		$y^{(2)}_{3\cdot 0+2,1}$ $y^{(2)}_{3\cdot 0+3,1}$	$y^{(2)}_{3\cdot 0+1,2}$ $y^{(2)}_{3\cdot 0+3,2}$	$y^{(2)}_{5\cdot 0+1,3}$ $y^{(2)}_{3\cdot 0+3,3}$	$y^{(2)}_{3\cdot 0+1,4}$ $y^{(2)}_{3\cdot 0+2,4}$	$y^{(2)}_{3\cdot 0+1,5}$ $y^{(2)}_{3\cdot 0+2,5}$
	rnd. 2	$y^{(1)}_{2\cdot 3+2,1} + y^{(2)}_{3\cdot 0+1,1}$	$y^{(1)}_{2\cdot 3+1,2} + y^{(2)}_{3\cdot 0+2,2}$	$y^{(1)}_{2\cdot 3+1,3} + y^{(2)}_{3\cdot 0+2,3}$	$y^{(1)}_{2\cdot 3+1,4} + y^{(2)}_{3\cdot 0+3,4}$	$y^{(1)}_{2\cdot 3+1,5} + y^{(2)}_{3\cdot 0+3,5}$
repetition 2	round 1	$y^{(1)}_{2(3-1)+1,1}$ $y^{(1)}_{2(3-1)+2,1}$	$y^{(1)}_{2(3-1)+1,2}$ $y^{(1)}_{2(3-1)+2,2}$	$y^{(1)}_{2(3-1)+1,3}$ $y^{(1)}_{2(3-1)+2,3}$	$y^{(1)}_{2(2-1)+1,4}$ $y^{(1)}_{2(2-1)+2,4}$	$y^{(1)}_{2(2-1)+1,5}$ $y^{(1)}_{2(2-1)+2,5}$
		$y^{(2)}_{3\cdot 1+2,1}$ $y^{(2)}_{3\cdot 1+3,1}$	$y^{(2)}_{3\cdot 1+1,2}$ $y^{(2)}_{3\cdot 1+3,2}$	$y^{(2)}_{3\cdot 1+1,3}$ $y^{(2)}_{3\cdot 1+3,3}$	$y^{(2)}_{3\cdot 1+1,4}$ $y^{(2)}_{3\cdot 1+2,4}$	$y^{(2)}_{3\cdot 1+1,5}$ $y^{(2)}_{3\cdot 1+2,5}$
	rnd. 2	$y^{(1)}_{2\cdot 3+3,1} + y^{(2)}_{3\cdot 1+1,1}$	$y^{(1)}_{2\cdot 3+3,2} + y^{(2)}_{3\cdot 1+2,2}$	$y^{(1)}_{2\cdot 3+3,3} + y^{(2)}_{3\cdot 1+2,3}$	$y^{(1)}_{2\cdot 3+2,4} + y^{(2)}_{3\cdot 1+3,4}$	$y^{(1)}_{2\cdot 3+2,5} + y^{(2)}_{3\cdot 1+3,5}$

Table IV.1: Protocol 1 with a $[5,3]$ non-MDS-PIR capacity-achieving code for $f = 2$.

4.1 Protocol 1 From [9] is Not Optimal in General

Example 1. *Consider the* $[5,3]$ *code* $\mathcal{C}$ *with generator matrix*

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The smallest possible value of $\frac{\kappa}{\nu}$ *for which a PIR achievable rate matrix exists is* $\frac{2}{3}$ *and a corresponding PIR achievable rate matrix is*

$$\boldsymbol{\Lambda}_{2,3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

It is easy to verify that $\boldsymbol{\Lambda}_{2,3}$ *above is a PIR achievable rate matrix for code* $\mathcal{C}$*. Thus, the largest PIR rate for* $f = 2$ *files with Protocol 1 from [9] is* $\mathrm{R}_{2,\mathrm{S}} = \frac{3^3}{5\cdot 10} = \frac{27}{50}$*. In Table IV.1 (taken from [9, Sec. IV]), we list the downloaded sums of code symbols when retrieving file* $\boldsymbol{X}^{(1)}$ *and* $f = 2$ *files are stored. In the table, for each* $m \in \mathbb{N}_2$ *and* $\beta = \nu^f = 3^2$*, the interleaved code array* $\boldsymbol{Y}^{(m)}$ *with row vectors* $\boldsymbol{y}_i^{(m)} = \boldsymbol{c}_{\pi(i)}^{(m)}$*,* $i \in \mathbb{N}_{3^2}$*, is generated (according to Protocol 1 from [9]) by a randomly selected permutation function* $\pi(\cdot)$*. Observe that since* $\{2,3,4\} \subset \chi(\boldsymbol{\lambda}_1) = \{2,3,4,5\}$ *is an information set of* $\mathcal{C}$*, the five sums of*

$$\left\{y^{(1)}_{2(1-1)+1,5}, y^{(1)}_{2(1-1)+2,5}, y^{(2)}_{3\cdot 0+1,5}, y^{(1)}_{2\cdot 3+1,5} + y^{(2)}_{3\cdot 0+3,5}, y^{(2)}_{3\cdot 1+1,5}\right\}$$

are not necessarily required to recover $\boldsymbol{X}^{(1)}$*. For privacy concerns, notice that the remaining sums of code symbols from the* 5*-th node would be*

$$\left\{y^{(2)}_{3\cdot 0+2,5}, y^{(1)}_{2(2-1)+1,5}, y^{(1)}_{2\cdot(2-1)+2,5}, y^{(2)}_{3\cdot 1+2,5}, y^{(1)}_{2\cdot 3+2,5} + y^{(2)}_{3\cdot 1+3,5}\right\}.$$

This ensures the privacy condition, since for every combination of files, the user downloads the same number of linear sums. This shows that by allowing asymmetry in the responses from the storage nodes, the PIR rate can be improved to $\frac{27}{50-5} = \frac{27}{45} = \frac{3}{5}$*, which is much closer to the MDS-PIR capacity* $\mathrm{C}_2^{[5,3]} = \frac{1}{1+\frac{3}{5}} = \frac{5}{8}$*.*

Example 1 indicates that for a coded DSS using a non-MDS-PIR capacity-achieving code, there may exist an asymmetric PIR scheme that improves the PIR rate of the symmetric Protocol 1 from [9].

4.2 Protocol A: A General Asymmetric PIR Protocol

In this subsection, we show that for non-MDS-PIR capacity-achieving codes, by discarding the redundant coordinates that are not required to form an information set within $\chi(\boldsymbol{\lambda}_i)$, $i \in \mathbb{N}_\nu$, it is always possible to obtain a larger PIR rate compared to that of Protocol 1 from [9].

Theorem 5. *Consider a DSS that uses an $[n,k]$ code $\mathcal{C}$ to store f files. If a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ exists, then the PIR rate*

$$\mathrm{R}_{f,\mathsf{A}}(\mathcal{C}) \triangleq \left(1-\frac{\kappa}{\nu}\right)\left[1-\left(\frac{\kappa}{\nu}\right)^f\right]^{-1} \tag{IV.6}$$

is achievable.

Proof. See Appendix B. □

We will make use of the following lemma from [9, Lem. 2].

Lemma 1. *If a matrix $\boldsymbol{\Lambda}_{\nu,\kappa}(\mathcal{C})$ exists for an $[n,k]$ code $\mathcal{C}$, then we have*

$$\frac{\kappa}{\nu} \geq \frac{k}{n},$$

where equality holds if $\chi(\boldsymbol{\lambda}_i)$, $i \in \mathbb{N}_\nu$, are all information sets.

Proposition 1 can be easily verified using Lemma 1.

Proposition 5. *Consider a DSS that uses an $[n,k]$ code $\mathcal{C}$ to store f files. Then, $\mathrm{R}_{f,\mathsf{S}}(\mathcal{C}) \leq \mathrm{R}_{f,\mathsf{A}}(\mathcal{C}) \leq \mathrm{C}_f^{[n,k]}$ with equality if and only if $\mathcal{C}$ is an MDS-PIR capacity-achieving code.*

Proof. The result follows since

$$\begin{aligned}
\mathrm{R}_{f,\mathsf{S}}(\mathcal{C}) &= \frac{(\nu-\kappa)k}{\kappa n}\left[1-\left(\frac{\kappa}{\nu}\right)^f\right]^{-1} \\
&\leq \frac{(\nu-\kappa)k}{\kappa n-(\kappa n-\nu k)}\left[1-\left(\frac{\kappa}{\nu}\right)^f\right]^{-1} && \text{(IV.7)} \\
&= \left(1-\frac{\kappa}{\nu}\right)\left[1-\left(\frac{\kappa}{\nu}\right)^f\right]^{-1} = \mathrm{R}_{f,\mathsf{A}}(\mathcal{C}) \\
&= \left[1+\frac{\kappa}{\nu}+\cdots+\left(\frac{\kappa}{\nu}\right)^{f-1}\right]^{-1} && \text{(IV.8)} \\
&\leq \left[1+\frac{k}{n}+\cdots+\left(\frac{k}{n}\right)^{f-1}\right]^{-1} = \mathrm{C}_f^{[n,k]},
\end{aligned}$$

where both (IV.7) and (IV.8) hold since $\frac{\kappa}{\nu} \geq \frac{k}{n}$. □

In the following, we refer to the asymmetric PIR protocol that achieves the PIR rate in Theorem 5 as Protocol A (thus the subscript A in $\mathrm{R}_{f,\mathrm{A}}(\mathcal{C})$ in (IV.6)). Similar to Theorem 1, there also exists an asymmetric file-independent PIR protocol that achieves the asymptotic PIR rate $\mathrm{R}_{\infty,\mathrm{A}}(\mathcal{C}) \triangleq \lim_{f\to\infty} \mathrm{R}_{f,\mathrm{A}}(\mathcal{C}) = 1-\frac{\kappa}{\nu}$ and we simply refer to this protocol as the file-independent Protocol A.[1], $\Lambda_{\kappa,\nu}(\mathcal{C})$ can be used for both the file-dependent Protocol A and the file-independent Protocol A.

4.3 Protocol B: An Asymmetric PIR Protocol for a Special Class of Non-MDS-PIR Capacity-Achieving Codes

In this subsection, we focus on designing an asymmetric PIR protocol, referred to as Protocol B, for a special class of $[n,k]$ non-MDS-PIR capacity-achieving codes, where the code is isometric to a *direct sum* of $P \in \mathbb{N}_n$ MDS-PIR capacity-achieving codes [12, Ch. 2]. Without loss of generality, we assume that the generator matrix $\boldsymbol{G}$ of an $[n,k]$ non-MDS-PIR capacity-achieving code $\mathcal{C}$ has the structure

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{G}_1 & & & \\ & \boldsymbol{G}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{G}_P \end{pmatrix}, \tag{IV.9}$$

where $\boldsymbol{G}_p$, of size $k_p \times n_p$, is the generator matrix of a punctured MDS-PIR capacity-achieving subcode $\mathcal{C}^{\boldsymbol{G}_p}$, $p \in \mathbb{N}_P$.

Theorem 6. *Consider a DSS that uses an $[n,k]$ non-MDS-PIR capacity-achieving code $\mathcal{C}$ to store f files. If the code $\mathcal{C}$ is isometric to a direct sum of $P \in \mathbb{N}_n$ MDS-PIR capacity-achieving codes as in* (IV.9)*, then the PIR rate*

$$\mathrm{R}_{f,\mathrm{B}}(\mathcal{C}) \triangleq \left(\sum_{p=1}^{P} \frac{k_p}{k} \left(\mathrm{C}_f^{[n_p,k_p]} \right)^{-1} \right)^{-1} \tag{IV.10}$$

is achievable. Moreover, the asymptotic PIR rate

$$\mathrm{R}_{\infty,\mathrm{B}}(\mathcal{C}) \triangleq \lim_{f\to\infty} \mathrm{R}_{f,\mathrm{B}}(\mathcal{C}) = \left(\sum_{p=1}^{P} \frac{k_p}{k} \left(\mathrm{C}_\infty^{[n_p,k_p]} \right)^{-1} \right)^{-1} \tag{IV.11}$$

is achievable by a file-independent PIR protocol.

Proof. See Appendix C. □

We remark that Protocol B requires $\beta = \mathsf{LCM}(\beta_1, \ldots, \beta_P)$ stripes, where β_p, $p \in \mathbb{N}_P$, is the smallest number of stripes of either Protocol 1 or Protocol 2 for a DSS that uses only the punctured MDS-PIR capacity-achieving subcode $\mathcal{C}^{\boldsymbol{G}_p}$ to store f files (see the proof in Appendix C and Theorem 2 for the smallest number of stripes β_p).

Theorem 6 can be used to obtain a larger PIR rate for the non-MDS-PIR capacity-achieving code in Example 1.

[1] As for Protocol 1 and Protocol 2 from [9, Remark 2]

Subresponses	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9
Subresponse 1	$I_1 + x_{1,1}^{(m)}$	I_2	$I_3 + x_{1,3}^{(m)}$	$I_4 + x_{1,4}^{(m)}$	$I_5 + x_{1,5}^{(m)}$	$I_4 + I_5$	$I_3 + I_5$	$I_3 + I_4 + I_5$	$I_1 + I_2 + I_4 + I_5$
Subresponse 2	I_6	$I_7 + x_{1,2}^{(m)}$		I_9	I_{10}				$I_6 + I_7 + I_9 + I_{10}$

Table IV.2: Responses by Protocol C with a $[9, 5]$ non-MDS-PIR capacity-achieving code

Example 2. *Continuing with Example 1, by elementary matrix operations, the generator matrix of the* $[5, 3]$ *code of Example 1 is equivalent to the generator matrix*

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}_1 & \\ & \boldsymbol{G}_2 \end{pmatrix}.$$

It can easily be verified that both $\mathcal{C}^{\boldsymbol{G}_1}$ *and* $\mathcal{C}^{\boldsymbol{G}_2}$ *are MDS-PIR capacity-achieving codes. Hence, from Theorem 6, the asymptotic PIR rate*

$$\mathrm{R}_{\infty,\,\mathrm{B}} = \left(\frac{2}{3} \frac{1}{1 - \frac{2}{3}} + \frac{1}{3} \frac{1}{1 - \frac{1}{2}} \right)^{-1} = \frac{3}{8}$$

is achievable. $\mathrm{R}_{\infty,\,\mathrm{B}} = \frac{3}{8}$ *is strictly larger than both* $\mathrm{R}_{\infty,\,\mathrm{S}} = \frac{3}{10}$ *and* $\mathrm{R}_{\infty,\,\mathrm{A}} = \frac{1}{3}$.

4.4 Protocol C: Code-Dependent Asymmetric PIR Protocol

In this subsection, we provide a code-dependent, but file-independent asymmetric PIR protocol for non-MDS-PIR capacity-achieving codes that cannot be decomposed into a direct sum of MDS-PIR capacity-achieving codes as in (IV.9). The protocol is tailor-made for each class of storage codes. The main principle of the protocol is to further reduce the number of downloaded symbols by looking at punctured MDS-PIR capacity-achieving subcodes. Compared to Protocol A, which is simpler and allows for a closed-form expression for its PIR rate, Protocol C gives larger PIR rates.

The file-independent Protocol 2 from [9] utilizes *interference symbols*. An interference symbol can be defined through a summation as [9]

$$I_{k(h-1)+h'} \triangleq \sum_{m=1}^{f} \sum_{j=(m-1)\beta+1}^{m\beta} u_{h,j} x_{j-(m-1)\beta,h'}^{(m)},$$

where $h, h' \in \mathbb{N}_k$ and the symbols $u_{h,j}$ are chosen independently and uniformly at random from the same field as the code symbols.

Example 3. *Consider a* $[9, 5]$ *code* $\mathcal{C}$ *with generator matrix*

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

It has $d_2^{\mathcal{C}} = 3 < \frac{9}{5} \cdot 2$*, thus it is not MDS-PIR capacity-achieving (see Theorem 3). Note that this code cannot be decomposed into a direct sum of MDS-PIR capacity-achieving codes as in* (IV.9)*. The smallest* $\frac{\kappa}{\nu}$ *for which a PIR achievable rate matrix exists for this code is* $\frac{2}{3}$*, and a corresponding PIR achievable rate matrix is*

$$\boldsymbol{\Lambda}_{2,3} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The idea of the file-independent Protocol 2 from [9] is to use the information sets $\mathcal{I}_1 = \{2, 6, 7, 8, 9\}$ *and* $\mathcal{I}_2 = \{1, 3, 4, 5, 9\}$ *to recover the* $\beta k = 1 \cdot 5$ *requested file symbols that are located in* $\mathcal{I}_3 = \{1, 2, 3, 4, 5\}$*. Specifically, we use the information set* $\mathcal{I}_1$ *to reconstruct the required code symbols located in* $\chi(\boldsymbol{\lambda}_1)^c = \{1, 3, 4, 5\}$ *and* $\mathcal{I}_2 \subseteq \chi(\boldsymbol{\lambda}_2) = \{1, 3, 4, 5, 6, 7, 8, 9\}$ *to reconstruct the required code symbol located in* $\chi(\boldsymbol{\lambda}_2)^c = \{2\}$*. Since the code coordinates* $\{1, 2, 4, 5, 9\}$ *form an* $[n', k'] = [5, 4]$ *punctured MDS-PIR capacity-achieving subcode* $\mathcal{C}^{\boldsymbol{G}'}$ *with generator matrix*

$$\boldsymbol{G}' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

it can be seen that the code coordinates $\{1, 4, 5, 9\}$ *are sufficient to correct the erasure located in* $\chi(\boldsymbol{\lambda}_2)^c$*. Therefore, compared to Protocol A, we can further reduce the required number of downloaded symbols. The responses from the nodes when retrieving file* $\boldsymbol{X}^{(m)}$ *are listed in Table IV.2. The PIR rate of Protocol C is then equal to*

$$\mathrm{R}_{\infty,\,\mathrm{C}} = \frac{1 \cdot 5}{n + n'} = \frac{5}{14} < \frac{4}{9} = \mathrm{C}_{\infty}^{[9,5]},$$

which is strictly larger than $\mathrm{R}_{\infty,\,\mathrm{A}} = \frac{1}{3}$*. Notice that it can readily be seen from Table IV.2 that the privacy condition in (IV.1) is ensured.*

Finally, we remark that, using the same principle as outlined above, other punctured MDS-PIR capacity-achieving subcodes can be used to construct a valid protocol, giving the same PIR rate. For instance, we could pick the two punctured subcodes $\mathcal{C}^{\boldsymbol{G}_1}$ *and* $\mathcal{C}^{\boldsymbol{G}_2}$ *with generator matrices*

$$\boldsymbol{G}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \text{ and } \boldsymbol{G}_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

respectively.

Example 3 above illustrates the main working principle of Protocol C and how the redundant set of code coordinates is taken into account. Its general description will be given in a forthcoming extended version. However, some numerical results are given below, showing that it can attain larger PIR rates than Protocol A.

Code	$\frac{\kappa}{\nu}$	$R_{\infty,S}$	$R_{\infty,A}$	$R_{\infty,B}$	$R_{\infty,C}$	$C_{\infty}^{[n.k]}$
$\mathcal{C}_1 : [5,3]$	2/3	0.3	0.3333	0.375	0.375	0.4
$\mathcal{C}_2 : [9,5]$	2/3	0.2778	0.3333	–	0.3571	0.4444
$\mathcal{C}_3 : [7,4]$	3/5	0.3810	0.4	–	0.4	0.4286
$\mathcal{C}_4 : [11,6]$	3/4	0.1818	0.25	–	0.2824	0.4545

Table IV.3: PIR rate for different codes and protocols

5 Numerical Results

In Table IV.3, we compare the PIR rates for different protocols using several binary linear codes. The second column gives the smallest fraction $\frac{\kappa}{\nu}$ for which a PIR achievable rate matrix exists. In the table, code $\mathcal{C}_1$ is from Example 1, code $\mathcal{C}_2$ is from Example 3, $\mathcal{C}_3$ is a $[7,4]$ code with generator matrix $(1,2,4,8,8,14,5)$ (in decimal form, e.g., $(1,0,1,1)^\mathsf{T}$ is represented by 13) and $d_3^{\mathcal{C}_3} = 5 < \frac{7}{4} \cdot 3$, and $\mathcal{C}_4$ is an $[11,6]$ code with generator matrix $(1,2,4,8,16,32,48,40,24,56,55)$ and $d_3^{\mathcal{C}_4} = 4 < \frac{11}{6} \cdot 3$. Note that $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$ cannot be decomposed into a direct sum of MDS-PIR capacity-achieving codes as in (IV.9). For all presented codes except $\mathcal{C}_3$, Protocol C achieves strictly larger PIR rate than Protocol A, although smaller than the MDS-PIR capacity.

6 Conclusion

We proved that the PIR capacity for MDS-PIR capacity-achieving codes is equal to the MDS-PIR capacity for the case of noncolluding nodes, giving the first family of non-MDS codes for which the PIR capacity is known. We also showed that allowing asymmetry in the responses from the storage nodes yields larger PIR rates compared to symmetric protocols in the literature when the storage code is a non-MDS-PIR capacity-achieving code. We proposed three asymmetric protocols and compared them in terms of PIR rate for different storage codes.

A Proof of Theorem 4

Achievability is by Theorem 1 and Corollary 12. Hence, in this appendix, we only provide the converse proof of Theorem 4.

Before we proceed with the converse proof, we give some general results that hold for any PIR protocol.

1. Given a query $Q_l^{(m)}$ sent to the l-th node, $m \in \mathbb{N}_f$, the response $A_l^{(m)}$ received by the user is a function of $Q_l^{(m)}$ and the f coded chunks (denoted by $\boldsymbol{c}_l \triangleq \left(c_{1,l}^{(1)}, \ldots, c_{\beta,l}^{(1)}, c_{1,l}^{(2)}, \ldots, c_{\beta,l}^{(f)}\right)^\mathsf{T}$) that are stored in the l-th node. It follows that

$$\mathrm{H}\left(A_l^{(m)} \,\middle|\, Q_l^{(m)}, \boldsymbol{X}^{\mathbb{N}_f}\right) = \qquad \mathrm{H}\left(A_l^{(m)} \,\middle|\, Q_l^{(m)}, \boldsymbol{c}_l\right) = 0. \tag{IV.12}$$

2. From the condition of privacy, the l-th node should not be able to differentiate between the responses $A_l^{(m)}$ and $A_l^{(m')}$ when the user requests $\boldsymbol{X}^{(m)}$, $m \neq m'$. Hence,

$$\mathsf{H}\big(A_l^{(m)} \,\big|\, \mathcal{Q}, \boldsymbol{X}^{(m)}\big) = \mathsf{H}\big(A_l^{(m')} \,\big|\, \mathcal{Q}, \boldsymbol{X}^{(m)}\big), \tag{IV.13}$$

where $\mathcal{Q} \triangleq \big\{Q_l^{(m)} \colon m \in \mathbb{N}_f, l \in \mathbb{N}_n\big\}$ denotes the set of all possible queries made by the user. Although this seems to be intuitively true, a proof of this property is still required and can be found in [13, Lem. 3].

3. Consider a PIR protocol for a coded DSS that uses an $[n, k]$ code $\mathcal{C}$ to store f files. For any subset of files $\mathcal{M} \subseteq \mathbb{N}_f$ and for any information set $\mathcal{I}$ of $\mathcal{C}$, we have

$$\mathsf{H}\big(A_{\mathcal{I}}^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) = \sum_{l \in \mathcal{I}} \mathsf{H}\big(A_l^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big). \tag{IV.14}$$

The proof uses the linear independence of the columns of a generator matrix of $\mathcal{C}$ corresponding to an information set, and can be seen as a simple extension of [7, Lem. 2] or [13, Lem. 4].

Next, we state Shearer's Lemma, which represents a very useful entropy method for combinatorial problems.

Lemma 2 (Shearer's Lemma [14])**.** *Let $\mathcal{S}$ be a collection of subsets of $\mathbb{N}_n$, with each $l \in \mathbb{N}_n$ included in at least κ members of $\mathcal{S}$. For random variables $Z_1, \ldots, Z_n$, we have*

$$\sum_{\mathcal{S} \in \mathcal{S}} \mathsf{H}(Z_{\mathcal{S}}) \geq \kappa\, \mathsf{H}(Z_1, \ldots, Z_n).$$

Now, we are ready for the converse proof. By Lemma 1, since the code $\mathcal{C}$ is MDS-PIR capacity-achieving, there exist ν information sets $\mathcal{I}_1, \ldots, \mathcal{I}_\nu$ such that each coordinate $l \in \mathbb{N}_n$ is included in exactly κ members of $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_\nu\}$ with $\frac{\kappa}{\nu} = \frac{k}{n}$.

Applying the chain rule of entropy we have

$$\mathsf{H}\big(A_{\mathbb{N}_n}^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) \geq \mathsf{H}\big(A_{\mathcal{I}_i}^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big), \quad \forall\, i \in \mathbb{N}_\nu.$$

Let $m \in \mathcal{M}$ and $m' \in \mathcal{M}^{\mathrm{c}} \triangleq \mathbb{N}_f \setminus \mathcal{M}$. Following similar steps as in the proof

given in [7, 13], we get

$$
\begin{aligned}
\nu \,\mathsf{H}\big(A_{\mathbb{N}_n}^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) &\geq \sum_{i=1}^{\nu} \mathsf{H}\big(A_{\mathcal{I}_i}^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) \\
&= \sum_{i=1}^{\nu} \left(\sum_{l \in \mathcal{I}_i} \mathsf{H}\big(A_l^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) \right) && \text{(IV.15)} \\
&= \sum_{i=1}^{\nu} \left(\sum_{l \in \mathcal{I}_i} \mathsf{H}\big(A_l^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) \right) && \text{(IV.16)} \\
&= \sum_{i=1}^{\nu} \mathsf{H}\big(A_{\mathcal{I}_i}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) && \text{(IV.17)} \\
&\geq \kappa \,\mathsf{H}\big(A_{\mathbb{N}_n}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) && \text{(IV.18)} \\
&= \kappa \Big[\mathsf{H}\big(A_{\mathbb{N}_n}^{(m')}, \boldsymbol{X}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) - \mathsf{H}\big(\boldsymbol{X}^{(m')} \,\big|\, A_{\mathbb{N}_n}^{(m')}, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) \Big] \\
&= \kappa \Big[\mathsf{H}\big(\boldsymbol{X}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) + \mathsf{H}\big(A_{\mathbb{N}_n}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \boldsymbol{X}^{(m')}, \mathcal{Q}\big) - 0 \Big] && \text{(IV.19)} \\
&= \kappa \Big[\mathsf{H}\big(\boldsymbol{X}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}\big) + \mathsf{H}\big(A_{\mathbb{N}_n}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \boldsymbol{X}^{(m')}, \mathcal{Q}\big) \Big], && \text{(IV.20)}
\end{aligned}
$$

where (IV.15) and (IV.17) follow from (IV.14), (IV.16) is because of (IV.13), (IV.18) is due to Shearer's Lemma, (IV.19) is from the fact that the m'-th file $\boldsymbol{X}^{(m')}$ is determined by the responses $A_{\mathbb{N}_n}^{(m')}$ and the queries $\mathcal{Q}$, and finally, (IV.20) follows from the independence between the queries and the files. Therefore, we can conclude that

$$
\begin{aligned}
\mathsf{H}\big(A_{\mathbb{N}_n}^{(m)} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \mathcal{Q}\big) &\geq \frac{\kappa}{\nu} \mathsf{H}\big(\boldsymbol{X}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}\big) + \frac{\kappa}{\nu} \mathsf{H}\big(A_{\mathbb{N}_n}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \boldsymbol{X}^{(m')}, \mathcal{Q}\big) \\
&= \frac{k}{n} \mathsf{H}\big(\boldsymbol{X}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}\big) + \frac{k}{n} \mathsf{H}\big(A_{\mathbb{N}_n}^{(m')} \,\big|\, \boldsymbol{X}^{\mathcal{M}}, \boldsymbol{X}^{(m')}, \mathcal{Q}\big), && \text{(IV.21)}
\end{aligned}
$$

where we have used Definition 4 to obtain (IV.21).

Since there are in total f files, we can recursively use (IV.21) $f - 1$ times to obtain

$$
\begin{aligned}
\mathsf{H}\big(A_{\mathbb{N}_n}^{(1)} \,\big|\, \boldsymbol{X}^{(1)}, \mathcal{Q}\big) &\geq \sum_{m=1}^{f-1} \Big(\frac{k}{n}\Big)^m \mathsf{H}\big(\boldsymbol{X}^{(m+1)} \,\big|\, \boldsymbol{X}^{\mathbb{N}_m}\big) + \Big(\frac{k}{n}\Big)^{f-1} \mathsf{H}\big(A_{\mathbb{N}_n}^{(f)} \,\big|\, \boldsymbol{X}^{\mathbb{N}_f}, \mathcal{Q}\big) \\
&= \sum_{m=1}^{f-1} \Big(\frac{k}{n}\Big)^m \mathsf{H}\big(\boldsymbol{X}^{(m+1)} \,\big|\, \boldsymbol{X}^{\mathbb{N}_m}\big) && \text{(IV.22)} \\
&= \sum_{m=1}^{f-1} \Big(\frac{k}{n}\Big)^m \mathsf{L}, && \text{(IV.23)}
\end{aligned}
$$

where (IV.22) follows from (IV.12). (IV.23) holds since $\mathsf{H}\big(\boldsymbol{X}^{(m+1)} \mid \boldsymbol{X}^{\mathbb{N}_m}\big) = \mathsf{H}\big(\boldsymbol{X}^{(m+1)}\big) = \mathsf{L}$.

Now,

$$\begin{aligned}
\mathsf{L} &= \mathsf{H}\big(\boldsymbol{X}^{(1)}\big) \\
&= \mathsf{H}\big(\boldsymbol{X}^{(1)} \,\big|\, \mathcal{Q}\big) - \underbrace{\mathsf{H}\big(\boldsymbol{X}^{(1)} \,\big|\, A^{(1)}_{\mathbb{N}_n}, \mathcal{Q}\big)}_{=0} && \text{(IV.24)} \\
&= \mathsf{I}\big(\boldsymbol{X}^{(1)}; A^{(1)}_{\mathbb{N}_n} \,\big|\, \mathcal{Q}\big) \\
&= \mathsf{H}\Big(A^{(1)}_{\mathbb{N}_n} \,\Big|\, \mathcal{Q}\Big) - \mathsf{H}\Big(A^{(1)}_{\mathbb{N}_n} \,\Big|\, \boldsymbol{X}^{(1)}, \mathcal{Q}\Big) \\
&\leq \mathsf{H}\Big(A^{(1)}_{\mathbb{N}_n} \,\Big|\, \mathcal{Q}\Big) - \sum_{m=1}^{f-1} \Big(\frac{k}{n}\Big)^m \mathsf{L}, && \text{(IV.25)}
\end{aligned}$$

where (IV.24) follows since any file is independent of the queries $\mathcal{Q}$, and knowing the responses $A^{(1)}_{\mathbb{N}_n}$ and the queries $\mathcal{Q}$, one can determine $\boldsymbol{X}^{(1)}$. Inequality (IV.25) holds because of (IV.23).

Finally, the converse proof is completed by showing that

$$\begin{aligned}
\mathsf{R} &= \frac{\mathsf{L}}{\sum_{l=1}^{n} \mathsf{H}\Big(A^{(1)}_{l}\Big)} \\
&\leq \frac{\mathsf{L}}{\mathsf{H}\Big(A^{(1)}_{\mathbb{N}_n}\Big)} && \text{(IV.26)} \\
&\leq \frac{\mathsf{L}}{\mathsf{H}\Big(A^{(1)}_{\mathbb{N}_n} \,\Big|\, \mathcal{Q}\Big)} && \text{(IV.27)} \\
&\leq \frac{1}{1 + \sum_{m=1}^{f-1} \big(\frac{k}{n}\big)^m} = \mathsf{C}_f^{[n,k]}, && \text{(IV.28)}
\end{aligned}$$

where (IV.26) holds because of the chain rule of entropy, (IV.27) is due to the fact that conditioning reduces entropy, and we apply (IV.25) to obtain (IV.28).

B Proof of Theorem 5

The theorem is proved by showing that some downloaded symbols in Protocol 1 from [9] are not really necessary both from the recovery and the privacy perspective. The resulting protocol is named Protocol A, and the proof is based on the fact that for a PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C})$ of a code $\mathcal{C}$, to recover a file of size $\beta \times k$, exactly νk code coordinates of the ν information sets $\{\chi(\boldsymbol{\lambda}_i)\}_{i \in \mathbb{N}_\nu}$ are required to be exploited in Protocol 1. In order to illustrate the achievability proof, we have to review the steps and proof of Protocol 1 in [9, Sec. IV and App. B], and we refer the reader to [9] for the details. In particular, Protocol 1 in [9] is constructed from two matrices as defined below.

Definition 5. *For a given $\nu \times n$ PIR achievable rate matrix $\boldsymbol{\Lambda}_{\kappa,\nu}(\mathcal{C}) = (\lambda_{u,l})$, we define the PIR interference matrices $\boldsymbol{A}_{\kappa\times n} = (a_{i,l})$ and $\boldsymbol{B}_{(\nu-\kappa)\times n} = (b_{i,l})$ for the code $\mathcal{C}$ with*

$$a_{i,l} \triangleq u \text{ if } \lambda_{u,l} = 1, \forall\, l \in \mathbb{N}_n,\, i \in \mathbb{N}_\kappa, u \in \mathbb{N}_\nu,$$
$$b_{i,l} \triangleq u \text{ if } \lambda_{u,l} = 0, \forall\, l \in \mathbb{N}_n,\, i \in \mathbb{N}_{\nu-\kappa}, u \in \mathbb{N}_\nu.$$

Note that in Definition 5, for each $l \in \mathbb{N}_n$, distinct values of $u \in \mathbb{N}_\nu$ should be assigned for all i. Thus, the assignment is not unique in the sense that the order of the entries of each column of $\boldsymbol{A}$ and $\boldsymbol{B}$ can be permuted. Further, by $\mathcal{S}(a|\boldsymbol{A}_{\kappa\times n})$ we denote the set of column coordinates of matrix $\boldsymbol{A}_{\kappa\times n} = (a_{i,l})$ in which at least one of its entries is equal to a, i.e.,

$$\mathcal{S}(a|\boldsymbol{A}_{\kappa\times n}) \triangleq \{l \in \mathbb{N}_n \colon \exists\, a_{i,l} = a, i \in \mathbb{N}_\kappa\}.$$

Thus, Definition 5 leads to the following claim.

Claim 2 ([9, Claim 1])**.** *$\mathcal{S}(a|\boldsymbol{A}_{\kappa\times n})$ contains an information set of code $\mathcal{C}$, $\forall\, a \in \mathbb{N}_\nu$. Moreover, for an arbitrary entry $b_{i,l}$ of $\boldsymbol{B}_{(\nu-\kappa)\times n}$, $\mathcal{S}(b_{i,l}|\boldsymbol{A}_{\kappa\times n}) = \mathcal{S}(a|\boldsymbol{A}_{\kappa\times n}) \subseteq \mathbb{N}_n \setminus \{l\}$ if $b_{i,l} = a$.*

From Definition 5 we see that there are in total κn entries in $\boldsymbol{A}$ and each entry $a_{i,l}$ is related to a coordinate within $\chi(\boldsymbol{\lambda}_i)$, $i \in \mathbb{N}_\nu$, $l \in \mathbb{N}_n$. In Protocol 1 the user downloads the needed symbols in a total of κ repetitions and in the i-th repetition, $i \in \mathbb{N}_\kappa$, the user downloads the required symbols in a total of f rounds. Two types of symbols are downloaded by the user, *desired symbols*, which are directly related to the requested file (say $\boldsymbol{X}^{(1)}$), and *undesired symbols*, which are not related to the requested file, but are exploited to decode the requested file from the desired symbols.

Consider a fixed $i \in \mathbb{N}_\kappa$ and denote by $\mathrm{D}(a_{i,l})$ the total download cost of Protocol 1 resulting from a particular entry $a_{i,l}$, $l \in \mathbb{N}_n$. First, we focus on the undesired symbols downloaded in Step 2 of Protocol 1. In each repetition the user downloads

$$\kappa\frac{\binom{f-1}{\ell}\big[\mathrm{U}(\ell) - 1 - \mathrm{U}(\ell-1) + 1\big]}{\kappa} = \binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell-1}$$

undesired symbols resulting from a particular $a_{i,l}$ in the ℓ-th round, $\ell \in \mathbb{N}_{f-1}$, where $\mathrm{U}(\ell) \triangleq \sum_{h=1}^{\ell} \kappa^{f-(h+1)}(\nu-\kappa)^{h-1}$. Hence, for the undesired symbols associated with $a_{i,l}$, in total

$$\kappa\binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell-1} = \binom{f-1}{\ell}\kappa^{f-\ell}(\nu-\kappa)^{\ell-1} \tag{IV.29}$$

symbols are downloaded in every ℓ-th round of all κ repetitions.

Secondly, for a particular entry $a_{i,l}$ in the i-th repetition, the user downloads κ^{f-1} desired symbols from the l-th node in round $\ell = 1$, and

$$W(\ell) - 1 - W(\ell-1) + 1 = \binom{f-1}{\ell}\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell} \tag{IV.30}$$

extra desired symbols in the $(\ell+1)$-th round, $\ell \in \mathbb{N}_{f-1}$, where $W(\ell)$ is defined as

$$W(\ell) \triangleq \kappa^{f-1} + \sum_{h=1}^{\ell} \binom{f-1}{h} \kappa^{f-(h+1)} (\nu-\kappa)^h.$$

In summary, using (IV.29) and (IV.30), the download cost associated to entry $a_{i,l}$ is obtained as

$$\begin{aligned} \mathrm{D}(a_{i,l}) &= \sum_{\ell=1}^{f-1} \binom{f-1}{\ell} \kappa^{f-\ell} (\nu-\kappa)^{\ell-1} + \sum_{\ell=0}^{f-1} \binom{f-1}{\ell} \kappa^{f-(\ell+1)} (\nu-\kappa)^{\ell} \\ &= \frac{\nu^f - \kappa^f}{\nu-\kappa}. \end{aligned}$$

In the part of Step 2 of Protocol 1 that exploits side information, we only require ν information sets induced by the matrix $\boldsymbol{A}$ to reconstruct code symbols induced by $\boldsymbol{B}$. Moreover, from [9, App. B], after Step 2 of Protocol 1, $\beta = \nu^f$ rows of code symbols of length n have been downloaded, and again the information sets induced by the matrix $\boldsymbol{A}$ are enough to recover all length-k stripes of the requested file. In other words, $\kappa n - \nu k$ entries of $\boldsymbol{A}$ are redundant for the reconstruction of all $\beta = \nu^f$ stripes of the requested file. Thus, the improved PIR rate becomes

$$\begin{aligned} \frac{\beta k}{\mathrm{D}} &= \frac{\nu^f k}{\text{download cost of Protocol 1} - (\kappa n - \nu k)\mathrm{D}(a_{i,l})} \\ &= \frac{\nu^f k}{\frac{\kappa n}{\nu-\kappa}\left[\nu^f - \kappa^f\right] - \frac{\kappa n - \nu k}{\nu-\kappa}\left[\nu^f - \kappa^f\right]} \\ &= \frac{\nu^f k}{\frac{\nu k}{\nu-\kappa}\left[\nu^f - \kappa^f\right]} = \left(1 - \frac{\kappa}{\nu}\right)\left[1 - \left(\frac{\kappa}{\nu}\right)^f\right]^{-1}. \end{aligned}$$

Finally, we would like to emphasize that by removing the redundant downloaded sums of code symbols in Protocol 1, it can be shown that within each storage node in each round $\ell \in \mathbb{N}_f$ of all repetitions, file symmetry still remains. This follows from a similar argumentation as in the privacy part of the proof of Protocol 1 in [9, App. B]. In the following, we briefly explain that in each round $\ell \in \mathbb{N}_f$ of all repetitions, for each particular entry $a_{i,l}$ and for every combination of files $\mathcal{M} \subseteq \mathbb{N}_f$ with $|\mathcal{M}| = \ell$, the user requests the same number of every possible combination of files in $\mathrm{D}(a_{i,l})$.

- In the first round ($\ell = 1$) of all κ repetitions, it follows from (IV.29) that, for each $m' \in \mathbb{N}_{2:f}$, the number of downloaded undesired symbols resulting from a particular entry $a_{i,l}$ is κ^{f-1}, the same as the number of downloaded desired symbols resulting from $a_{i,l}$.

- In the $(\ell+1)$-th round of all κ repetitions, $\ell \in \mathbb{N}_{f-2}$, arbitrarily choose a combination of files $\mathcal{M} \subseteq \mathbb{N}_{2:f}$, where $|\mathcal{M}| = \ell$. For a particular entry $a_{i,l}$, it follows from (IV.30) that the total number of downloaded desired symbols for files pertaining to $\{1\} \cup \mathcal{M}$ is equal to $\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell}$. On the other hand, for the undesired symbols resulting from a particular $a_{i,l}$, it follows from (IV.29) that in the $(\ell+1)$-th round the user downloads $\kappa^{f-(\ell+1)}(\nu-\kappa)^{\ell}$ linear sums for a combination of files $\mathcal{M} \subseteq \mathbb{N}_{2:f}$, $|\mathcal{M}| = \ell+1$. Thus, in rounds $\mathbb{N}_{f-1} \setminus \{1\}$, an equal number of linear sums for all combinations of files $\mathcal{M} \subseteq \mathbb{N}_f$ are downloaded.

- In the f-th round, only desired symbols are downloaded. Since each desired symbol is a linear combination of code symbols from all f files, an equal number of linear sums is again downloaded from each file.

In summary, in response to each particular $a_{i,l}$, the user downloads the same number of linear sums for every possible combination of files. As illustrated above, this is inherent from Protocol 1, and hence the privacy condition of (IV.1) is still satisfied.

C Proof of Theorem 6

The result follows by treating Protocol 1 and Protocol 2 from [9] as subprotocols for each punctured MDS-PIR capacity-achieving subcode $\mathcal{C}^{\boldsymbol{G}_p}$, $p \in \mathbb{N}_P$. If Protocol 1 is used as a subprotocol, then we obtain the file-dependent Protocol B and the PIR rate in (IV.10), while if Protocol 2 is used as a subprotocol, then we obtain the file-independent Protocol B and the PIR rate in (IV.11).

For the asymmetric Protocol B, we require $\beta = \mathsf{LCM}(\beta_1, \ldots, \beta_P)$ stripes, where β_p, $p \in \mathbb{N}_P$, is the smallest number of stripes of either Protocol 1 or Protocol 2 for a DSS that uses only the punctured MDS-PIR capacity-achieving subcode $\mathcal{C}^{\boldsymbol{G}_p}$ to store f files (see Theorem 2). Note that for Protocol 1 the *index preparation*[2] should be made for all β stripes. Since $\sum_{p=1}^{P} k_p = k$ and $\sum_{p=1}^{P} n_p = n$, to privately retrieve the entire requested file consisting of k symbols in each stripe, we have to privately recover all P substripes of all β stripes, where the p-th substripe is of length k_p, by processing the subprotocol (either Protocol 1 or Protocol 2) for every punctured subcode $\mathcal{C}^{\boldsymbol{G}_p}$. In particular, for each punctured subcode $\mathcal{C}^{\boldsymbol{G}_p}$ we repeat the subprotocol β/β_p times to recover all the length-k_p requested substripes. This can be done since both Protocol 1 and Protocol 2 recover β_p stripes of length k_p, while repeating it β/β_p times enables the recovery of β length-k_p substripes. Note that privacy is ensured since the storage nodes of each punctured subcode are disjoint and within the nodes associated with each punctured subcode $\mathcal{C}^{\boldsymbol{G}_p}$ the subprotocol (Protocol 1 or Protocol 2) yields privacy against each server [9].

Denote by D_p the total download cost for each node for the punctured subcode $\mathcal{C}^{\boldsymbol{G}_p}$ using the subprotocol, $p \in \mathbb{N}_P$. The PIR rates of the file-dependent and

[2] This terminology was introduced in Step 1 of Protocol 1 from [9], i.e., the indices of the rows for each file are interleaved randomly and independently of each other.

file-independent Protocol B are given by

$$\frac{\beta k}{\mathrm{D}} = \frac{\beta k}{\sum_{p=1}^{P} \frac{\beta}{\beta_p} n_p \mathrm{D}_p} \tag{IV.31}$$

$$= \left(\sum_{p=1}^{P} \frac{1}{k} \frac{n_p \mathrm{D}_p}{\beta_p} \right)^{-1}$$

$$= \begin{cases} \left(\sum_{p=1}^{P} \frac{1}{k} k_p \left(\mathrm{C}_f^{[n_p, k_p]} \right)^{-1} \right)^{-1} & \text{if Protocol 1 is used as subprotocol,} \\ \left(\sum_{p=1}^{P} \frac{1}{k} k_p \left(\mathrm{C}_\infty^{[n_p, k_p]} \right)^{-1} \right)^{-1} & \text{if Protocol 2 is used as subprotocol,} \end{cases} \tag{IV.32}$$

where (IV.31) holds since within each punctured subcode, the subprotocol is required to be repeated $\frac{\beta}{\beta_p}$ times and (IV.32) follows from (IV.5).

References

[1] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. 36th IEEE Symp. Found. Comp. Sci.*, Milwaukee, WI, USA, Oct. 23–25, 1995, pp. 41–50.

[2] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, HI, USA, Jun. 29 – Jul. 4, 2014, pp. 856–860.

[3] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, China, Jun. 14–19, 2015, pp. 2842–2846.

[4] R. Tajeddine and S. El Rouayheb, "Private information retrieval from MDS coded data in distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 10–15, 2016, pp. 1411–1415.

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

[6] ——, "The capacity of robust private information retrieval with colluding databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2361–2370, Apr. 2018.

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

[8] S. Kumar, E. Rosnes, and A. Graell i Amat, "Private information retrieval in distributed storage systems using an arbitrary linear code," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Germany, Jun. 25–30, 2017, pp. 1421–1425.

[9] S. Kumar, H.-Y. Lin, E. Rosnes, and A. Graell i Amat, "Achieving maximum distance separable private information retrieval capacity with linear codes," Dec. 2017, arXiv:1712.03898v3 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1712.03898

[10] H.-Y. Lin, S. Kumar, E. Rosnes, and A. Graell i Amat, "An MDS-PIR capacity-achieving protocol for distributed storage using non-MDS linear codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, CO, USA, Jun. 17–22, 2018, pp. 966–970.

[11] H. Sun and S. A. Jafar, "Private information retrieval from MDS coded data with colluding servers: Settling a conjecture by Freij-Hollanti et al." in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Germany, Jun. 25–30, 2017, pp. 1893–1897.

[12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[13] J. Xu and Z. Zhang, "On sub-packetization of capacity-achieving PIR schemes for MDS coded databases," Dec. 2017, arXiv:1712.02466v2 [cs.IT]. [Online]. Available: http://arxiv.org/abs/1712.02466

[14] J. Radhakrishnan, "Entropy and counting," in *Computational mathematics, modelling and algorithms*, J. C. Misra, Ed. Narosa Publishing House, 2003, pp. 146–168.

Paper V

Private Information Retrieval From a Cellular Network With Caching at the Edge

Siddhartha Kumar, Alexandre Graell i Amat, Eirik Rosnes, and Linda Senigagliesi

Submitted to IEEE Transactions on Communications, August 2018.

The layout has been revised.

Abstract

We consider the problem of downloading content from a cellular network where content is cached at the wireless edge while achieving privacy. In particular, we consider private information retrieval (PIR) of content from a library of files, i.e., the user wishes to download a file and does not want the network to learn any information about which file she is interested in. To reduce the backhaul usage, content is cached at the wireless edge in a number of small-cell base station (SBS) using maximum distance separable codes. We propose a PIR scheme for this scenario that achieves privacy against a number of spy SBSs that (possibly) collaborate. The proposed PIR scheme is an extension of a recently introduced scheme by Kumar *et al.* to the case of multiple code rates, suitable for the scenario where files have different popularities. We then derive the backhaul rate and optimize the content placement to minimize it. We prove that uniform content placement is optimal, i.e., all files that are cached should be stored using the same code rate. This is in contrast to the case where no PIR is required. Furthermore, we show numerically that popular content placement is optimal for some scenarios.

1 Introduction

Bringing content closer to the end user in wireless networks, the so-called caching at the wireless edge, has emerged as a promising technique to reduce the backhaul usage. The literature on wireless caching is vast. Information-theoretic aspects of caching were studied in [1, 2]. To leverage the potential gains of caching, several papers proposed to cache files in densely deployed small-cell base stations (SBSs) with large storage capacity, see, e.g., [3–7]. In [5], content is cached in SBSs using maximum distance separable (MDS) codes to reduce the download delay. This scenario was further studied in [7], where the authors optimized the MDS-coded caching to minimize the backhaul rate. Caching content directly in the mobile devices and exploiting device-to-device communication has been considered in, e.g., [8–12].

Recently, private information retrieval (PIR) has attracted a significant interest in the research community [13–23]. In PIR, a user would like to retrieve data from a distributed storage system (DSS) in the presence of spy nodes, without revealing any information about the piece of data she is interested in to the spy nodes. PIR was first studied by Chor *et al.* [24] for the case where a binary database is replicated among n servers (nodes) and the aim is to privately retrieve a single bit from the database in the presence of a single spy node (referred to as the noncolluding case), while minimizing the total communication cost. In the last few years, spurred by the rise of DSSs, research on PIR has been focusing on the more general case where data is stored using a storage code.

The PIR capacity, i.e., the maximum achievable PIR rate, was studied in [18,

19, 21–23]. In [19, 23], the PIR capacity was derived for the scenario where data is stored in a DSS using a repetition code. In [22], for the noncolluding case, the authors derived the PIR capacity for the scenario where data is stored using an (single) MDS code, referred to as the MDS-PIR capacity. For the case where several spy nodes collaborate with each other, referred to as the colluding case, the MDS-PIR capacity is in general still unknown, except for some special cases [18] (and for repetition codes [23]). PIR protocols for DSSs have been proposed in [14, 16, 17, 20, 21]. In [16], a PIR protocol for MDS-coded DSSs was proposed and shown to achieve the MDS-PIR capacity for the case of noncolluding nodes when the number of files stored in the DSS goes to infinity. PIR protocols for the case where data is stored using non-MDS codes were proposed in [17, 20, 21].

In this paper, we consider PIR of content from a cellular network. In particular, we consider the private retrieval of content from a library of files that have different popularities. We consider a similar scenario as in [7] where, to reduce the backhaul usage, content is cached in SBSs using MDS codes. We propose a PIR scheme for this scenario that achieves privacy against a number of spy SBSs that possibly collude. The proposed PIR scheme is an extension of Protocol 3 in [21] to the case of multiple code rates, suitable for the scenario where files have different popularities. We also propose an MDS-coded content placement slightly different than the one in [7] but that is more adapted to the PIR case. We show that, for the conventional content retrieval scenario with no privacy, the proposed content placement is equivalent to the one in [7], in the sense that it yields the same average backhaul rate. We then derive the backhaul rate for the PIR case as a function of the content placement. We prove that uniform content placement, i.e., all files that are cached are encoded with the same code rate, is optimal. This is a somewhat surprising result, in contrast to the case where no PIR is considered, where optimal content placement is far from uniform [7]. We further consider the minimization of a weighted sum of the backhaul rate and the communication rate from the SBSs, relevant for the case where limiting the communication from the SBSs is also important. We finally report numerical results for both the scenario where SBSs are placed regularly in a grid and for a Poisson point process (PPP) deployment model where SBSs are distributed over the plane according to a PPP. We show numerically that popular content placement is optimal for some system parameters. To the best of our knowledge, PIR for the wireless caching scenario has not been considered before.

Notation: We use lower case bold letters to denote vectors, upper case bold letters to denote matrices, and calligraphic upper case letters to denote sets. For example, $\boldsymbol{x}$, $\boldsymbol{X}$, and $\mathcal{X}$ denote a vector, a matrix, and a set, respectively. We denote a submatrix of $\boldsymbol{X}$ that is restricted in columns by the set $\mathcal{I}$ by $\boldsymbol{X}|_{\mathcal{I}}$. $\mathcal{C}$ will denote a linear code over the finite field $\mathrm{GF}(q)$. The multiplicative subgroup of $\mathrm{GF}(q)$ (not containing the zero element) is denoted by $\mathrm{GF}(q)^{\times}$. We use the customary code parameters (n, k) to denote a code $\mathcal{C}$ of blocklength n and dimension k. A generator matrix for $\mathcal{C}$ will be denoted by $\boldsymbol{G}^{\mathcal{C}}$ and a parity-check matrix by $\boldsymbol{H}^{\mathcal{C}}$. A set of coordinates of $\mathcal{C}$, $\mathcal{I} \subseteq \{1, \ldots, n\}$, of size k is said to be an *information set* if and only if $\boldsymbol{G}^{\mathcal{C}}|_{\mathcal{I}}$ is invertible. The Hadamard product of two linear subspaces $\mathcal{C}$ and $\mathcal{C}'$, denoted by $\mathcal{C} \circ \mathcal{C}'$, is the space generated by the Hadamard products $\boldsymbol{c} \circ \boldsymbol{c}' \triangleq (c_1 c'_1, \ldots, c_n c'_n)$ for all pairs $\boldsymbol{c} \in \mathcal{C}$, $\boldsymbol{c}' \in \mathcal{C}'$. The inner product of two

vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ is denoted by $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$, while $w_{\mathsf{H}}(\boldsymbol{x})$ denotes the Hamming weight of $\boldsymbol{x}$. $(\cdot)^{\mathsf{T}}$ represents the transpose of its argument, while $\mathsf{H}(\cdot)$ represents the entropy function. With some abuse of language, we sometimes interchangeably refer to binary vectors as erasure patterns under the implicit assumption that the ones represent erasures. An erasure pattern (or binary vector) $\boldsymbol{x}$ is said to be correctable by a code $\mathcal{C}$ if matrix $\boldsymbol{H}^{\mathcal{C}}|_{\chi(\boldsymbol{x})}$ has rank $|\chi(\boldsymbol{x})|$.

2 System Model

We consider a cellular network where a macro-cell is served by a macro base station (MBS). Mobile users wish to download files from a library of F files that is always available at the MBS through a backhaul link. We assume all files of equal size.[1] In particular, each file consists of βL bits and is represented by a $\beta \times L$ matrix $\boldsymbol{X}^{(i)}$,

$$\boldsymbol{X}^{(i)} = \begin{pmatrix} \tilde{\boldsymbol{x}}_1^{(i)} \\ \vdots \\ \tilde{\boldsymbol{x}}_\beta^{(i)} \end{pmatrix}$$

where upperindex $i = 1, \ldots, F$ is the file index. Therefore, each file can be seen as divided into β stripes $\tilde{\boldsymbol{x}}_1^{(i)}, \ldots, \tilde{\boldsymbol{x}}_\beta^{(i)}$ of L bits each. The file library has popularity distribution $\boldsymbol{p} = (p_1, \ldots, p_F)$, where file $\boldsymbol{X}^{(i)}$ is requested with probability p_i. We also assume that N_{SBS} SBSs are deployed to serve requests and offload traffic from the MBS whenever possible. To this purpose, each SBS has a cache size equivalent to M files. The considered scenario is depicted in Fig. V.1.

2.1 Content Placement

File $\boldsymbol{X}^{(i)}$ is partitioned into βk_i packets of size L/k_i bits and encoded before being cached in the SBSs. In particular, each packet is mapped onto a symbol of the field $\mathrm{GF}(q^{\delta_i})$, with $\delta_i \geq \frac{L}{k_i \log_2 q}$. For simplicity, we assume that $\frac{L}{k_i \log_2 q}$ is integer and set $\delta_i = \frac{L}{k_i \log_2 q}$. Thus, stripe $\tilde{\boldsymbol{x}}_a^{(i)}$ can be equivalently represented by a stripe $\boldsymbol{x}_a^{(i)}$, $a = 1, \ldots, \beta$, of symbols over $\mathrm{GF}(q^{\delta_i})$. Each stripe $\boldsymbol{x}_a^{(i)}$ is then encoded using an (N_{SBS}, k_i) MDS code $\mathcal{C}_i$ over $\mathrm{GF}(q)$ into a codeword $\boldsymbol{c}_a^{(i)} = (c_{a,1}^{(i)}, \ldots, c_{a,N_{\mathsf{SBS}}}^{(i)})$, where code symbols $c_{a,j}^{(i)}$, $j = 1, \ldots, N_{\mathsf{SBS}}$, are over $\mathrm{GF}(q^{\delta_i})$. For later use, we define $k_{\min} \triangleq \min\{k_i\}$, $k_{\max} \triangleq \max\{k_i\}$, and $\delta_{\max} \triangleq \frac{L}{k_{\min} \log_2 q}$.

The encoded file can be represented by a $\beta \times N_{\mathsf{SBS}}$ matrix $\boldsymbol{C}^{(i)} = (c_{a,j}^{(i)})$. Code symbols $c_{a,j}^{(i)}$ are then stored in the j-th SBS (the ordering is unimportant). Thus, for each file $\boldsymbol{X}^{(i)}$, each SBS caches one coded symbol of each stripe of the file, i.e., a fraction $\mu_i = 1/k_i$ of the i-th file. As $k_i \in \{1, \ldots, N_{\mathsf{SBS}} - 1\}$,

$$\mu_i \in \mathcal{M} \triangleq \{0, 1/(N_{\mathsf{SBS}} - 1), \ldots, 1/2, 1\},$$

[1] Assuming files of equal size is without loss of generality, since content can always be divided into chunks of equal size.

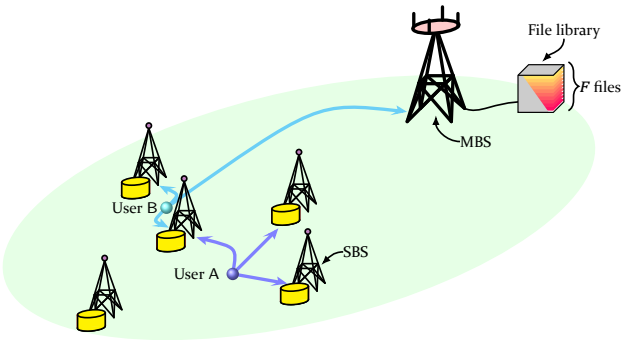

Figure V.1: A wireless network for content delivery consisting of an MBS and five SBSs. Users download files from a library of F files. The MBS has access to the library through a backhaul link. Some files are also cached at SBSs using a $(5, 3)$ MDS code. User A retrieves a cached file from the three SBSs within range. User B retrieves a fraction $2/3$ of a cached file from the two SBSs within range and the remaining fraction from the MBS.

where $\mu_i = 0$ implies that file $\boldsymbol{X}^{(i)}$ is not cached. Note that, to achieve privacy, $k_i < N_{\mathsf{SBS}}$, i.e., files need to be cached with redundancy. As a result, $\mu_i = 1/N_{\mathsf{SBS}}$ is not allowed. This is in contrast to the case of no PIR, where $k_i = N_{\mathsf{SBS}}$ (and hence $\mu_i = 1/N_{\mathsf{SBS}}$) is possible.

Since each SBS can cache the equivalent of M files, the μ_i's must satisfy

$$\sum_{i=1}^{F} \mu_i \leq M.$$

We define the vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_F)$ and refer to it as the *content placement*. Also, we denote by $\mathcal{C}^{\boldsymbol{\mu}}_{\mathsf{MDS}}$ the caching scheme that uses MDS codes $\{\mathcal{C}_i\}$ according to the content placement $\boldsymbol{\mu}$. For later use, we define $\mu_{\mathsf{min}} \triangleq \min\{\mu_i | \mu_i \neq 0\}$ and $\mu_{\mathsf{max}} \triangleq \max\{\mu_i\}$.

We remark that the content placement above is slightly different than the content placement proposed in [7]. In particular, we assume fixed code length (equal to the number of SBSs, N_{SBS}) and variable k_i, such that, for each file cached, each SBS caches a single symbol from each stripe of the file. In [7], the content placement is done by first dividing each file into k symbols and encoding them using an $(\tilde{n}_i, k)$ MDS code, where $\tilde{n}_i = k + (N_{\mathsf{SBS}} - 1)m_i$, $m_i \leq k$. Then, m_i (different) symbols of the i-th file are stored in each SBS and the MBS stores $k - m_i$ symbols.[2] Our formulation is perhaps a bit simpler and more natural from a coding perspective. Furthermore, we will show in Section 4 that the proposed content placement is equivalent to the one in [7], in the sense that it yields the same average backhaul rate.

[2]This is because the model in [7] assumes that one SBS is always accessible to the user. If this is not the case, the MBS must store all k symbols of the file. Here, we consider the case where the MBS must store all k symbols because it is a bit more general.

2.2 File Request

Mobile devices request files according to the popularity distribution $\boldsymbol{p} = (p_1, \ldots, p_F)$. Without loss of generality, we assume $p_1 \geq p_2 \geq \ldots \geq p_F$. The user request is initially served by the SBSs within communication range. We denote by γ_b the probability that the user is served by b SBSs and define $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{N_{\mathsf{SBS}}})$. If the user is not able to completely retrieve $\boldsymbol{X}^{(i)}$ from the SBSs, the additional required symbols are fetched from the MBS. Using the terminology in [7], the average fraction of files that are downloaded from the MBS is referred to as the backhaul rate, denoted by R, and defined as

$$\mathrm{R} \triangleq \frac{\text{average no. of bits downloaded from the MBS}}{\beta L}.$$

Note that for the case of no caching $\mathrm{R} = 1$.

As in [7], we assume that the communication is error free.

2.3 Private Information Retrieval and Problem Formulation

We assume that some of the SBSs are spy nodes that (potentially) collaborate with each other. On the other hand, we assume that the MBS can be trusted. The users wish to retrieve files from the cellular network, but do not want the spy nodes to learn any information about which file is requested by the user. The goal is to retrieve data from the network privately while minimizing the use of the backhaul link, i.e., while minimizing R. Thus, the goal is to optimize the content placement $\boldsymbol{\mu}$ to minimize R.

3 Private Information Retrieval Protocol

In this section, we present a PIR protocol for the caching scenario. The PIR protocol proposed here is an extension of Protocol 3 in [21] to the case of multiple code rates.[3]

Assume without loss of generality that the user wants to download file $\boldsymbol{X}^{(i)}$. To retrieve the file, the user generates $n \leq N_{\mathsf{SBS}}$ query matrices, $\boldsymbol{Q}^{(l)}$, $l = 1, \ldots, n$, where $\boldsymbol{Q}^{(1)}, \ldots, \boldsymbol{Q}^{(b)}$ are the queries sent to the b SBSs within visibility and the remaining $n - b$ queries $\boldsymbol{Q}^{(b+1)}, \ldots, \boldsymbol{Q}^{(n)}$ are sent to the MBS. Note that n is a parameter that needs to be optimized. Each query matrix is of size $d \times \beta F$ symbols (from $\mathrm{GF}(q)$) and has the following structure,

$$\boldsymbol{Q}^{(l)} = \begin{pmatrix} \boldsymbol{q}_1^{(l)} \\ \boldsymbol{q}_2^{(l)} \\ \vdots \\ \boldsymbol{q}_d^{(l)} \end{pmatrix} = \begin{pmatrix} q_{1,1}^{(l)} & q_{1,2}^{(l)} & \cdots & q_{1,\beta F}^{(l)} \\ q_{2,1}^{(l)} & q_{2,2}^{(l)} & \cdots & q_{2,\beta F}^{(l)} \\ \vdots & \vdots & \cdots & \vdots \\ q_{d,1}^{(l)} & q_{d,2}^{(l)} & \cdots & q_{d,\beta F}^{(l)} \end{pmatrix}.$$

[3] Protocol 3 in [21] is based on and improves the protocol in [20], in the sense that it achieves higher PIR rates.

The query matrix $\boldsymbol{Q}^{(l)}$ consists of d subqueries $\boldsymbol{q}_j^{(l)}$, $j = 1, \dots, d$, of length βF symbols each. In response to query matrix $\boldsymbol{Q}^{(l)}$, a SBS (or the MBS) sends back to the user a response vector $\boldsymbol{r}^{(l)} = (r_1^{(l)}, \dots, r_d^{(l)})^\top$ of length d, computed as

$$\boldsymbol{r}^{(l)} = (r_1^{(l)}, \dots, r_d^{(l)})^\top = \boldsymbol{Q}^{(l)}\left(c_{1,l}^{(1)}, \dots, c_{\beta,l}^{(1)}, \dots, c_{\beta,l}^{(F)}\right)^\top. \tag{V.1}$$

We will denote the j-th entry of the response vector $\boldsymbol{r}^{(l)}$, i.e., $r_j^{(l)}$, as the j-th subresponse of $\boldsymbol{r}^{(l)}$. Each response vector consists of d subresponses, each being a linear combination of βF symbols. Note that the operations are performed over the largest extension field, i.e., $\mathrm{GF}(q^{\delta_{\max}})$, and the subresponses are also over this field, i.e., each subresponse is of size $L/k_{\min} = L\mu_{\max}$ bits and hence each response is of size $dL\mu_{\max}$ bits.

The queries and the responses must be such that privacy is ensured and the user is able to recover the requested file. More precisely, information-theoretic PIR in the context of wireless caching with spy SBSs is defined as follows.

Definition 1. *Consider a wireless caching scenario with N_{SBS} SBSs that cache parts of a library of F files and in which a set $\mathcal{T}$ of T SBSs act as colluding spies. A user wishes to retrieve the i-th file and generates queries $\boldsymbol{Q}^{(l)}$, $l = 1, \dots, n$. In response to the queries the SBSs and (potentially) the MBS send back the responses $\boldsymbol{r}^{(l)}$. This scheme achieves perfect information-theoretic PIR if and only if*

$$\textit{Privacy:} \qquad \mathrm{H}\left(i | \boldsymbol{Q}^{(l)}, l \in \mathcal{T}\right) = \mathrm{H}(i); \tag{V.2a}$$

$$\textit{Recovery:} \qquad \mathrm{H}\left(\boldsymbol{X}^{(i)} | \boldsymbol{r}^{(1)}, \dots, \boldsymbol{r}^{(n)}\right) = 0. \tag{V.2b}$$

Condition (V.2a) means that the spy SBSs gain no additional information about which file is requested from the queries (i.e., the uncertainty about the file requested after observing the queries is identical to the a priori uncertainty determined by the popularity distribution), while Condition (V.2b) guarantees that the user is able to recover the file from the n response vectors.

We define the (n, k_i) code $\mathcal{C}_i'$, $i = 1, \dots, F$, as the code obtained by puncturing the underlying (N_{SBS}, k_i) storage code $\mathcal{C}_i$, and by $\mathcal{C}_{\max}'$ the code with parameters $(n, k_{\max})$.[4] For the protocol to work, we require that $k_{\min}$ divides k_i for all i, i.e., $k_{\min} \mid k_i$. This ensures that $\mathrm{GF}(q^{\delta_i}) \subseteq \mathrm{GF}(q^{\delta_{\max}})$. Furthermore, we require the codes $\mathcal{C}_i'$ to be such that $\mathcal{C}_i' \subseteq \mathcal{C}_{\max}'$. The protocol is characterized by the codes $\{\mathcal{C}_i'\}$ and by two other codes, $\bar{\mathcal{C}}$ and $\tilde{\mathcal{C}}$. Code $\bar{\mathcal{C}}$ (over $\mathrm{GF}(q)$) has parameters $(n, \bar{k})$ and characterizes the queries sent to the SBSs and the MBS, while code $\tilde{\mathcal{C}}$ (defined below) defines the responses sent back to the user from the SBSs and the MBS. The designed protocol achieves PIR against a number of colluding SBSs $T \leq d_{\min}^{\bar{\mathcal{C}}^\perp} - 1$, where $d_{\min}^{\bar{\mathcal{C}}^\perp}$ is the minimum Hamming distance of the dual code of $\bar{\mathcal{C}}$.

3.1 Query Construction

The queries must be constructed such that privacy is preserved and the user can retrieve the requested file from the n response vectors $\boldsymbol{r}^{(l)}$, $l = 1, \dots, n$. In partic-

[4] Without loss of generality, to simplify notation we assume that the last coordinates of the code are puntured.

ular, the protocol is designed such that the subresponses $r_j^{(l)}$, $l = 1, \ldots, n$, corresponding to the n subqueries $\boldsymbol{q}_j^{(1)}, \ldots, \boldsymbol{q}_j^{(n)}$ recover Γ unique code symbols of the file $\boldsymbol{X}^{(i)}$.

The queries are constructed as follows. The user chooses βF codewords $\bar{\boldsymbol{c}}_m^{(i)} = (c_{m,1}^{(i)}, \ldots, c_{m,n}^{(i)}) \in \bar{\mathcal{C}}$, $m = 1, \ldots, \beta$, $i = 1, \ldots, F$, independently and uniformly at random. Then, the user constructs n vectors,

$$\mathring{\boldsymbol{c}}_l = (\mathring{\boldsymbol{c}}_l^{(1)}, \ldots, \mathring{\boldsymbol{c}}_l^{(F)}), \quad l = 1, \ldots, n, \tag{V.3}$$

where $\mathring{\boldsymbol{c}}_l^{(i)}$ collects the l-th coordinates of the β codewords $\bar{\boldsymbol{c}}_m^{(i)}$, $m = 1, \ldots, \beta$, i.e., $\mathring{\boldsymbol{c}}_l^{(i)} = (\bar{c}_{1,l}^{(i)}, \ldots, \bar{c}_{\beta,l}^{(i)})$.

Assume that the user wants to retrieve file $\boldsymbol{X}^{(i)}$. Then, subquery $\boldsymbol{q}_j^{(l)}$ is constructed as

$$\boldsymbol{q}_j^{(l)} = \mathring{\boldsymbol{c}}_l + \boldsymbol{\delta}_j^{(l)}, \tag{V.4}$$

where

$$\boldsymbol{\delta}_j^{(l)} = \begin{cases} \boldsymbol{\omega}_{\beta(i-1)+s_j^{(l)}} & \text{if } l \in \mathcal{J}_j, \\ \boldsymbol{\omega}_0 & \text{otherwise,} \end{cases} \tag{V.5}$$

for some set $\mathcal{J}_j$ that will be defined below. Vector $\boldsymbol{\omega}_t$, $t = 1, \ldots, \beta F$, denotes the t-th (βF)-dimensional unit vector, i.e., the length-βF vector with a one in the t-th coordinate and zeroes in all other coordinates, and $\boldsymbol{\omega}_0$ the all-zero vector. The meaning of index $s_j^{(l)}$ will become apparent later.

According to (V.4), each subquery vector is the sum of two vectors, $\mathring{\boldsymbol{c}}_l$ and $\boldsymbol{\delta}_j^{(l)}$. The purpose of $\mathring{\boldsymbol{c}}_l$ is to make the subquery appear random and thus ensure privacy (i.e., Condition (V.2a)). On the other hand, the vectors $\boldsymbol{\delta}_j^{(l)}$ are deterministic vectors which must be properly constructed such that the user is able to retrieve the requested file from the response vectors (i.e., Condition (V.2b)). Similar to Protocol 3 in [21], the vectors $\boldsymbol{\delta}_j^{(l)}$ are constructed from a $d \times n$ binary matrix $\hat{\boldsymbol{E}}$ where each row represents a weight-Γ erasure pattern that is correctable by $\tilde{\mathcal{C}}$ and where the weights of its columns are determined from β information sets $\mathcal{I}_m$, $m = 1, \ldots, \beta$, of $\mathcal{C}'_{\max}$.

The construction of $\hat{\boldsymbol{E}}$ is addressed below. We define the set $\mathcal{F}_l$ as the index set of information sets $\mathcal{I}_m$ that contain the l-th coordinate of $\mathcal{C}'_{\max}$, i.e., $\mathcal{F}_l = \{m : l \in \mathcal{I}_m\}$. To allow the user to recover the requested file from the response vectors, $\hat{\boldsymbol{E}}$ is constructed such that it satisfies the following conditions.

C1. The user should be able to recover Γ unique code symbols of the requested file $\boldsymbol{X}^{(i)}$ from the responses to each set of n subqueries $\boldsymbol{q}_j^{(l)}$, $l = 1, \ldots, n$. This is to say that each row of $\hat{\boldsymbol{E}}$ should have exactly Γ ones. We denote by $\mathcal{J}_j$ the support of the j-th row of $\hat{\boldsymbol{E}}$.

C2. The user should be able to recover $\Gamma d \geq \beta k_i$ unique code symbols of the requested file $\boldsymbol{X}^{(i)}$, at least k_i symbols from each stripe. This means that each row $\hat{\boldsymbol{e}}_j = (\hat{e}_{j,1}, \ldots, \hat{e}_{j,n})$, $j = 1, \ldots, d$, of $\hat{\boldsymbol{E}}$ should correspond to an erasure pattern that is correctable by $\tilde{\mathcal{C}}$.

C3. Let $\boldsymbol{t}_l$, $l = 1, \ldots, n$, be the l-th column vector of $\hat{\boldsymbol{E}}$. The protocol should be able to recover $w_{\mathsf{H}}(\boldsymbol{t}_l)$ unique code symbols from the l-th response vector, which means that it is required that $w_{\mathsf{H}}(\boldsymbol{t}_l) = |\mathcal{F}_l|$. We call the vector $(w_{\mathsf{H}}(\boldsymbol{t}_1), \ldots, w_{\mathsf{H}}(\boldsymbol{t}_n))$ the *column weight profile of* $\hat{\boldsymbol{E}}$.

Finally, from $\hat{\boldsymbol{E}}$ we construct the vectors $\boldsymbol{\delta}_j^{(l)}$ in (V.5). In particular, index $s_j^{(l)}$ in (V.5) is such that $s_j^{(l)} \in \mathcal{F}_l$ and $s_j^{(l)} \neq s_{j'}^{(l)}$ for $j \neq j'$, $j, j' = 1, \ldots, d$.

3.2 Response Vectors

The j-th subresponse corresponding to subquery $\boldsymbol{q}_j^{(l)}$, $j = 1, \ldots, d$, is (see (V.1))

$$r_j^{(l)} = \langle \boldsymbol{q}_j^{(l)}, (c_{1,l}^{(1)}, \ldots, c_{\beta,l}^{(F)}) \rangle.$$

The user collects the n subresponses $r_j^{(l)}$, $l = 1, \ldots n$, in the vector $\boldsymbol{\rho}_j$,

$$\boldsymbol{\rho}_j = \begin{pmatrix} r_j^{(1)} \\ r_j^{(2)} \\ \vdots \\ r_j^{(n)} \end{pmatrix} = \sum_{m=1}^{\beta} \underbrace{\begin{pmatrix} \bar{c}_{m,1}^{(1)} c_{m,1}^{(1)} \\ \bar{c}_{m,2}^{(1)} c_{m,2}^{(1)} \\ \vdots \\ \bar{c}_{m,n}^{(1)} c_{m,n}^{(1)} \end{pmatrix}}_{\in \left\{ \boldsymbol{x} \in (\mathrm{GF}(q^{\delta_{\max}}))^n : \boldsymbol{H}^{\mathcal{C}_1' \circ \bar{\mathcal{C}}} \boldsymbol{x} = \boldsymbol{0} \right\}} + \underbrace{\begin{pmatrix} \bar{c}_{m,1}^{(2)} c_{m,1}^{(2)} \\ \bar{c}_{m,2}^{(2)} c_{m,2}^{(2)} \\ \vdots \\ \bar{c}_{m,n}^{(2)} c_{m,n}^{(2)} \end{pmatrix}}_{\in \left\{ \boldsymbol{x} \in (\mathrm{GF}(q^{\delta_{\max}}))^n : \boldsymbol{H}^{\mathcal{C}_2' \circ \bar{\mathcal{C}}} \boldsymbol{x} = \boldsymbol{0} \right\}} + \cdots$$

$$+ \underbrace{\begin{pmatrix} \bar{c}_{m,1}^{(F)} c_{m,1}^{(F)} \\ \bar{c}_{m,2}^{(F)} c_{m,2}^{(F)} \\ \vdots \\ \bar{c}_{m,n}^{(F)} c_{m,n}^{(F)} \end{pmatrix}}_{\in \left\{ \boldsymbol{x} \in (\mathrm{GF}(q^{\delta_{\max}}))^n : \boldsymbol{H}^{\mathcal{C}_{\max}' \circ \bar{\mathcal{C}}} \boldsymbol{x} = \boldsymbol{0} \right\}} + \begin{pmatrix} o_j^{(1)} \\ o_j^{(2)} \\ \vdots \\ o_j^{(n)} \end{pmatrix}, \tag{V.6}$$

where symbol $o_j^{(l)}$ represents the code symbol from file $\boldsymbol{X}^{(i)}$ downloaded in the j-th subresponse from the l-th response vector. Due to the structure of the queries obtained from $\hat{\boldsymbol{E}}$, the user retrieves Γ code symbols from the set of n subresponses to the j-th subqueries. Consider a retrieval code $\tilde{\mathcal{C}}$ of the form

$$\tilde{\mathcal{C}} = \sum_{i=1}^{F} \mathcal{C}_i' \circ \bar{\mathcal{C}} \overset{(a)}{=} \left(\sum_{i=1}^{F} \mathcal{C}_i' \right) \circ \bar{\mathcal{C}}, \tag{V.7}$$

where $\mathcal{C}_i' + \mathcal{C}_j'$ denotes the sum of subspaces $\mathcal{C}_i'$ and $\mathcal{C}_j'$, resulting in the set consisting of all elements $\boldsymbol{c} + \boldsymbol{c}'$ for any $\boldsymbol{c} \in \mathcal{C}_i'$ and $\boldsymbol{c}' \in \mathcal{C}_j'$, and where (a) follows due to the fact that the Hadamard product is distributive over addition.

The symbols requested by the user are then obtained solving the system of linear equations defined by

$$\boldsymbol{H}^{\tilde{\mathcal{C}}}\boldsymbol{\rho}_j = \boldsymbol{H}^{\tilde{\mathcal{C}}}\begin{pmatrix} o_j^{(1)} \\ o_j^{(2)} \\ \vdots \\ o_j^{(n)} \end{pmatrix}.$$

3.3 Privacy

For the retrieval, we require $\tilde{\mathcal{C}}$ to be a valid code, i.e., it must have a code rate strictly less than 1. For a given number of colluding SBSs T, the combination of conditions on $\bar{\mathcal{C}}$ and $\tilde{\mathcal{C}}$ restricts the choice for the underlying storage codes $\{\mathcal{C}_i\}$. In the following theorem, we present a family of MDS codes, namely generalized Reed-Solomon (GRS) codes, that work with the protocol. A GRS code $\mathcal{C}$ over $\mathrm{GF}(q)$ of length n and dimension k is a weighted polynomial evaluation code of degree k defined by some weighting vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in (\mathrm{GF}(q)^{\times})^n$ and evaluation vector $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_n) \in (\mathrm{GF}(q)^{\times})^n$ satisfying $\kappa_i \neq \kappa_j$ for all $i \neq j$ [25, Ch. 5]. In the sequel, we refer to $(n, k, \boldsymbol{v}, \boldsymbol{\kappa})$ as the parameters of a GRS code $\mathcal{C}$.

Lemma 1. *Given an $(n, k_{\max}, \boldsymbol{v}, \boldsymbol{\kappa})$ GRS code $\mathcal{C}_{\max}$, for all $k < k_{\max}$, there exists an $(n, k, \boldsymbol{v}, \boldsymbol{\kappa})$ GRS code that is a subcode of $\mathcal{C}_{\max}$.*

Proof. The canonical generator matrix for an $(n, k_{\max}, \boldsymbol{v}, \boldsymbol{\kappa})$ GRS code $\mathcal{C}_{\max}$ is given by

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ \kappa_1 & \kappa_2 & \ldots & \kappa_n \\ \vdots & \vdots & \ldots & \vdots \\ \kappa_1^{k_{\max}-1} & \kappa_2^{k_{\max}-1} & \ldots & \kappa_n^{k_{\max}-1} \end{pmatrix}\begin{pmatrix} v_1 & 0 & \ldots & 0 \\ 0 & v_2 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & v_n \end{pmatrix}. \tag{V.8}$$

Clearly, taking the first k rows of the leftmost matrix of (V.8) and multiplying it with the rightmost diagonal matrix generates an (n, k) subcode of $\mathcal{C}_{\max}$ which by itself is an $(n, k, \boldsymbol{v}, \boldsymbol{\kappa})$ GRS code with the same weighting vector $\boldsymbol{v}$. Thus, GRS codes are naturally nested, and the result follows. □

Theorem 1. *Let $\mathcal{C}_{\mathrm{MDS}}^{\boldsymbol{\mu}}$ be a caching scheme with GRS codes $\{\mathcal{C}_i\}$ of parameters $(N_{\mathrm{SBS}}, k_i, \boldsymbol{v}, (\kappa_1, \ldots, \kappa_{N_{\mathrm{SBS}}}))$ and let $\mathcal{C}_i'$ be the (n, k_i) code obtained by puncturing $\mathcal{C}_i$. Also, let $\bar{\mathcal{C}}$ be an $(n, T, \bar{\boldsymbol{v}}, (\kappa_1, \ldots, \kappa_n))$ GRS code. Then, for $\beta = \Gamma = n - (k_{\max} + T - 1)$ and $d = k_{\max}$, the protocol achieves PIR against T colluding SBSs.*

Proof. The proof is given in the appendix. □

Note that the retrieval code $\bar{\mathcal{C}}$ depends on the n SBSs within visibility that are contacted by the user through its evaluation vector. Finally, we remark that, with some slight modifications, the proposed protocol can be adapted to work with non-MDS codes.

3.4 Example

As an example, consider the case of $F = 2$ files, $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$, both of size βL bits. The first file $\boldsymbol{X}^{(1)}$ is stored in the SBSs according to Fig. V.2 using an $(N_{\mathsf{SBS}} = 6, k_1 = 1)$ binary repetition code $\mathcal{C}_1$. Similarly, the second file $\boldsymbol{X}^{(2)}$ is stored (again according to Fig. V.2) using an $(N_{\mathsf{SBS}} = 6, k_2 = 5)$ binary single parity-check code $\mathcal{C}_2$. Assume $n = N_{\mathsf{SBS}} = 6$ (i.e., no puncturing) and that none of the SBSs collude, i.e., $T = 1$. Furthermore, we assume that the user wants to retrieve $\boldsymbol{X}^{(1)}$ and is able to contact $b = n = 6$ SBS (i.e., we consider the extreme case where the user is not contacting the MBS). According to Theorem 1, we can choose $\beta = \Gamma = n - (k_{\mathsf{max}} + T - 1) = 6 - (5 + 1 - 1) = 1$ and $d = k_{\mathsf{max}} = 5$. Finally, we choose $\bar{\mathcal{C}}$ as an $(n = 6, T = 1)$ binary repetition code.

According to (V.7), the retrieval code $\tilde{\mathcal{C}} = (\mathcal{C}_1 + \mathcal{C}_2) \circ \bar{\mathcal{C}} = \mathcal{C}_1 + \mathcal{C}_2 = \mathcal{C}_2$ and can be generated by

$$\boldsymbol{G}^{\tilde{\mathcal{C}}} = \boldsymbol{G}^{\mathcal{C}_2} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Moreover, let

$$\hat{\boldsymbol{E}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \mathcal{I}_1 = \{1, 2, 3, 4, 5\},$$

where $\mathcal{I}_1$ is an information set of $\mathcal{C}_{\mathsf{max}} = \mathcal{C}_2$ (the submatrix $\boldsymbol{G}_{\mathcal{I}_1}^{\mathcal{C}_2}$ has rank $k_2 = 5$). Note that $\hat{\boldsymbol{E}}$ satisfies all three conditions C1–C3 and has column weight profile $(1, 1, 1, 1, 1, 0) = (|\mathcal{F}_1|, \dots, |\mathcal{F}_6|)$.

Query Construction. The user generates $\beta F = 2$ codewords $\bar{\boldsymbol{c}}_1^{(1)}$ and $\bar{\boldsymbol{c}}_1^{(2)}$ independently and uniformly at random from $\bar{\mathcal{C}}$. Without loss of generality, let $\bar{\boldsymbol{c}}_1^{(1)} = \bar{\boldsymbol{c}}_1^{(2)} = (1, \dots, 1)$. Next, the $n = 6$ subqueries $q_1^{(l)}$, $l = 1, \dots, 6$, are constructed according to (V.4), (V.5) as

$$\boldsymbol{q}_1^{(l)} = \begin{cases} \mathring{\boldsymbol{c}}_l + (1, 0) & \text{if } l = 1, \\ \mathring{\boldsymbol{c}}_l + (0, 0) & \text{otherwise,} \end{cases}$$

where $\mathring{\boldsymbol{c}}_l$ is defined in (V.3).

File Retrieval. Consider the $n = 6$ subresponses $r_1^{(l)}$, $l = 1, \dots, 6$. Then, accord-

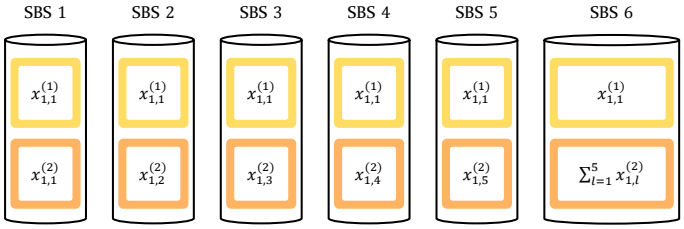

Figure V.2: Wireless caching scenario in which there are $N_{\mathrm{SBS}} = 6$ SBSs. The SBSs store $F = 2$ files, $\boldsymbol{X}^{(1)} = (x_{1,1}^{(1)}) \in \mathrm{GF}(2^5)^{1\times 1}$ and $\boldsymbol{X}^{(2)} = (x_{1,1}^{(2)}, x_{1,2}^{(2)}, x_{1,3}^{(2)}, x_{1,4}^{(2)}, x_{1,5}^{(2)}) \in \mathrm{GF}(2)^{1\times 5}$, of $\beta L = 5$ bits each. The first file $\boldsymbol{X}^{(1)}$ is encoded using an $(N_{\mathrm{SBS}} = 6, k_1 = 1)$ binary repetition code $\mathcal{C}_1$, while the second file $\boldsymbol{X}^{(2)}$ is encoded using an $(N_{\mathrm{SBS}} = 6, k_2 = 5)$ binary single parity-check code $\mathcal{C}_2$.

ing to (V.6),

$$\boldsymbol{\rho}_1 = \begin{pmatrix} r_1^{(1)} \\ r_1^{(2)} \\ r_1^{(3)} \\ r_1^{(4)} \\ r_1^{(5)} \\ r_1^{(6)} \end{pmatrix} = \underbrace{\begin{pmatrix} \bar{c}_{1,1}^{(1)} c_{1,1}^{(1)} \\ \bar{c}_{1,2}^{(1)} c_{1,2}^{(1)} \\ \vdots \\ \bar{c}_{1,6}^{(1)} c_{1,6}^{(1)} \end{pmatrix}}_{\in \left\{\boldsymbol{x}\in(\mathrm{GF}(2^5))^n :\ \boldsymbol{H}^{\mathcal{C}_1'\circ\bar{\mathcal{C}}}\boldsymbol{x}=\boldsymbol{0}\right\}} + \underbrace{\begin{pmatrix} \bar{c}_{1,1}^{(2)} c_{1,1}^{(2)} \\ \bar{c}_{1,2}^{(2)} c_{1,2}^{(2)} \\ \vdots \\ \bar{c}_{1,6}^{(2)} c_{1,6}^{(2)} \end{pmatrix}}_{\in \left\{\boldsymbol{x}\in(\mathrm{GF}(2^5))^n :\ \boldsymbol{H}^{\mathcal{C}_2'\circ\bar{\mathcal{C}}}\boldsymbol{x}=\boldsymbol{0}\right\}} + \begin{pmatrix} o_1^{(1)} \\ o_1^{(2)} \\ \vdots \\ o_1^{(6)} \end{pmatrix}$$

$$= \begin{pmatrix} x_{1,1}^{(1)} \\ x_{1,1}^{(1)} \\ x_{1,1}^{(1)} \\ x_{1,1}^{(1)} \\ x_{1,1}^{(1)} \\ x_{1,1}^{(1)} \end{pmatrix} + \begin{pmatrix} x_{1,1}^{(2)} \\ x_{1,2}^{(2)} \\ x_{1,3}^{(2)} \\ x_{1,4}^{(2)} \\ x_{1,5}^{(2)} \\ \sum_{l=1}^{5} x_{1,l}^{(2)} \end{pmatrix} + \begin{pmatrix} x_{1,1}^{(1)} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and the code symbol $x_{1,1}^{(1)}$ of the file $\boldsymbol{X}^{(1)}$ is recovered from

$$\boldsymbol{H}^{\bar{\mathcal{C}}}\boldsymbol{\rho}_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_{1,1}^{(1)} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = x_{1,1}^{(1)}.$$

Note that in order to retain privacy across the two files of the library, we need to send $d = k_{\max} = 5$ subqueries to each SBS, thus generating 5 subresponses from each SBS (even if the first file can be recovered from the $n = 6$ subresponses $r_1^{(l)}$, $l = 1, \ldots, 6$).

4 Backhaul Rate Analysis: No PIR Case

In this section, we derive the backhaul rate for the proposed caching scheme for the case of no PIR, i.e., the conventional caching scenario where PIR is not required.

Proposition 6. *The average backhaul rate for the caching scheme $\mathcal{C}^{\boldsymbol{\mu}}_{\mathsf{MDS}}$ in Section 2 for the case of no PIR is*

$$\mathrm{R}_{\mathsf{noPIR}} = \sum_{i=1}^{F} p_i \lceil \mu_i \rceil \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \max\left(0, 1/\mu_i - b\right) \mu_i + \sum_{i=1}^{F} p_i \lfloor 1 - \mu_i \rfloor. \tag{V.9}$$

Proof. To download file $\boldsymbol{X}^{(i)}$, if the user is in communication range of a number of SBSs, b, larger than or equal to $1/\mu_i$, the user can retrieve the file from the SBSs and there is no contribution to the backhaul rate. Otherwise, if $b < 1/\mu_i$, the user retrieves a fraction $L/k_i = L\mu_i$ of the file from each of the b SBSs, i.e., a total of $b\beta L\mu_i$ bits, and downloads the remaining $(1/\mu_i - b)\beta L\mu_i$ bits from the MBS. Averaging over $\boldsymbol{\gamma}$ and $\boldsymbol{p}$ (for the files cached) and normalizing by the file size βL, the contribution to the backhaul rate of the retrieval of files that are cached in the SBSs is

$$\sum_{i=1}^{F} p_i \lceil \mu_i \rceil \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \max\left(0, 1/\mu_i - b\right) \mu_i. \tag{V.10}$$

On the other hand, the files that are not cached are retrieved completely from the MBS, and their contribution to the backhaul rate is

$$\sum_{i=1}^{F} p_i \lfloor 1 - \mu_i \rfloor. \tag{V.11}$$

Combining (V.10) and (V.11) completes the proof. □

We denote by $\mathrm{R}^*_{\mathsf{noPIR}}$ the maximum PIR rate resulting from the optimization of the content placement. $\mathrm{R}^*_{\mathsf{noPIR}}$ can be obtained solving the following optimization problem,

$$\begin{aligned} \mathrm{R}^*_{\mathsf{noPIR}} = \min_{\mu_i \in \mathcal{M}'} \quad & \sum_{i=1}^{F} p_i \lceil \mu_i \rceil \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \max\left(0, 1/\mu_i - b\right) \mu_i + \sum_{i=1}^{F} p_i \lfloor 1 - \mu_i \rfloor \\ \text{s.t.} \quad & \sum_{i=1}^{F} \mu_i \leq M, \end{aligned}$$

where $\mathcal{M}' = \mathcal{M} \cup \{1/N_{\mathsf{SBS}}\}$, as $\mu_i = 1/N_{\mathsf{SBS}}$ is a valid value for the case where PIR is not required.

In the following lemma, we show that the proposed content placement is equivalent to the one in [7], in the sense that it yields the same average backhaul rate.

Lemma 2. *The average backhaul rate given by* (V.9) *for the caching scheme* $\mathcal{C}^{\boldsymbol{\mu}}_{\mathsf{MDS}}$ *in Section 2 is equal to the one given by the caching scheme in [7], i.e., the two content placements are equivalent.*

Proof. We can rewrite (V.9) using simple math as

$$\begin{aligned}
\mathrm{R}_{\mathsf{noPIR}} &= \sum_{i=1}^{F} p_i \lceil \mu_i \rceil \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \max\big(0, 1/\mu_i - b\big)\mu_i + \sum_{i=1}^{F} p_i \lfloor 1-\mu_i \rfloor \\
&= \sum_{i=1}^{F} p_i \lceil \mu_i \rceil \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \max\big(0, 1 - b\mu_i\big) + \sum_{i=1}^{F} p_i \lfloor 1-\mu_i \rfloor \\
&= \sum_{i=1}^{F} p_i \lceil \mu_i \rceil \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \big(1 - \min\big(1, b\mu_i\big)\big) + \sum_{i=1}^{F} p_i \lfloor 1-\mu_i \rfloor \\
&\overset{(a)}{=} \sum_{i=1}^{F} p_i (\lceil \mu_i \rceil + \lfloor 1-\mu_i \rfloor) \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \big(1 - \min\big(1, b\mu_i\big)\big) \\
&= \sum_{i=1}^{F} p_i \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \big(1 - \min\big(1, b\mu_i\big)\big),
\end{aligned}$$

which is the expression in [7, eq. (1)]. (a) follows from the fact that we can write $p_i \lfloor 1-\mu_i \rfloor$ as $p_i \lfloor 1-\mu_i \rfloor \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \big(1 - \min\big(1, b\mu_i\big)\big)$. For $0 < \mu_i \leq 1$ both expressions are zero, while for $\mu_i = 0$ both expressions boil down to p_i as $p_i \lfloor 1-\mu_i \rfloor \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b \big(1 - \min\big(1, b\mu_i\big)\big) = p_i \sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b$ and $\sum_{b=0}^{N_{\mathsf{SBS}}} \gamma_b = 1$. □

For popular content placement, i.e., the case where the M most popular files are cached in all SBSs (this corresponds to caching the M most popular files using an $(N_{\mathsf{SBS}}, 1)$ repetition code, i.e., $\mu_i = 1$ for $i \leq M$ and $\mu_i = 0$ for $i > M$), the backhaul rate is given by

$$\mathrm{R}^{\mathsf{pop}}_{\mathsf{noPIR}} = \gamma_0 \sum_{i=1}^{M} p_i + \sum_{i=M+1}^{F} p_i. \tag{V.12}$$

5 Backhaul Rate Analysis: PIR Case

In this section, we derive the backhaul rate for the case of PIR (i.e., when the user wishes to download content privately) and we prove that uniform content placement (under the PIR protocol in Section 3 with GRS codes) is optimal. The average backhaul rate is given in the following proposition.

Proposition 7. *The average backhaul rate for the caching scheme* $\mathcal{C}^{\boldsymbol{\mu}}_{\mathsf{MDS}}$ *in Section 2*

(with GRS codes) for the PIR case is

$$\mathsf{R}_{\mathsf{PIR}} = \frac{\mu_{\max}}{\mu_{\min}(n-T+1)-1}\sum_{i=1}^{F} p_i\lceil\mu_i\rceil\sum_{b=0}^{n}\gamma_b(n-b) + \sum_{i=1}^{F} p_i\lfloor 1-\mu_i\rfloor. \tag{V.13}$$

Proof. To download file $\boldsymbol{X}^{(i)}$, the user generates n query matrices. If the user is in communication range of b SBSs, it receives b responses (one from each SBS). The responses to the remaining $n-b$ query matrices need to be downloaded from the MBS. Since each response consists of d subresponses of size $L\mu_{\max}$ bits, the user downloads $(n-b)dL\mu_{\max}$ bits from the MBS. Averaging over $\boldsymbol{\gamma}$ and $\boldsymbol{p}$ (for the files cached) and normalizing by the file size βL, the contribution to the backhaul rate of the retrieval of files that are cached in the SBSs is

$$\frac{1}{\beta}\sum_{i=1}^{F} p_i\lceil\mu_i\rceil\sum_{b=0}^{n}\gamma_b(n-b)d\mu_{\max}. \tag{V.14}$$

Now, using the fact that $\beta = \Gamma = n-(k_{\max}+T-1) = \frac{\mu_{\min}(n-T+1)-1}{\mu_{\min}}$ and $d = k_{\max} = 1/\mu_{\min}$ (see Theorem 1), we can rewrite (V.14) as

$$\frac{\mu_{\max}}{\mu_{\min}(n-T+1)-1}\sum_{i=1}^{F} p_i\lceil\mu_i\rceil\sum_{b=0}^{n}\gamma_b(n-b). \tag{V.15}$$

On the other hand, the files that are not cached are retrieved completely from the MBS, and their contribution to the backhaul rate is (as for the no PIR case)

$$\sum_{i=1}^{F} p_i\lfloor 1-\mu_i\rfloor. \tag{V.16}$$

Combining (V.15) and (V.16) completes the proof. □

5.1 Optimal Content Placement

Let $\mathsf{R}^*_{\mathsf{PIR}}$ be the maximum PIR rate resulting from the optimization of the content placement. $\mathsf{R}^*_{\mathsf{PIR}}$ can be obtained solving the following optimization problem,

$$\mathsf{R}^*_{\mathsf{PIR}} = \min_{\substack{\mu_i\in\mathcal{M}\\ n\in\mathcal{A}}} \frac{\mu_{\max}}{\mu_{\min}(n-T+1)-1}\sum_{i=1}^{F} p_i\lceil\mu_i\rceil\sum_{b=0}^{n}\gamma_b(n-b) + \sum_{i=1}^{F} p_i\lfloor 1-\mu_i\rfloor \tag{V.17}$$
$$\text{s.t.} \sum_{i=1}^{F}\mu_i \le M \text{ and } k_{\min} \mid k_i,$$

where $\mathcal{A} = \{1/\mu_{\min}+T, \ldots, N_{\mathsf{SBS}}\}$ and the minimum value that n can take on, i.e., $1/\mu_{\min}+T$, comes from the fact that $\mu_{\min}(n-T+1)-1$ has to be positive.

Lemma 3. *Uniform content allocation, i.e., $\mu_i = \mu$ for all files that are cached, is optimal. Furthermore, the optimal number of files to cache is the maximum possible, i.e., $\mu_i = \mu$ for $i \leq \min(M/\mu, F)$.*

Proof. We first prove the first part of the lemma. We need to show that either the optimal solution to the optimization problem in (V.17) is the all-zero vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_F) = (0, \ldots, 0)$, or there exists a nonzero optimal solution $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_F)$ for which $\mu_{\max} = \mu_{\min}$. Consider the second case, and let $\boldsymbol{\mu}$ denote any nonzero feasible solution to (V.17), i.e., a nonzero solution that satisfies the cache size constraint. Furthermore, let $\boldsymbol{\mu}' = (\mu'_1, \ldots, \mu'_F)$ denote the length-F vector obtained from $\boldsymbol{\mu}$ as $\mu'_i = \mu_{\min}$ for $\mu_i \neq 0$ and $\mu'_i = 0$ otherwise. Clearly, $\boldsymbol{\mu}'$ satisfies the cache size constraint as well. Note that $\mu'_{\max} = \mu'_{\min} = \mu_{\min}$. Thus,

$$\frac{\mu'_{\max}}{\mu'_{\min}(n-T+1)-1} = \frac{\mu_{\min}}{\mu_{\min}(n-T+1)-1} \leq \frac{\mu_{\max}}{\mu_{\min}(n-T+1)-1}.$$

Furthermore, since both the double summation in the first term of the objective function in (V.17) and the second term in (V.17) only depend on the support of $\boldsymbol{\mu}$, it follows that the value of the objective function for $\boldsymbol{\mu}'$ is smaller than or equal to the value of the objective function for $\boldsymbol{\mu}$. Thus, for any nonzero feasible solution $\boldsymbol{\mu}$ there exists another at least as good nonzero feasible solution $\boldsymbol{\mu}'$ for which all nonzero entries are the same (i.e., $\mu'_{\min} = \mu'_{\max} = \mu$), and the result follows by applying the above procedure to a (nonzero) optimal solution to (V.17).

We now prove the second part of the lemma. Caching a file helps in reducing the backhaul rate if

$$\frac{\mu}{\mu(n-T+1)-1} \sum_{b=0}^{n} \gamma_b (n-b) < 1, \tag{V.18}$$

for some $n \in \mathcal{A}$ and $\mu \in \mathcal{M}$. This is independent of the file index i. Thus, if the optimal solution is to cache at least one file ($\boldsymbol{\mu} \neq \mathbf{0}$), (V.18) is met for some $n \in \mathcal{A}$ and caching other files (as many files as permitted up to the cache size constraint, with decreasing order of popularity) is optimal as it further reduces the backhaul rate. □

Following Lemma 3, the optimization problem in (V.17) can be rewritten as

$$\mathrm{R}^*_{\mathrm{PIR}} = \min_{\substack{\mu \in \mathcal{M} \\ n \in \mathcal{A}}} \frac{\mu}{\mu(n-T+1)-1} \sum_{i=1}^{\min(M/\mu, F)} p_i \sum_{b=0}^{n} \gamma_b (n-b) + \sum_{i=M/\mu+1}^{F} p_i. \tag{V.19}$$

5.2 Popular Content Placement

For popular content placement, the backhaul rate is given by

$$\mathrm{R}^{\mathrm{pop}}_{\mathrm{PIR}} = \min_{n \in \mathcal{A}} \frac{1}{n-T} \sum_{i=1}^{M} p_i \sum_{b=0}^{n} \gamma_b (n-b) + \sum_{i=M+1}^{F} p_i. \tag{V.20}$$

Note that the optimization over n is still required.

6 Weighted Communication Rate

So far, we have considered only the backhaul rate. However, it might also be desirable to limit the communication rate from SBSs to the user. We thus consider the weighted communication rate, $\mathsf{C}_{\mathsf{PIR}}$, defined as[5]

$$\mathsf{C}_{\mathsf{PIR}} = \mathsf{R}_{\mathsf{PIR}} + \theta \mathsf{D}_{\mathsf{PIR}},$$

where $\mathsf{D}_{\mathsf{PIR}}$ is the average communication rate (normalized by the file size βL) from the SBSs, and θ is a weighting parameter. We consider $\theta \leq 1$, stemming from the fact that the bottleneck is the backhaul. Note that minimizing the average backhaul rate corresponds to $\theta = 0$.

Proposition 8. *The average communication rate from the SBSs for the caching scheme $\mathcal{C}^{\boldsymbol{\mu}}_{\mathsf{MDS}}$ in Section 2 (with GRS codes) for the PIR case is*

$$\mathsf{D}_{\mathsf{PIR}} = \frac{\mu_{\max}}{\mu_{\min}(n-T+1)-1} \sum_{b=0}^{n} \tilde{\gamma}_b b, \tag{V.21}$$

where $\tilde{\gamma}_b = \gamma_b$ for $b < n$ and $\tilde{\gamma}_n = \sum_{b=n}^{N_{\mathsf{SBS}}} \gamma_b$.

Proof. To ensure privacy, the user needs to download data from the SBSs within visibility regardless whether the requested file is cached or not. This is in contrast to the case of no PIR. Note that, if the user queries the SBSs only in the case the requested file is cached, then the spy SBSs would infer that the user is interested in one of the files cached, thus gaining some information about the file requested. In other words, the user sends dummy queries and downloads data that is useless for the retrieval of the file but is necessary to achieve privacy. The user receives b responses from the b SBSs within communication range, each of size $dL\mu_{\max}$ bits. Let $\tilde{\gamma}_b$ denote the probability to receive responses from b SBSs. For $b < n$, $\tilde{\gamma}_b$ is equal to the probability that b SBSs are within communication range, i.e., $\tilde{\gamma}_b = \gamma_b$. On the other hand, the probability to receive responses from n SBSs, $\tilde{\gamma}_n$, is the probability that at least n SBSs are within communication range, i.e., $\tilde{\gamma}_n = \sum_{b=n}^{N_{\mathsf{SBS}}} \gamma_b$. Averaging over $\tilde{\boldsymbol{\gamma}}$ and $\boldsymbol{p}$ (for all files, cached and not cached) and normalizing by the file size βL, the contribution to the communication rate of the retrieval of a file from the SBSs is

$$\frac{1}{\beta} \sum_{i=1}^{F} p_i \sum_{b=0}^{n} \tilde{\gamma}_b b d \mu_{\max}. \tag{V.22}$$

Now, using the fact that $\beta = \Gamma = n - (k_{\max} + T - 1) = \frac{\mu_{\min}(n-T+1)-1}{\mu_{\min}}$ and $d = k_{\max} = 1/\mu_{\min}$ (see Theorem 1), we can rewrite (V.22) as (V.21). □

[5]For the case of no PIR, a linear scalarization of the MBS and SBS download delays was considered in [5]. The communication rate is directly related to the download delay.

The corresponding optimization problem is

$$\mathrm{C}^*_{\mathsf{PIR}} = \min_{\substack{\mu_i \in \mathcal{M} \\ n \in \mathcal{A}}} \mathrm{R}_{\mathsf{PIR}} + \theta \mathrm{D}_{\mathsf{PIR}} \qquad \text{(V.23)}$$
$$\text{s.t.} \sum_{i=1}^{F} \mu_i \leq M \text{ and } k_{\min} \mid k_i,$$

where $\mathrm{R}_{\mathsf{PIR}}$ is given in (V.13).

Lemma 4. *Uniform content allocation, i.e., $\mu_i = \mu$ for all files that are cached, is optimal. Furthermore, the optimal number of files to cache is the maximum possible, i.e., $\mu_i = \mu$ for $i \leq \min(M/\mu, F)$.*

Proof. The proof of Lemma 3 applies to both terms in (V.23) and the result follows. □

Following Lemma 4, the optimization problem in (V.23) can be rewritten as

$$\mathrm{C}^*_{\mathsf{PIR}} = \min_{\substack{\mu \in \mathcal{M} \\ n \in \mathcal{A}}} \frac{\mu}{\mu(n-T+1)-1} \sum_{i=1}^{\min(M/\mu,F)} p_i \sum_{b=0}^{n} \gamma_b (n-b)$$
$$+ \sum_{i=M/\mu+1}^{F} p_i + \theta \frac{\mu}{\mu(n-T+1)-1} \sum_{b=0}^{n} \tilde{\gamma}_b b. \qquad \text{(V.24)}$$

7 Numerical Results

For the numerical results in this section, we assume that the files popularity distribution $\boldsymbol{p}$ follows the Zipf law [26], i.e., the popularity of file $\boldsymbol{X}^{(i)}$ is

$$p_i = \frac{1/i^{\alpha}}{\sum_{\ell} 1/\ell^{\alpha}},$$

where $\alpha \in [0.5, 1.5]$ is the skewness factor [7] and by definition $p_1 \geq p_2 \geq \ldots \geq p_F$. In Figs. V.3 and V.4, we consider a network topology where SBSs are deployed over a macro-cell of radius D meters according to a regular grid with distance d meters between them [5, 7]. Each SBS has a communication radius of r meters. Let $\mathcal{R}_b$ be the area where a user can be served by b SBSs. Then, assuming that the users are uniformly distributed over the macro-cell area with density ϕ users per square meter, the probability that a user is in communication range of b SBSs can be calculated as in [7]

$$\gamma_b = \frac{\phi \mathcal{R}_b}{\phi \sum_{a=1}^{N_{\max}} \mathcal{R}_a},$$

where the areas $\mathcal{R}_b$ can be easily obtained by simple geometrical evaluations, and $N_{\max}$ is the maximum number of SBSs within communication range of a user.

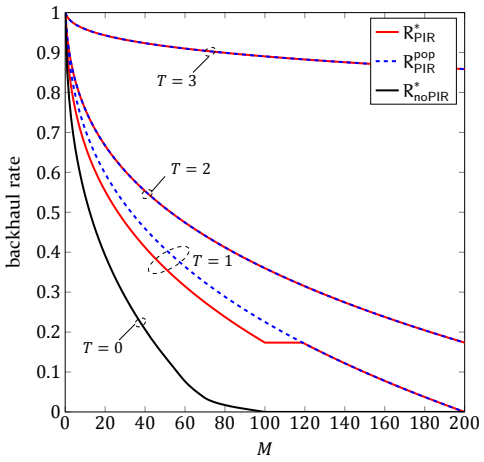

Figure V.3: Backhaul rate as a function of the cache size constraint M for a system with $F = 200$ files, $N_{\mathsf{SBS}} = 316$, and $\alpha = 0.7$.

For the results in Figs. V.3 and V.4, the system parameters (taken from [7]) are $D = 500$ meters, which results in $N_{\mathsf{SBS}} = 316$ over the macro-cell area, $F = 200$ files, $\alpha = 0.7$, and $r = 60$ meters. This results in $\boldsymbol{\gamma} = (0, 0, 0.1736, 0.5113, 0.3151, 0, \ldots, 0)$, i.e., the maximum number of SBSs in visibility of a user is $N_{\mathsf{max}} = 4$.

In Fig. V.3, we plot the optimized backhaul rate $\mathsf{R}^*_{\mathsf{PIR}}$ (red, solid lines) according to (V.19) as a function of the cache size constraint M for the noncolluding case ($T = 1$) and $T = 2$ and $T = 3$ colluding SBSs. The curves in Fig. V.3 should be interpreted as the minimum backhaul rate that is necessary in order to achieve privacy against T spy SBSs out of the n SBSs that are contacted by the user. For the particular system parameters considered, the optimal value of n is 3 for $T = 1$ and $T = 2$, and all values of M, i.e., the scheme yields privacy against T spy SBSs out of the $n = 3$ SBSs contacted. For $T = 3$ the optimal value of n is 4 for all values of M, and thus the scheme yields privacy against 3 spy SBSs out of $n = 4$ SBSs. We also plot the optimized backhaul rate $\mathsf{R}^*_{\mathsf{noPIR}}$ for the case of no PIR.[6] As can be seen in the figure, caching helps in significantly reducing the backhaul rate for $T = 1$ and $T = 2$. For $T = 3$ caching also helps in reducing the backhaul rate, but the reduction is smaller. Also, as expected, compared to the case of no PIR ($\mathsf{R}^*_{\mathsf{noPIR}}$, black, solid line) achieving privacy requires a higher backhaul rate. The required backhaul rate increases with the number of colluding SBSs T.

For $M \geq 100$ and no PIR, the backhaul rate is zero, as all files can be downloaded from the SBSs. Indeed, for $M = 100$, we can select $k_i = 2\ \forall i$ and cache one coded symbol from each stripe of each file in each SBS (thus satisfying the constraint $\sum_{i=1}^{F} \mu_i \leq M$ as $\sum_{i=1}^{200} \mu_i = \sum_{i=1}^{200} 1/k_i = \sum_{i=1}^{200} 0.5 = 100$). Since for no PIR to retrieve each stripe of a file it is enough to download 2 symbols from each

[6] The curve $\mathsf{R}^*_{\mathsf{noPIR}}$ in the figure is identical to that in [7, Fig. 4]. As proved in Lemma 2, while the proposed content placement is different from the one in [7], they are equivalent in terms of average backhaul rate.

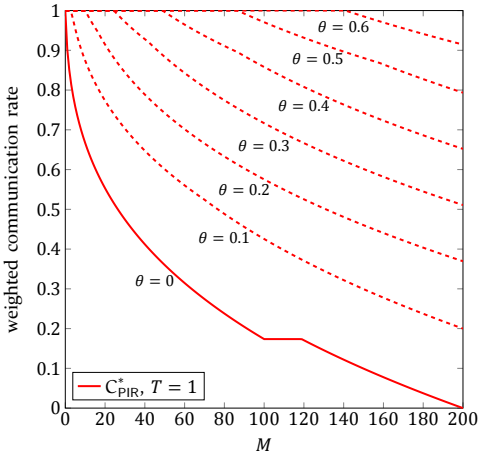

Figure V.4: Optimized weighted communication rate as a function of the cache size constraint M for a system with $T = 1$ spy SBS, $F = 200$ files, $N_{\mathsf{SBS}} = 316$, $\alpha = 0.7$, and several values of θ.

stripe of the file (due to the MDS property) and according to $\boldsymbol{\gamma}$ at least 2 SBSs are within range, for $M = 100$ (and hence for $M > 100$ as well) the user can always retrieve the file from the SBSs and the backhaul rate is zero. For the case of PIR and $T = 1$, on the other hand, the required backhaul rate is positive unless all complete files can be cached in all SBSs, i.e., $M = F$. For $T = 2$ and $T = 3$, even for $M = F$ the backhaul rate is not zero. This is because in this case the user needs to receive $n = 3$ and $n = 4$ responses $\boldsymbol{r}^{(l)}$, $l = 1, \ldots, n$, respectively (from the SBSs or the MBS). However, for the considered system parameters the probability that the user has $b \geq 3$ SBSs within range is not one, thus the user always needs to download data from the MBS to recover the file and the backhaul rate is positive.

For comparison purposes, in the figure we also plot the backhaul rate for the case of popular content placement $\mathsf{R}_{\mathsf{PIR}}^{\mathsf{pop}}$ in (V.20) (blue, dashed lines). In this case, the optimal value of n is 2, 3, and 4 for $T = 1$, $T = 2$, and $T = 3$, respectively. We remark that the curve $\mathsf{R}_{\mathsf{PIR}}^{\mathsf{pop}}$ for $T = 1$ overlaps with the curve $\mathsf{R}_{\mathsf{noPIR}}^{\mathsf{pop}}$. This is due to the fact that for $T = 1$, $n = 2$, and $\gamma_0 = \gamma_1 = 0$, $\mathsf{R}_{\mathsf{PIR}}^{\mathsf{pop}}$ in (V.20) boils down to $\sum_{i=M+1}^{F} p_i$, which is $\mathsf{R}_{\mathsf{noPIR}}^{\mathsf{pop}}$ in (V.12). However, for the general case, i.e., other $\boldsymbol{\gamma}$, $\mathsf{R}_{\mathsf{PIR}}^{\mathsf{pop}}$ and $\mathsf{R}_{\mathsf{noPIR}}^{\mathsf{pop}}$ may differ. As already shown in [7], for no PIR the optimized content placement yields significantly lower backhaul rate than popular content placement. For the PIR case and $T = 1$, up to $M = 118$ the optimized content placement also yields some performance gains with respect to popular content placement, albeit not as significant as for the case of no PIR. Interestingly, as shown in the figure, for $M \geq 119$, PIR popular content placement is optimal. Furthermore, as shown in the figure, for $T = 2$ and $T = 3$ popular content placement is optimal for all M.

In Fig. V.4, we plot the optimized weighted communication rate $\mathsf{C}_{\mathsf{PIR}}^{*}$ in (V.24) for the noncolluding case ($T = 1$) as a function of the cache size constraint M and several values of θ. For the considered system parameters, caching is still useful

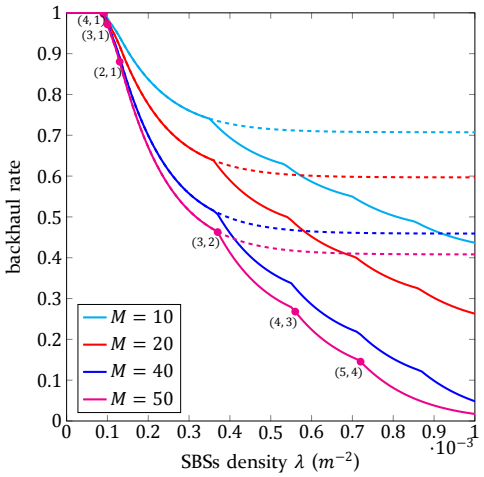

Figure V.5: Backhaul rate as a function of the density of SBSs λ and several values M for the scenario where SBSs are distributed according to a PPP and $T = 1$. $F = 200$ files and $\alpha = 0.7$. Solid lines correspond to optimal content placement ($\mathsf{R}^{*}_{\mathsf{PIR}}$ in (V.19)) and dashed lines to popular content placement ($\mathsf{R}^{\mathsf{pop}}_{\mathsf{PIR}}$ in (V.20)).

for small values of θ if the cache size is big enough. For example, for $\theta = 0.5$ caching helps in reducing the weighted communication rate with respect to no caching for $M \geq 87$. For $\theta \geq 0.7$, caching does not bring any reduction of the weighted communication rate.

In Figs. V.5 and V.6, we plot the backhaul rate for a PPP deployment model where SBSs are distributed over the plane according to a PPP and a user at an arbitrary location in the plane can connect to all SBSs that are within radius r_{u}. Let λ be the density of SBSs per square meter. For this scenario, the probability that a user is in communication range of b SBSs is given by [27]

$$\gamma_b = \mathrm{e}^{-\psi}\frac{\psi^b}{b!},$$

where $\psi = \lambda\pi r_{\mathsf{u}}^2$. In Fig. V.5, we plot the optimized backhaul rate ($\mathsf{R}^{*}_{\mathsf{PIR}}$ in (V.19), solid lines) as a function of the density λ for $F = 200$ files, $\alpha = 0.7$, $r_{\mathsf{u}} = 60$ meters, different cache size constraint M, and a single spy SBS, i.e., $T = 1$. For small densities, caching does not help in reducing the backhaul rate. However, as expected, the required backhaul rate diminishes by increasing the density of SBSs. For comparison purposes, we also plot the backhaul rate for popular content placement ($\mathsf{R}^{\mathsf{pop}}_{\mathsf{PIR}}$ in (V.20), dashed lines). Interestingly, popular content placement is optimal up to a given density of SBSs, after which optimizing the content placement brings a significant reduction of the required backhaul rate. Similar results are observed for $T = 2$ and $T = 4$ colluding SBSs in Fig. V.6 with the same system parameters as in Fig. V.5. In Figs. V.5 and V.6, for each M the optimal value of n and μ depends on the density of SBSs. Typically, a pair (n, μ) is optimal for a range of densities. In the figures, we give the optimal values of n and k for $M = 50$ (in particular we give the pair (n, k), with $k = 1/\nu$, which is also the code parameters of

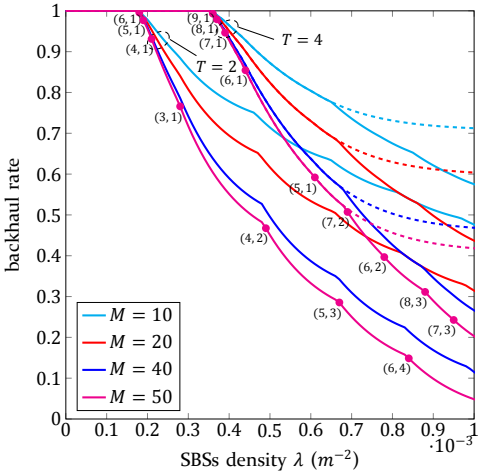

Figure V.6: Backhaul rate as a function of the density of SBSs λ and several values of M for the scenario where SBSs are distributed according to a PPP and $T = 2$ and $T = 4$. $F = 200$ files and $\alpha = 0.7$. Solid lines correspond to optimal content placement ($\mathrm{R}^{*}_{\mathrm{PIR}}$ in (V.19)) and dashed lines to popular content placement ($\mathrm{R}^{\mathrm{pop}}_{\mathrm{PIR}}$ in (V.20)).

the punctured code $\mathcal{C}'$). For convenience, in the figures we only give the parameters for the densities where the optimal pair (n, k) changes. The values should be read as follows: In Fig. V.5, walking the curve from top-left to bottom-right, no caching is optimal for densities up to $\lambda = 8 \cdot 10^{-5}$. For $\lambda = 9 \cdot 10^{-5}$, $(4, 1)$ is optimal. Then, $(3, 1)$ is optimal for densities $\lambda = 10^{-4}$ to $\lambda = 1.2 \cdot 10^{-4}$. From $\lambda = 1.3 \cdot 10^{-4}$ to $\lambda = 3.2 \cdot 10^{-4}$ the optimal value is $(2, 1)$, and so on (the curves are plotted with steps of 10^{-5}).

8 Conclusion

We proposed a private information retrieval scheme that allows to download files of different popularities from a cellular network, where to reduce the backhaul usage content is cached at the wireless edge in SBSs, while achieving privacy against a number of spy SBSs. We derived the backhaul rate for this scheme and formulated the content placement optimization. We showed that, as for the no PIR case, up to a number of spy SBSs caching helps in reducing the backhaul rate. Interestingly, contrary to the no PIR case, uniform content placement is optimal. Furthermore, popular content placement is optimal for some scenarios. Although uniform content placement is optimal, the proposed PIR scheme for multiple code rates may be useful in other scenarios, e.g., for distributed storage where data is stored using codes of different rates.

A Proof of Theorem 1

To prove that the protocol achieves PIR against T colluding SBSs, we need to prove that both the privacy condition in (V.2a) and the recovery condition in (V.2b) are satisfied. We first prove that the recovery condition in (V.2b) is satisfied.

According to Lemma 1, GRS codes with a fixed weighting vector $\boldsymbol{v}$ and evaluation vector $\boldsymbol{\kappa}$ are naturally nested. Furthermore, puncturing a GRS code results in another GRS code, since GRS codes are weighted evaluation codes [25, Ch. 5]. Thus, $\mathcal{C}'_i \subseteq \mathcal{C}'_{\max}$ for all i, and it follows from (V.7) that

$$\tilde{\mathcal{C}} = \left(\sum_{i=1}^{F} \mathcal{C}'_i \right) \circ \bar{\mathcal{C}} = \mathcal{C}'_{\max} \circ \bar{\mathcal{C}}.$$

Furthermore, it can easily be shown that the Hadamard product of two GRS codes with the same evaluation vector $(\kappa_1, \ldots, \kappa_n)$ is also a GRS code with dimension equal to the sum of the dimensions minus 1. Thus, $\tilde{\mathcal{C}}$ is a GRS code of dimension $k_{\max} + T - 1$. As $\tilde{\mathcal{C}}$ is an $(n, k_{\max} + T - 1)$ MDS code (GRS codes are MDS codes), it can correct arbitrary erasure patterns of up to $\Gamma = n - (k_{\max} + T - 1)$ erasures. This implies that one can construct a valid $k_{\max} \times n$ ($d = k_{\max}$) matrix $\hat{\boldsymbol{E}}$ (satisfying conditions C1–C3) from $\beta = \Gamma$ information sets $\{\mathcal{I}_m\}$ of $\mathcal{C}'_{\max}$ as shown below.

Let $\mathcal{J}_j = \{j, \ldots, (j + \Gamma - 1) \bmod n\}$, $j = 1, \ldots, k_{\max}$. Construct $\hat{\boldsymbol{E}}$ in such a way that $\mathcal{J}_j$ is the support of the j-th row of $\hat{\boldsymbol{E}}$. Hence, C1 is satisfied. Furthermore, since $\tilde{\mathcal{C}}$ is an $(n, k_{\max} + T - 1)$ MDS code and $\Gamma = n - (k_{\max} + T - 1)$, all rows of $\hat{\boldsymbol{E}}$ are correctable by $\tilde{\mathcal{C}}$, and thus C2 is satisfied. Finally, run Algorithm 1, which constructs $\beta = \Gamma$ information sets $\{\mathcal{I}_m\}$ of $\mathcal{C}'_{\max}$ (and the corresponding sets $\{\mathcal{F}_l\}$) such that C3 is satisfied. Note that since $\mathcal{C}'_{\max}$ is an MDS code, all coordinate sets of size $k_{\max}$ are information sets of $\mathcal{C}'_{\max}$, and hence Algorithm 1 will always succeed in constructing a valid set of information sets of $\mathcal{C}'_{\max}$ (the inequalities in Lines 6 and 7 together with the fact that the overall weight of $\hat{\boldsymbol{E}}$ is $\Gamma k_{\max}$ ensure that $\beta = \Gamma$ valid information sets for $\mathcal{C}'_{\max}$ are constructed). In particular, the while-loop in Line 6 will always terminate.

From the constructed matrix $\hat{\boldsymbol{E}}$, the user is able to recover $\Gamma d \geq \beta k_i$ unique code symbols of the requested file $\boldsymbol{X}^{(i)}$, at least k_i symbols from each stripe. Furthermore, a set of k_i recovered code symbols from each stripe corresponds to an information set of $\mathcal{C}'_i$ (any subset of size k_i of any information set of size $k_{\max}$ of $\mathcal{C}'_{\max}$ is an information set of $\mathcal{C}'_i$), and the requested file $\boldsymbol{X}^{(i)}$ can be recovered. This can be seen following a similar argument as in the proof of [21, Th. 6], and it follows that the recovery condition in (V.2b) is satisfied.

Secondly, we consider the privacy condition in (V.2a). A reasoning similar to the proof of [21, Lem. 6] shows that it is satisfied, and we refer the interested reader to this proof for further details. The fundamental reason is that addition of a deterministic vector in (V.5) does not change the joint probability distribution of $\{\boldsymbol{Q}^{(l)}, l \in \mathcal{T}\}$ for any set $\mathcal{T}$ size T, and the proof follows the same lines as the proof of [20, Th. 8]. However, note that there is a subtle difference in the sense that independent instances of the protocol may query different sets of SBSs. However, since the set of SBSs that are queried is independent of the requested file and

Algorithm 1: Construction of $\{\mathcal{I}_m\}$ for Theorem 1

Input: $\hat{\boldsymbol{E}}$, β, n, $k_{\max}$
Output: $\{\mathcal{I}_m\}$, $\{\mathcal{F}_l\}$
for $m \in \{1, \ldots, \beta\}$ **do**
  $\mathcal{I}_m \leftarrow \emptyset$
end
for $l \in \{1, \ldots, n\}$ **do**
  $\mathcal{F}_l \leftarrow \emptyset$, $m \leftarrow 1$
  while $|\mathcal{F}_l| \leq w_{\mathsf{H}}(\boldsymbol{t}_l)$ **do**
    if $|\mathcal{I}_m] < k_{\max}$ **then**
      $\mathcal{F}_l \leftarrow \mathcal{F}_l \cup \{m\}$
      $\mathcal{I}_m \leftarrow \mathcal{I}_m \cup \{l\}$
    end
    $m \leftarrow m + 1$
  end
end

depends only on which SBSs that are within communication range, this fact does not leak any additional information on which file is requested by the user.

References

[1] U. Niesen, D. Shah, and G. W. Wornell, "Caching in wireless networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6524–6540, Oct. 2012.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[3] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[4] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[5] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[6] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[7] V. Bioglio, F. Gabry, and I. Land, "Optimizing MDS codes for caching at the edge," in *Proc. Global Commun. Conf. (GLOBECOM)*, San Diego, CA, Dec. 2015.

[8] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[9] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, "Base-station assisted device-to-device communications for high-throughput wireless video networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3665–3676, Jul. 2014.

[10] J. Pedersen, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Distributed storage in mobile wireless networks with device-to-device communication," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4862–4878, Nov. 2016.

[11] A. Piemontese and A. Graell i Amat, "MDS-coded distributed storage for low delay wireless content delivery," in *Proc. 2016 9th Int. Symp. Turbo Codes & Iterative Inform. Process. (ISTC)*, Brest, France, 2016, pp. 320–324.

[12] J. Pedersen, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Optimizing MDS coded caching in wireless networks with device-to-device communication," Jan. 2017, arXiv:1701.06289v2 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1701.06289

[13] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 36th Annual ACM Symp. Theory Comput. (STOC)*, Chicago, IL, Jun. 2004, pp. 262–271.

[14] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, Jun./Jul. 2014, pp. 856–860.

[15] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, China, Jun. 2015, pp. 2842–2846.

[16] R. Tajeddine and S. El Rouayheb, "Private information retrieval from MDS coded data in distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1411–1415.

[17] S. Kumar, E. Rosnes, and A. Graell i Amat, "Private information retrieval in distributed storage systems using an arbitrary linear code," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 1421–1425.

[18] H. Sun and S. A. Jafar, "Private information retrieval from MDS coded data with colluding servers: Settling a conjecture by Freij-Hollanti et al." in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 1893–1897.

[19] ——, "The capacity of private information retrieval," *IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4075–4088, Jul. 2017.

[20] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, and D. A. Karpuk, "Private information retrieval from coded databases with colluding servers," *SIAM J. Appl. Algebra Geom.*, vol. 1, no. 1, pp. 647–664, Nov. 2017.

[21] S. Kumar, H.-Y. Lin, E. Rosnes, and A. Graell i Amat, "Achieving maximum distance separable private information retrieval capacity with linear codes," 2017, arXiv:1712.03898v3 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1712.03898

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

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

[24] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. 36th IEEE Symp. Found. Comp. Sci. (FOCS)*, Milwaukee, WI, Oct. 1995, pp. 41–50.

[25] W. C. Huffman and V. Pless, Eds., *Fundamentals of Error-Correcting Codes*. Cambridge, UK: Cambridge University Press, 2010.

[26] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE Joint Conf. Comput. Commun. Soc. (INFOCOM)*, New York, NY, Mar. 1999, pp. 126–134.

[27] B. Serbetci and J. Goseling, "On optimal geographical caching in heterogeneous cellular networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, Mar. 2017.

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS

uib.no

ISBN: 978-82-308-3651-4