Private Information Retrieval Using Regenerating Codes

Chatdanai Dorkson

Royal Holloway, University of London

A thesis presented for the degree of Doctor of Philosophy

2020

Declaration of Authorship

I, Chatdanai Dorkson, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed: ชัชดนัย ดอกสน (Chatdanai Dorkson)

Date: 15 September 2020

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr Siaw-Lynn Ng, for her consistent support throughout my PhD research studies. All the guidance, encouragement, and insightful feedback are invaluable.

Special thanks also to Professor Simon Blackburn and Professor Mark Wildon for helpful comments during the annual reviews every year.

I gratefully acknowledge the Development and Promotion of Science and Technology Talents Project (Royal Thai Government Scholarship) for providing the financial funding over the past 10 years.

I am thankful to all my friends in the UK for all their supports, both physically and mentally. Time is precious and I am grateful that I have spent it with the right people. My life in London can never be completed without all of you.

Finally, I would like to thank my parents who are always there for me.

Abstract

Private Information Retrieval (PIR) schemes allow a user to download a record from a database without disclosing the identity of the desired record. Downloading the entire database is clearly a solution, but it could be highly inefficient when the database is large. In the classical case, PIR is performed on a database replicated among non-communicating servers, resulting in a high storage cost. This motivates the use of erasure codes, where only a fraction of the database is stored in each server, and the design of code-based PIR. However, many erasure codes suffer from needing a large download, sometimes the entire database, to repair a failed server. In this thesis we consider code-based PIR that also allows efficient repair, and construct several PIR schemes using regenerating codes, a class of codes providing efficient repair.

Furthermore, we explore a multi-message scenario when a user wants to retrieve multiple records. Obviously, the user can repeatedly use a single-message PIR scheme, but we wish for a more efficient way. We propose a general multi-message PIR (MPIR) model for a particular class of regenerating codes and derive relationships between costs of storage, retrieval and repair. We extend our result on single-message PIR to build MPIR schemes that lie on the repair-retrieval trade-off curve. Additionally, we include an application of an averaging technique which was first introduced by Blackburn, Etzion and Paterson to improve retrieval rate for existing PIR schemes in replicated databases. We answer an open question about application of the technique to coded databases by applying the technique to our MPIR schemes. We also give an improvement factor in general when applying the technique to existing code-based PIR schemes in a similar way, and lastly propose a PIR scheme achieving the best improvement factor.

Contents

List of Abbreviations

PIR	Private Information Retrieval
MPIR	Multi-Message Private Information Retrieval
DSSs	Distributed Storage Systems
MDS	Maximum Distance Separable
MBR	Minimum Bandwidth Regenerating
MSR	Minimum Storage Regenerating
PM-MBR	Product-Matrix Minimum Bandwidth Regenerating
PM-MSR	Product-Matrix Minimum Storage Regenerating
SO	Storage Overhead
cPoP	communication Price of Privacy
RR	Repair Ratio

Chapter 1

Introduction

Preserving the identity of data retrieved from public online databases has captured a growing research interest during the past decades. A *private information retrieval* (PIR) scheme, introduced by Chor et al. [1], allows a user to download records from a database without revealing the identity of desired records to the database server. There are many real-world applications: notable examples include investors who attempt to keep the identity of interested stocks secret to preserve the integrity of the market price, or researchers who are in search of existing patents and do not want to disclose their research topic. PIR can be divided into two main classes in regard to the privacy guarantees they provide. The first class is *computational* PIR where privacy is established against computationally bounded servers; therefore, computing the identity of retrieved records is beyond the computational limits of the servers (see more work on computational PIR in [2–9]). The second class is *information-theoretic* PIR where privacy is established against computationally unbounded servers. In this thesis, we only consider information-theoretic PIR, and will refer to it simply as PIR.

1.1 Motivation

A trivial solution for PIR is that the user can download the entire database and examine the information of their interest. This is highly inefficient. However, in the case of the database stored on a single server, it is proved in [1] that this is essentially the best possible solution. Accordingly, we consider PIR in the database stored on multiple servers. We call infrastructure that can store a large amount of information on a network of servers *distributed storage systems* (DSSs).

PIR schemes in DSSs using replication, where the whole database is stored on each server, are called *replication-based* PIR schemes. In the original setting for replication-based PIR [1], there are n non-colluding servers, where each server stores the whole database, and the length of each record in the database is just one bit. The following example will illustrate a basic idea of how PIR schemes work. Suppose that we have three records $X^1, X^2, X^3 \in \mathbb{F}_2$ replicated across two servers, and a user wants to download the record X^1. The user generates a 3-bit vector (u_1, u_2, u_3) uniformly at random. Servers 1 and 2 are given the queries (u_1, u_2, u_3) and $(u_1 + 1, u_2, u_3)$, and then requested to return

$$u_1X^1 \oplus u_2X^2 \oplus u_3X^3,$$

and

$$(u_1 + 1)X^1 \oplus u_2X^2 \oplus u_3X^3,$$

respectively. Hence the user can reconstruct the record X^1 by computing the XOR of responses from both servers. Instead of downloading all three bits in order to obtain one bit of information, the user only downloads two bits. The minimisation of download cost is the focus of much of the literature and also of this work. More work on replication-based PIR can be found in the survey [10] and references therein.

The downside of replication-based PIR schemes is the high storage cost (for instance, in the previous example, the savings in downloading are at the expense of doubling the storage). This motivates the use of erasure codes in DSSs, where only a fraction of the whole database is stored in each server, resulting in lower storage cost. PIR schemes in DSSs using erasure codes are called *code-based* PIR schemes. The literature review on code-based PIR will be found in Chapter 4.

Additionally, PIR has been extended to many variations. For example, PIR with colluding servers where the servers can share information [11–14], and symmetric PIR (SPIR) where on top of the privacy of the desired record, the user also learns nothing about undesired records [15, 16]. One interesting scenario is when a user wants to retrieve multiple records without having to repeatedly use a PIR scheme that can retrieve one record (we refer this as a *single-message* scheme) multiple times. PIR schemes that can retrieve multiple records at a time are called *multi-message* PIR (MPIR) schemes.

An important problem in DSSs using erasure codes is what happens during a server failure. This is called a *repair problem*. For example, in the Facebook warehouse, approximately 1% of servers are unavailable per day, and 10-20% of the total average of 2 PB/day network traffic is for the repair. The database in the Facebook warehouse is encoded by [14,10] Reed-Solomon codes [17]. When an $[n, k]$ MDS code (of which Reed-Solomon codes are an example) is used in DSSs, a conventional method to repair a failed server requires a connection to some set of k servers to download all data for the reconstruction of the entire database, so we can extract the data that was stored in the failed server. Downloading these amounts of data to only repair a single server is extravagant, paving the way for a new concept of regenerating codes introduced by Dimakis et al. [18], which is a class of codes that provides reliability of data and efficient repair of failed servers in DSSs. (See more literature on regenerating codes in this survey [19].)

In recent literature, the repair problem in DSSs using erasure codes and PIR in DSSs using erasure codes are considered separately. This motivates our work in this thesis by addressing these problems in an integral manner. We aim to investigate the repair problem on code-based PIR. We introduce a new metric called *repair ratio* to measure the efficiency of repair. We construct single-message PIR schemes using regenerating codes in order to minimise the repair ratio when a server failure occurs in the system. Then, we explore a multi-message PIR problem. We propose a general model for MPIR in DSSs using regenerating codes, and analyse a trade-off between download cost and repair ratio under this model. Afterwards, we construct MPIR schemes using regenerating codes that attain the trade-off between download cost and repair ratio in this model. Furthermore, we discuss an averaging technique, which is first introduced by Blackburn et al. [20] for replication-based PIR. This technique is used to transform a PIR scheme into a new scheme with an improved download cost. Our contribution is the first application of the averaging technique on code-based PIR.

1.2 Thesis Structure

We summarise the contents of the thesis in this section. Our original contributions appear in Chapters 3, 5, 6, 7 with the main contributions on PIR constructions and analyses in Chapters 5, 6 and 7. This thesis is structured as follows.

An overview of codes for DSSs is given in Chapter 2. We recall basic knowledge in coding theory including linear codes and MDS codes, and discuss how to use these codes in DSSs. We explain the concept of regenerating codes, and introduce minimum bandwidth regenerating (MBR) codes and minimum storage regenerating (MSR) codes, which are classes of regenerating codes that are optimal in terms of the storage and repair bandwidth trade-off. We also present the explicit constructions

of product-matrix regenerating codes from [21], which are practical regenerating codes as they use a compact matrix presentation of codes with efficient encoding and decoding algorithms.

In Chapter 3, we formally explain the setting for code-based PIR schemes. We contribute a general model that explains the encoding of the database in code-based PIR, so we can compare the efficiency of schemes in related works under the same encoding model. We present efficiency metrics to measure storage cost and retrieval cost of PIR schemes. We also define a new metric to measure the efficiency of repair.

We subsequently discuss related literature in code-based PIR with non-colluding servers in Chapter 4.

In Chapter 5, we propose various single-message PIR schemes using the product-matrix minimum bandwidth regenerating (PM-MBR) codes and product-matrix minimum storage regenerating (PM-MSR) codes from [21] in the DSSs. The use of regenerating codes reduces the repair cost; hence our schemes obtain more efficient repair compared to schemes using MDS codes.

Chapter 6 is concerned with the multi-message scenario. We present a general MPIR model where the product-matrix regenerating codes are used for storage. We discover a trade-off between download cost and repair ratio under the proposed model. After that, we modify our single-message PIR schemes from Chapter 5 to multi-message PIR schemes that lie on the trade-off curve between download cost and repair cost. Note that our work is the first to explore multi-message PIR with regenerating codes.

In Chapter 7, we answer the open question (Q8) of [20] by introducing the application of an averaging technique from [20] on coded databases. We detail how to apply the technique to our MPIR scheme using PM-MBR codes to improve retrieval rates. We then provide an improvement factor in general when the technique is applied to other existing PIR schemes in code-based PIR in a

similar way. We construct a PIR scheme using $[2k, k]$ MDS codes that achieves the highest improvement factor after applying the averaging technique.

Lastly, we summarise our contributions and propose possible future research directions in Chapter 8.

Chapter 2

An Overview of Codes for Distributed Storage

It is important to be able to communicate reliably through noisy communication channels. Coding theory was first studied to deal with this problem. The key idea is to add some redundancy to the encoded message to enable the recovery of the sent message in the presence of errors. An interesting application of coding theory is in data storage. Distributed storage systems (DSSs) are infrastructure that can store a large amount of information on a network of servers in which redundancy is introduced to ensure reliability and availability of data when server failures inevitably occur. Data can be stored in DSSs with replication, however erasure codes can be used in DSSs to improve storage overhead. In this chapter, we briefly present an overview of codes for distributed storage, including important definitions and properties of relevant classes of codes that will be used in this thesis.

2.1 Fundamentals

In this section, we review fundamentals on coding theory. Most of the material of this section can be found in greater detail in [22].

Definition 2.1. An *alphabet* $\mathcal{A}$ is a finite set of symbols. A *code* C *of length* n is a nonempty subset of $\mathcal{A}^n$. A column vector in C is called a *codeword*. A *dimension* of a code C, denoted by $\dim(C)$, is $\log_{|\mathcal{A}|} |C|$, and a *code rate* R_c is $\frac{\dim(C)}{n}$.

Definition 2.2. An *array* code over an alphabet $\mathcal{A}$ is defined as a set of nonempty subset of $\mathbb{M}_{n\times\alpha}(\mathcal{A})$, where $\mathbb{M}_{n\times\alpha}(\mathcal{A})$ is the set of $n \times \alpha$ matrices over $\mathcal{A}$. Hence, a codeword of an array code is an $n \times \alpha$ matrix over $\mathcal{A}$.

Two common models of communication channels that have been used in coding theory are the q-ary symmetric channel and q-ary erasure channel.

Definition 2.3. The *q-ary symmetric channel* (q-SC) is a communication channel where the set of inputs and outputs in the channel are an alphabet of size q with the probability of error p. That is, when each transmitted symbol is considered independently, the probability that the transmitted symbol is correct is $1 - p$, and the probability that the transmitted symbol is one of the other $q - 1$ symbols is $\frac{p}{q-1}$.

Definition 2.4. The *q-ary erasure channel* (q-EC) is a communication channel where a transmitted symbol has probability p of not being received ("erased") and probability $1-p$ of being received correctly. The set of inputs is an alphabet of size q and the set of outputs is the set of inputs with the extra symbol e for the erasure symbol. Each symbol is erased independently with probability p. The receiver is aware when a symbol is erased, and hence the positions of the errors are known.

A code can be considered as a set of possible transmitted messages that can circumvent errors or erasures in noisy channels. We usually refer to a code that

can detect and correct errors in a q-SC as an *error-correcting* code, and a code that can manage with erasure symbols in a q-EC as an *erasure* code. In DSSs, it is possible to have errors in the information stored on the servers, so here we regard server failures as erasures. Erasure codes could be used in DSSs by considering each position in a codeword as a content of a server. This ensures the reliability of data while decreasing the storage overhead of replicated servers.

Next, we introduce the Hamming distance, which is the measurement of distance between codewords.

Definition 2.5. The *Hamming distance* $d_H(\cdot, \cdot)$ between two vectors of length n is the number of positions where they are different.

We use the Hamming distance to measure the minimum distance of a code, which is the smallest distance between different codewords. The minimum distance can be used to determine the capability of the code of detecting or correcting the errors during data transmission.

Definition 2.6. The *minimum distance* d of a code C is defined as

$$d = \min_{x,y \in C, x \neq y} d_H(x, y).$$

Theorem 2.7. A code C can detect t errors if and only if $d \geq t + 1$, and C can correct t errors if and only if $d \geq 2t + 1$.

Note that if there are s erasure symbols in a transmitted message, then the set of codewords restricted to unerased coordinates has the minimum distance at least $d - s$, so by Theorem 2.7, it can correct up to $\lfloor \frac{d-s-1}{2} \rfloor$ errors. Hence, we have the following corollaries.

Corollary 2.8. A code C can correct any combination of t errors and s erasure symbols if $d \geq 2t + s + 1$.

Corollary 2.9. A code C can correct s erasure symbols if $d \geq s + 1$.

The main work in coding theory is to construct "good" codes that have high rate and large minimum distance, so they correct many errors while using as little redundancy as possible. There are various bounds on parameters of codes, and we give an example of bounds, namely the Singleton bound, in next section.

2.1.1 Linear Codes

In this thesis, we only consider linear codes which is an important class of codes that is effective for practical applications. The alphabet in linear codes is a finite field $\mathbb{F}_q$ of size q where q is a prime power. Let $\mathbb{F}_q^n$ be the vector space of column vectors of length n over $\mathbb{F}_q$.

Definition 2.10. A code C is a *linear code* if any linear combination of two codewords is also a codeword, that is, if $x, y \in C \subset \mathbb{F}_q^n$, then $\alpha x + \beta y \in C$ for any $\alpha, \beta \in \mathbb{F}_q$.

By definition, a linear code is a subspace of $\mathbb{F}_q^n$, thus there always exists a basis for a linear code. Note that a basis of a linear code is not unique, but the size of a basis is, and that is the dimension of the code C. We refer to a linear code C over a finite field $\mathbb{F}_q$ of length n with dimension k and minimum distance d as an $[n, k, d]_q$ code. We sometimes omit parameters d or q.

Theorem 2.11. For a linear code C, $\dim(C) = k$ if and only if $|C| = q^k$.

There are two ways to represent a linear code. The first is by a generator matrix.

Definition 2.12. Let C be an $[n, k]_q$ linear code. A *generator matrix* G is an $n \times k$ matrix such that its k columns form a basis for the code C.

By this definition, each codeword in C can be written as a linear combination of columns from a generator matrix G, so a linear code C can be described as

$$C = \{Gx \mid x \in \mathbb{F}_q^k\}.$$

Notice that a code C provides an encoding function via a generator matrix G which transforms a message x of length k to a codeword $Gx \in C$ of length n. The *redundancy* of the code C is defined as $n - k$. A code is said to be *systematic* if an encoded message contains the original data in an uncoded form. This also indicates how a linear code is used in a DSS as a large amount of information can be divided to k packets to be encoded via its generator matrix into n packets stored in n different servers.

Since the columns of a generator matrix G form a basis for C, we can use elementary column operations to transform G to

$$G' = \begin{bmatrix} I_k \\ A \end{bmatrix},$$

where I_k is the $k \times k$ identity matrix, and A is an $(n-k) \times k$ matrix. This is called a generator matrix in the *standard form*. Note that a code C is systematic when a generator matrix in the standard form is used in the encoding since the original message is the first k symbols in the encoded message.

Definition 2.13. Let $C \subset \mathbb{F}_q^n$ be a linear code. Its *dual code* $C^\perp$ is the set of all vectors which are orthogonal to every vector in C; that is,

$$C^\perp = \{v \in \mathbb{F}_q^n \mid vc^T = 0, \forall c \in C\}.$$

Next we define a parity-check matrix, the second way to describe a linear code.

Definition 2.14. A *parity-check matrix* H of an $[n,k]_q$ linear code C is an $n\times(n-k)$ generator matrix for the dual code $C^{\perp}$.

Therefore, we can define a linear code C via a parity check matrix H as

$$C = \left\{ c \in \mathbb{F}_q^n \mid c^T \cdot H = \mathbf{0}_{1\times(n-k)} \right\},$$

and it is then obvious that $G^T \cdot H = 0$. Furthermore, it can be shown that if

$$G = \begin{bmatrix} I_k \\ A \end{bmatrix}$$

is a generator matrix in the standard form of an $[n,k]_q$ linear code C, then

$$H = \begin{bmatrix} -A^T \\ I_{n-k} \end{bmatrix},$$

is a parity-check matrix of C (in the *standard form*). The parity-check matrix in the standard form also provides another way to encode a message $x = (x_1, \ldots, x_k)^T$ of length k to a codeword $c = (c_1, \ldots, c_n)^T$ of length n as

$$\mathbf{0} = c^T \cdot H = (c_1, \ldots, c_n) \cdot \begin{bmatrix} -A^T \\ I_{n-k} \end{bmatrix}$$

implying that the redundancy bits are $(c_{k+1}, \ldots, c_n) = (c_1, \ldots, c_k) \cdot A^T = x^T A^T$.

Theorem 2.15. For a linear code C with a parity-check matrix H, the minimum distance of C is d if and only if every $d-1$ rows of H are linearly independent and there exist d linearly dependent rows in H.

From this theorem, since $\text{rank}(H) = n - k$, which means that the maximum number of linearly independent rows is $n - k$, we have a bound for linear codes.

Theorem 2.16. (Singleton bound for linear codes) An $[n, k, d]_q$ linear code satisfies

$$n - k \geq d - 1.$$

Remark that there is also a general version of the Singleton bound.

Theorem 2.17. (Singleton bound) For a code C of length n with minimum distance d over an alphabet of size q, we have

$$|C| \leq q^{n-d+1}.$$

2.1.2 MDS Codes

Definition 2.18. *Maximum distance separable* (MDS) codes are codes that attain the Singleton bound.

In this thesis, we only deal with *linear* MDS codes, which are linear codes that attain the Singleton bound in Theorem 2.16. We simply refer to linear MDS codes as MDS codes. MDS codes have the greatest capability of error correcting since the minimum distance is $d = n - k + 1$, reaching the upper bound. By Theorem 2.7 and Corollary 2.9, MDS codes can correct up to $\lfloor \frac{n-k}{2} \rfloor$ errors, and up to $n - k$ erasures. Therefore, DSSs using MDS codes can tolerate up to $n-k$ server failures.

It is shown in [22] that $[n, 1, n]$, $[n, n-1, 2]$ and $[n, n, 1]$ MDS codes exist over any $\mathbb{F}_q$, and we call these *trivial* MDS codes.

Theorem 2.19. The only binary MDS codes are the trivial MDS codes.

Theorem 2.20. The dual code of an MDS code is also an MDS code.

Combined Theorem 2.20 with Theorem 2.15, we have a characterisation of the MDS codes.

Corollary 2.21. Let C be an $[n, k, d]_q$ linear code. The following are equivalent:

(i) C is an MDS code,

(ii) Any $n - k$ rows of a parity-check matrix H are linearly independent,

(iii) Any k rows of a generator matrix G are linearly independent,

(iv) $C^{\perp}$ is an MDS code.

This corollary shows that when an MDS code is used in DSSs, the data stored in any k out of n servers suffices to recover the original information. This gives reassurance that the system can tolerate up to $n - k$ server failures as previously mentioned. However, in order to maintain the level of reliability when a server fails, we need to regenerate lost data at a new server. In order to do this in DSSs using MDS codes, we would need to connect to any k remaining servers to reconstruct the whole data and extract data that was stored in the failed server. This means that we need to download the information from k servers (essentially the entire database) in order to repair one server, and this is expensive.

In [18], a new class of array codes called regenerating codes is pioneered by Dimakis et al. in order to deal with the repair problem in DSSs, so single server failures can be repaired more cheaply. Dimakis et al. [18] use an information flow graph to derive a trade-off between storage and repair bandwidth (the amount of information to be downloaded for repair). Two interesting extremal points on the trade-off curve are called the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points, and codes with parameters meeting these points are called MSR and MBR codes. Since the repair problem in coded-PIR is our focus in this thesis, we are interested in exploring PIR in MBR and MSR codes. In the next section, we formally define regenerating codes, and discuss MBR and MSR codes.

2.2 Regenerating Codes

An $(n, k, r, \alpha, \beta, B)$ *regenerating code* is an array code that can store a database of size B over a finite field $\mathbb{F}_q$ among n servers where each server stores α symbols satisfying two properties:

(i) (recovery) the database can be recovered from the data from any k servers,

(ii) (repair) if a server fails, then a replacement server connects to some r helper servers, denoted by $h_1, \ldots, h_r$, where $k \leq r < n$, and downloads β symbols from each server in order to regenerate the α lost symbols.

Figure 2.1 illustrates the recovery and repair properties. The total amount of downloaded data in the repair process is $\gamma = r\beta$ symbols. This is called the *repair bandwidth*, and typically the repair bandwidth is smaller than the size of the whole database.

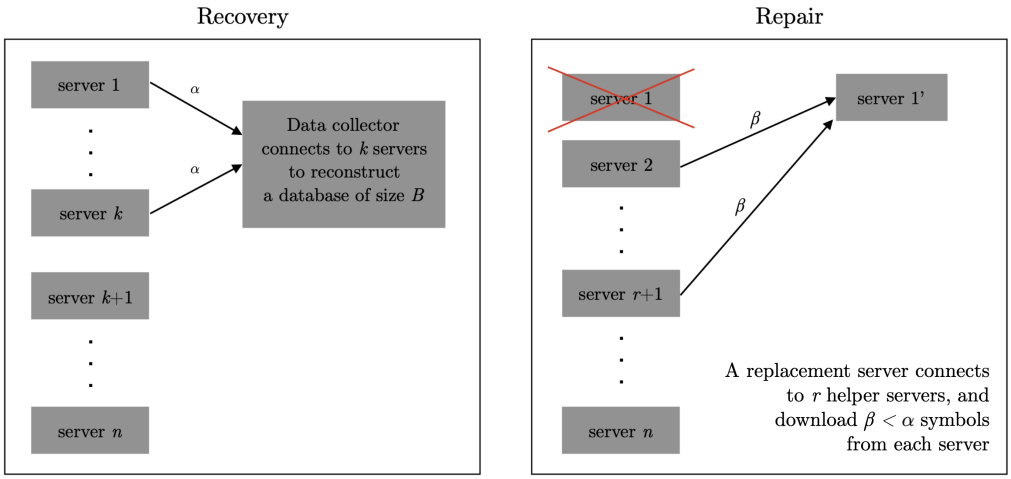

Figure 2.1: The recovery and repair properties of regenerating codes.

2.2.1 MBR and MSR Codes

In [23], the parameters of regenerating codes are shown to necessarily satisfy

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

and the achievable trade-off between storage and repair bandwidth is characterised by fixing the repair bandwidth, and then deriving the minimum α which satisfies Inequality (2.1). Two interesting extremal points on the optimal trade-off curve are the *minimum bandwidth regeneration* (MBR) point which minimises repair bandwidth first and then minimises storage overhead, and the *minimum storage regeneration* (MSR) point which minimises in the reverse order. It can be shown that the MBR point is achieved by

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2Br}{k(2r-k+1)}, \frac{2Br}{k(2r-k+1)}\right), \tag{2.2}$$

and *MBR codes* are $(n, k, r, \alpha, \beta, B)$ regenerating codes that satisfy Equation (2.2). Notice that repair bandwidth for MBR codes is equal to the amount of information stored in the failed server. For the MSR point, it is achieved by

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{B}{k}, \frac{Br}{k(r-k+1)}\right), \tag{2.3}$$

and *MSR codes* are $(n, k, r, \alpha, \beta, B)$ regenerating codes that satisfy Equation (2.3). Since a database of size B can be recovered from any k servers and each server stores $\frac{B}{k}$ symbols for the MSR codes, the amount of information downloaded for the recovery is exactly the size of database; i.e., it is optimal in terms of redundancy and reliability. Hence, MSR codes are equivalent to standard MDS codes, while MBR codes are not.

Note that for MBR and MSR codes the parameters α and B are multiples of β, and it has been proved that if there exists an MBR or MSR code with $\beta = 1$, then we can use it to construct an MBR or MSR codes with any higher value of β.

There have been many constructions of MBR and MSR codes (for example [21, 24–27]). In this thesis we are interested in the product-matrix constructions of MBR and MSR codes with $\beta = 1$ by Rashmi et al. [21] since they use a compact matrix presentation of regenerating codes with efficient reconstruction and repair algorithms. Next, we present the details of these product-matrix constructions.

2.2.2 Product-Matrix MBR Codes [21]

In this section, we explain an explicit construction of the product-matrix MBR (PM-MBR) codes by Rashmi et al. [21]. This construction is designed to have $\beta = 1$, so by Equation 2.2, $r = r\beta = \gamma = \alpha$. Consequently, we can calculate $B = \frac{\alpha k(2r-k+1)}{2r} = \frac{k(2r-k+1)}{2}$. Hence, the parameters of this construction are

$$(n, k, r, \alpha, \beta, B) = \left(n, k, r, r, 1, \frac{k(2r-k+1)}{2}\right)$$

over a finite field $\mathbb{F}_q$ where $q \geq n$. Under the product-matrix framework, each codeword is represented by an $(n \times \alpha)$ matrix C which is the product

$$C = \Psi \cdot \mathcal{M}$$

of an $(n \times r)$ *encoding matrix* Ψ and an $(r \times \alpha)$ *message matrix* $\mathcal{M}$. In the matrix C, row i consists of the α encoded symbols stored by server i for each $i \in [n]$. The encoding matrix Ψ is given by an $(n \times r)$ matrix

$$\Psi = \begin{bmatrix} \Phi & \Delta \end{bmatrix}$$

where Φ is an $(n \times k)$ matrix and Δ is an $(n \times (r-k))$ matrix such that

(i) any r rows of Ψ are linearly independent,

(ii) any k rows of Φ are linearly independent.

The property (i) is designed for repair and (ii) is for recovery. We will give choices of suitable encoding matrices at the end of this chapter. The row i of Ψ is denoted by Ψ_i for $i \in [n]$. The $(r \times r)$ message matrix $\mathcal{M}$ contains B message symbols over $\mathbb{F}_q$. Here $\mathcal{M}$ is defined to be a symmetric matrix as

$$\mathcal{M} = \begin{bmatrix} S_1 & S_2 \\ S_2^T & 0 \end{bmatrix}$$

where S_1 is a $(k \times k)$ matrix constructed such that $\binom{k+1}{2}$ entries in the upper-triangular part of each matrix are filled up by $\binom{k+1}{2}$ distinct message symbols and entries in the strictly lower-triangular are chosen to make the matrix symmetric, and the $(k \times (r-k))$ matrix S_2 are filled up by the remaining $k(r-k)$ message symbols. This is the PM-MBR codes we will use in our PIR constructions.

2.2.3 Product-Matrix MSR Codes [21]

Rashmi et al. [21] also give an explicit construction of the product-matrix MSR (PM-MSR) codes when $r = 2k-2$. Also, this construction is designed to have $\beta = 1$. By Equation 2.3, $r = \gamma = \frac{Br}{k(r-k+1)}$, so $B = k(r-k+1) = k(2k-2-k+1) = k(k-1)$, and $\alpha = \frac{B}{k} = k-1$. Hence, the parameters of this construction are

$$(n, k, r, \alpha, \beta, B) = (n, k, 2k-2, k-1, 1, k(k-1))$$

over a finite field $\mathbb{F}_q$ where $q \geq n\alpha$. First, the encoding matrix Ψ is given by an $(n \times r)$ matrix

$$\Psi = \begin{bmatrix} \Phi & \Lambda\Phi \end{bmatrix}$$

where Φ is an $(n \times \alpha)$ matrix and Λ is an $(n \times n)$ diagonal matrix such that

(i) any r rows of Ψ are linearly independent,

(ii) any $k-1$ rows of Φ are linearly independent,

(iii) the n diagonal elements of Λ are all distinct.

The property (i) is designed for repair and (ii),(iii) are for recovery. The row i of Ψ is denoted by Ψ_i for $i \in [n]$. Next, the $(r \times \alpha)$ message matrix $\mathcal{M}$ is defined as

$$\mathcal{M} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

where S_1 and S_2 are $(\alpha \times \alpha)$ symmetric matrices constructed such that $\binom{k}{2}$ entries in the upper-triangular part of each matrix are filled up by $\binom{k}{2}$ distinct message symbols and entries in the strictly lower-triangular are chosen to make the matrices symmetric. This is the PM-MSR code we will use in our PIR constructions.

2.2.4 Choices of Encoding Matrices for PM-MBR and PM-MSR codes

In the construction of product-matrix regenerating codes from Rashmi et al. [21], the choices of encoding matrices for PM-MBR codes can be any $(n \times r)$ matrices with full rank. However, for PM-MSR codes, full-rank encoding matrices need to be carefully chosen to satisfy the condition of the diagonal matrix Λ such that the n diagonal elements of Λ are all distinct. In this section, we present some possible choices of the encoding matrices.

Definition 2.22. [28] A *Vandermonde* matrix is an $m \times n$ matrix V such that

$$V_{i,j} = \alpha_i^{j-1}$$

where $\alpha_i \in \mathbb{F}_q$, i.e.,

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_m & \alpha_m^2 & \cdots & \alpha_m^{n-1} \end{bmatrix}.$$

Remark 2.23. Any $(n \times r)$ Vandermonde matrix such that all α_i are distinct has full rank. Hence, it could be used as an encoding matrix for PM-MBR codes. For PM-MSR codes, consider an $(n \times r)$ Vandermonde matrix

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{2\alpha-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{2\alpha-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{2\alpha-1} \end{bmatrix},$$

where α_i are all distinct, we can write V in the form $\begin{bmatrix} V' & \Lambda V' \end{bmatrix}$ where

$$V' = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{\alpha-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{\alpha-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{\alpha-1} \end{bmatrix},$$

and

$$\Lambda = \begin{bmatrix} \alpha_1^\alpha & & & \\ & \alpha_2^\alpha & & \\ & & \ddots & \\ & & & \alpha_n^\alpha \end{bmatrix}.$$

Hence, the Vandermonde matrix V can be used as an encoding matrix for PM-MSR codes if the diagonal elements $\alpha_1^\alpha, \ldots, \alpha_n^\alpha$ in the matrix Λ are all distinct. One way to achieve this is to choose $\alpha_i = g^{i-1}$ where g is the generator of the multiplicative group of a finite field $\mathbb{F}_q$ of size at least $n\alpha$.

Definition 2.24. [28] A *Cauchy* matrix is an $m \times n$ matrix V such that

$$V_{i,j} = \frac{1}{\alpha_i - \beta_j}$$

where $\alpha_i, \beta_j \in \mathbb{F}_q$ such that α_i are all distinct, β_j are all distinct, and $\alpha_i - \beta_j \neq 0$ for all $i \in [m], j \in [n]$; i.e.,

$$V = \begin{bmatrix} \frac{1}{\alpha_1-\beta_1} & \frac{1}{\alpha_1-\beta_2} & \cdots & \frac{1}{\alpha_1-\beta_n} \\ \frac{1}{\alpha_2-\beta_1} & \frac{1}{\alpha_2-\beta_2} & \cdots & \frac{1}{\alpha_2-\beta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\alpha_m-\beta_1} & \frac{1}{\alpha_m-\beta_2} & \cdots & \frac{1}{\alpha_m-\beta_n} \end{bmatrix}.$$

Remark 2.25. To construct a systematic PM-MBR code, an encoding matrix could be chosen as

$$\Psi = \begin{bmatrix} I_{k\times k} & \mathbf{0}_{k\times(r-k)} \\ \bar{\Phi}_{(n-k)\times k} & \bar{\Delta}_{(n-k)\times(r-k)} \end{bmatrix},$$

where $\begin{bmatrix} \bar{\Phi} & \bar{\Delta} \end{bmatrix}$ is a Cauchy matrix.

Chapter 3

Code-based PIR Settings

In this chapter, we describe the settings of code-based PIR, which consists of an encoding step and a retrieval step. In Section 3.1, we present a general encoding model, and contribute an encoding function that explains the encoding of the database in DSSs using erasure codes. We make explicit the two types of encoding models that are used in the literature: mixed coding architecture and separate coding architecture. In Section 3.2, we define the privacy of retrieval using entropy. Lastly, we discuss efficiency metrics to measure the cost of storage and retrieval, and define a new metric to measure the cost of repair in Section 3.3.

3.1 Encoding Step

Suppose that a database X consists of m records, each of length ℓ, denoted by $X^1, X^2, \ldots, X^m \in \mathbb{F}_q^\ell$. Let $\mathcal{X}$ be the sample space of records of length ℓ. All m records are encoded and distributed across n non-colluding servers, where each server stores α symbols via an encoding function

$$E_{m,n,\alpha} : \mathcal{X}^m \to \mathbb{M}_{n\times\alpha}(\mathbb{F}_q).$$

Therefore, $E_{m,n,\alpha}(X^1, \ldots, X^m)$ is an $n \times \alpha$ matrix over $\mathbb{F}_q$ where row i, denoted by C_i, consists of α encoded symbols of $X^1, \ldots, X^m$ stored in server i for $i \in [n]$.

We also consider a special case of encoding where each record is encoded by an erasure code independently and separately, so it can be written as

$$E_{m,n,\alpha}(X^1, \ldots, X^m) = \left[E_{1,n,\bar{\alpha}}(X^1) \quad E_{1,n,\bar{\alpha}}(X^2) \quad \cdots \quad E_{1,n,\bar{\alpha}}(X^m) \right],$$

where $E_{1,n,\bar{\alpha}} : \mathcal{X} \to \mathbb{M}_{n \times \bar{\alpha}}(\mathbb{F}_q)$ is an encoding function that encodes each record separately to distribute across n servers, and each record stores $\bar{\alpha}$ symbols where $m\bar{\alpha} = \alpha$. We refer to this encoding as *separate coding architecture*. Here each server stores an encoded part of each record; that is, server i stores row i of $E_{1,n,\bar{\alpha}}(X^j)$ for all $i \in [n], j \in [m]$. Otherwise, when more than one record (usually all records) are jointly encoded by an erasure code, we refer to this as *mixed coding architecture*. Remark that replicated databases follow the separate coding architecture where

$$E_{1,n,\ell}(X^j) = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{n \times 1} \cdot X^j_{1 \times \ell} = \begin{bmatrix} \text{—} & X^j & \text{—} \\ & \vdots & \\ \text{—} & X^j & \text{—} \end{bmatrix}_{n \times \ell}.$$

The encoding model in this section includes existing approaches in the literature, so we can compare PIR schemes in the literature under the same terminology. In the literature, most existing works in code-based PIR (for example [11–14, 16, 29–35]) implicitly assume the separate coding architecture, while others ([36, 37]) assume mixed coding architecture. It is important to be aware of these implicit assumptions so that we may compare PIR schemes correctly (or something with more emphasis on results that only apply under the same assumptions). This is the reason we divide the encoding into two types: separate coding architecture and mixed coding architecture to clearly explain what assumption is made in the encoding.

3.2 Retrieval Step

Entropy is used to measure the average amount of information in a random variable. Here it is used to define privacy of the retrieval step. We will give the standard definitions first.

Definition 3.1. Let X, Y be random variables. Let $p(x_i)$ be the probability of the event $X = x_i$, and $p(y_i|x_i)$ be the conditional probability of the event $Y = y_i$ given that $X = x_i$. The *entropy* of X is defined as

$$H(X) = -\sum_i p(x_i) \log p(x_i),$$

the entropy of Y *conditioned* on $X = x_i$ is defined as

$$H(Y|X = x_i) = -\sum_i p(y_i|x_i) \log p(y_i|x_i),$$

and the *conditional* entropy of Y given X is the average of $H(Y|X = x_i)$ over all possible value x_i, which is

$$H(Y|X) = \sum_i p(x_i) H(Y|X = x_i).$$

We have that $H(Y|X) = 0$ implies that the value of the random variable Y is completely determined by the value of random variable X, while $H(Y|X) = Y$ implies that the random variables Y and X are independent.

In the retrieval step we assume that a user wants to download a record X^f. The user submits a query matrix Q^i over $\mathbb{F}_q$ to server i, where $i \in [n]$. Then the server i computes and responds with an answer A^i.

Definition 3.2. A PIR scheme is said to be *information-theoretically perfect* if

(i) (privacy) $H(f|Q^i) = H(f)$ for every $i \in [n]$;

(ii) (decodability) $H(X^f|A^1,\ldots,A^n)=0$.

We can see that (i) implies that server i does not obtain any information about which record is being downloaded by the user, and (ii) ensures that the user can recover the desired record X^f with no errors from all responses $A^i, i\in[n]$.

3.3 Efficiency Metrics

We introduce metrics to measure the efficiency of PIR schemes. First, we measure the storage cost by *storage overhead* (SO), which is defined to be the ratio of the total storage used in the scheme to the total size of the whole database.

For the retrieval cost, we follow the approach from [29] by assuming that the size of each record is arbitrarily large, so we can divide the database to be many chunks and each chunk is operated independently and identically by the same PIR scheme (see Figure 3.1). Since we can use the same set of queries to perform PIR on each chunk, we can neglect the upload cost and only focus on the download cost. We use *communication Price of Privacy* (cPoP) which is defined as the ratio of the total amount of downloaded data to the total size of the desired record [30] to measure the download cost for PIR schemes. The *retrieval rate* (R_{PIR}) is the multiplicative inverse of the cPoP. The *capacity* of PIR is the maximum achievable retrieval rate over all PIR schemes for some fixed parameters.

Lastly, as the repair problem is a primary concern in this thesis, we define *repair ratio* (RR) as the ratio of the total amount of symbols downloaded for repairing a failed server to the size of the failed server. In general we would like SO, cPoP and RR to be small. However, when we operate PIR, these metrics might compete with each other: for example, there is a trade-off between SO and cPoP derived in [29], and a trade-off between cPoP and RR in our work on MPIR in Chapter 6.

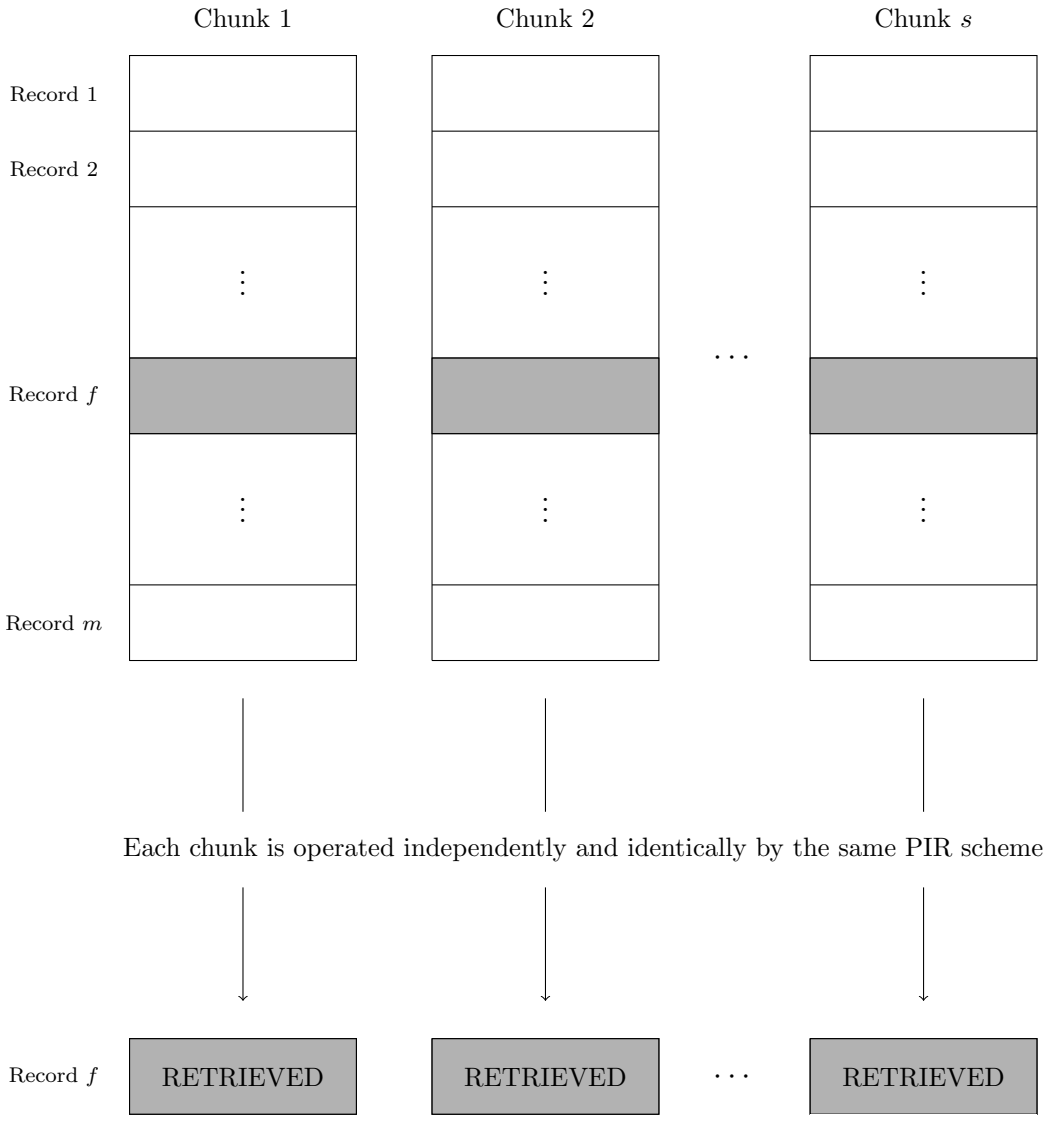

Figure 3.1: Dividing each record into s chunks where each chunk is operated independently and identically by the same PIR scheme.

Chapter 4

Literature Review

In this chapter, we review relevant literature on code-based PIR with non-colluding servers, which mean that a server does not know the content of other servers, and it also does not see queries sent to or replies from another server. Code-based PIR has emerged as an area of recent interest since it provides better storage overhead than replication-based PIR given the same level of redundancy. Indeed, in code-based PIR, if we use a $(k + c, k)$ linear code to build PIR schemes with redundancy c, then storage overhead is $1 + \frac{c}{k}$, while in replication-based PIR, storage overhead of PIR schemes with redundancy c is $1 + c$.

4.1 The First Work on Code-Based PIR

Shah et al. [36] pioneer the work of code-based PIR providing bounds on download cost (the total number of bits downloaded) at the expense of storage overhead, and they also show a compromise of the download cost for more reasonable storage cost.

In this paper, they prove that for $m \geq 3$ records, each of ℓ bits, any PIR scheme must download at least $\ell + 1$ bits in the worst case. For a PIR scheme that achieves this lower bound, in almost every PIR operation, $\ell + 1$ bits must be

downloaded individually from distinct $\ell + 1$ servers, one from each server, and the storage overhead must be super-linear in ℓ, that is, storage overhead approaches infinity as ℓ goes to infinity. An explicit PIR scheme that meets the lower bound on download for PIR, which is only one extra bit of download, is given which means that the cPoP is $1 + \frac{1}{\ell}$. The encoding in this construction follows the mixed coding architecture. However, this construction requires the number of servers n in a system to be exponential in the number of records m, which is $n = (\ell + 1)^{m-1}$ servers.

As a result of trying to reduce the storage cost from the lower-bound-achieving construction, Shah et al. provide another PIR scheme with the cPoP between 2 and 4 which is a small factor away from optimality, which is $1 + \frac{1}{\ell}$. This is the first and only PIR scheme that uses regenerating codes in the literature. This scheme also assumes the mixed coding architecture. Next, we give some details of this scheme since we will apply the technique from this scheme to construct a new PIR scheme using PM-MSR codes in Section 5.1.2.

Encoding Step

In this scheme, the database is encoded by using the PM-MBR code in Section 2.2.2, which is a regenerating code with parameters

$$(n, k, r, \alpha, \beta, B) = \left(n, k, r, r, 1, \frac{k(2r - k + 1)}{2}\right),$$

with $n \geq 2r$ (as in the retrieval process we need to perform PIR on $2r$ servers). For the encoding step, we suppose that each record has $\ell = \frac{2r-m+1}{2}$ symbols over a finite field $\mathbb{F}_q$. Hence, the whole database has $\frac{m(2r-m+1)}{2}$ symbols in total, which can be fitted in the PM-MBR code with $k = m$. Note that in this scheme the PM-MBR code is required to be systematic, which implies that the first m servers

contain the original $\frac{m(2r-m+1)}{2}$ symbols. Together with special arrangement of the message matrix, we can have that the data stored in server i contains record X^i in an uncoded form for the first m servers. By Remark 2.25, we can choose

$$\Psi = \begin{bmatrix} I_{m\times m} & 0 \\ \bar{\Phi} & \bar{\Delta} \end{bmatrix}_{n\times r}$$

where $\begin{bmatrix} \bar{\Phi} & \bar{\Delta} \end{bmatrix}$ is a Cauchy matrix to satisfy the systematic condition.

Next, we recall that the message matrix for PM-MBR codes in Section 2.2.2 is

$$\mathcal{M}(X^1, \ldots, X^m) = \begin{bmatrix} S_1 & S_2 \\ S_2^T & 0 \end{bmatrix}_{r\times r}$$

where S_1 is a symmetric $(m \times m)$ matrix where the upper-triangular half of the matrix are filled up by distinct message symbols, and S_2 is an $(m \times (r-m))$ matrix filled up by the remaining message symbols, and for notation simplicity, we will write $\mathcal{M}(X^1, \ldots, X^m)$ as $\mathcal{M}$. It requires some manipulation of how the message symbols should be arranged in the message matrix to have that server i contains record X^i. Note that these details of the arrangement are not included in the original paper. In this scheme, the message matrix $\mathcal{M}$ is rearranged as follows:

(i) Write $X^j = \{x_1^j, x_2^j, \ldots, x_\ell^j\}$.

(ii) Put the first $\frac{m+1}{2}$ symbols of record j into row j of the matrix S_1, starting from the (j, j) position and shifting circularly to the beginning of that row if necessary for every $j \in [m]$.

(iii) Fill up the remaining position in the matrix S_1 to make the matrix symmetric.

(iv) Put the remaining $r - m$ symbols of record j into row j of the matrix S_2.

For example, let $m = 5, r = 6$, so $\ell = \frac{2r-m+1}{2} = 4$. We have

$$S_1 = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & x_3^4 & x_2^5 \\ x_2^1 & x_1^2 & x_2^2 & x_3^2 & x_3^5 \\ x_3^1 & x_2^2 & x_1^3 & x_2^3 & x_3^3 \\ x_3^4 & x_3^2 & x_2^3 & x_1^4 & x_2^4 \\ x_2^5 & x_3^5 & x_3^3 & x_2^4 & x_1^5 \end{bmatrix}, \quad S_2 = \begin{bmatrix} x_4^1 \\ x_4^2 \\ x_4^3 \\ x_4^4 \\ x_4^5 \end{bmatrix}.$$

Therefore, the encoding function for this scheme is the product

$$E_{m,n,r}(X^1, \ldots, X^m) = \Psi \cdot \mathcal{M},$$

and for the first m servers, server i contains record X^i as required.

Retrieval Step

Let ψ_i denote the i^{th} row of Ψ. Then, we pick $u \in \mathbb{F}_q^r$ randomly. For the first r servers, we download

$$\{\psi_{h_1}\mathcal{M}u, \ldots, \psi_{h_r}\mathcal{M}u\},$$

and for the other r servers, we download

$$\{\psi_{h_{r+1}}\mathcal{M}(u + \psi_f), \ldots, \psi_{h_{2r}}\mathcal{M}(u + \psi_f)\}.$$

Since any r rows of Ψ are linearly independent, we obtain $\mathcal{M}u$ and $\mathcal{M}(u + \psi_f^T)$, respectively, and finally have $\mathcal{M}\psi_f^T = \psi_f\mathcal{M}$ which contains record X^f as desired. Notice that the idea of the retrieval is similar to the repair of a failed server in the PM-MBR codes in Section 2.2.2 with the additional random vector u in order to confuse the server.

Analysis

In this PIR scheme, SO is

$$\frac{n\alpha}{m\ell} = \frac{nr}{m(\frac{2r-m+1}{2})} = \frac{2r}{2r-m+1} \cdot \frac{n}{m},$$

and when $n = 2r = 2m$, SO is less than 4. The cPoP is

$$\Big(\sum_{i=1}^{n} |A_i|\Big)/\ell = \frac{2r}{\ell} = \frac{4r}{2r-m+1} \leq 4,$$

while RR is the smallest possible which is $\frac{r}{\alpha} = 1$ as the PM-MBR code is used in the storage system.

4.2 Subsequent Works under the Separate Coding Architecture

After the first work by Shah et al. [36], there are two general directions of research in the study of code-based PIR schemes. The first direction is to consider the relationship between SO and cPoP when we encode the database and retrieve record(s) in a specific way, and construct concrete schemes that have good storage overhead and cPoP. Another direction is to consider the problem of PIR capacity, defined as the maximum of retrieval rate over all possible PIR schemes for some fixed parameters, without assuming any conditions on the retrieval technique.

4.2.1 The First Direction

Here we will review three important papers of particular relevance to our research in this direction. The Chan et al. paper [29] proposes a general model for PIR schemes using linear storage codes, and studies fundamental limits on the costs of

storage and retrieval by deriving the trade-off between SO and cPoP in the context of their proposed PIR model. Subsequently, the Tajeddine et al. paper [30] gives an explicit PIR scheme which attains the trade-off between SO and cPoP using MDS codes in the storage. Afterwards, the Kumar et al. paper [31] presents a PIR scheme using an arbitrary systematic linear storage code of rate $> 1/2$.

4.2.1.1 The Work of Chan et al. [29]

In [29], Chan et al. set up a general encoding model for PIR schemes using linear codes assuming the separate coding architecture, which has been subsequently used by many works on code-based PIR. Suppose that the size of each record X^j is large, which is of length $\ell = \alpha k$ over $\mathbb{F}_q$. We can interpret that X^j has α stripes, and each stripe of length k is encoded independently via an $(n \times k)$ generator matrix G of an $[n, k]$ linear code. It can be formally written as

$$E_{m,n,m\alpha}(X^1, \ldots, X^m) = \begin{bmatrix} E_{1,n,\alpha}(X^1) & E_{1,n,\alpha}(X^2) & \cdots & E_{1,n,\alpha}(X^m) \end{bmatrix},$$

where

$$E_{1,n,\alpha}(X^j) = G_{n\times k} \cdot \begin{bmatrix} x^j_{1,1} & x^j_{1,2} & \cdots & x^j_{1,\alpha} \\ \vdots & \vdots & \ddots & \vdots \\ x^j_{k,1} & x^j_{k,2} & \cdots & x^j_{k,\alpha} \end{bmatrix}_{k\times\alpha},$$

for all $j \in [m]$ (see Figure 4.1). Therefore, each server stores $m\alpha$ symbols in total consisting of all the i^{th} position of encoded stripes of the record j for all $j \in [m]$.

They also provide a corresponding retrieval scheme where server i is given a $d \times m\alpha$ query matrix Q^i and then asked to return $A^i = Q^i C_i^T$. This query matrix Q^i is the sum of a uniformly random matrix and a deterministic matrix that is designed to access certain symbols stored to server i. Note that the parameter d corresponds to the number of subqueries and may be adjusted for different schemes.

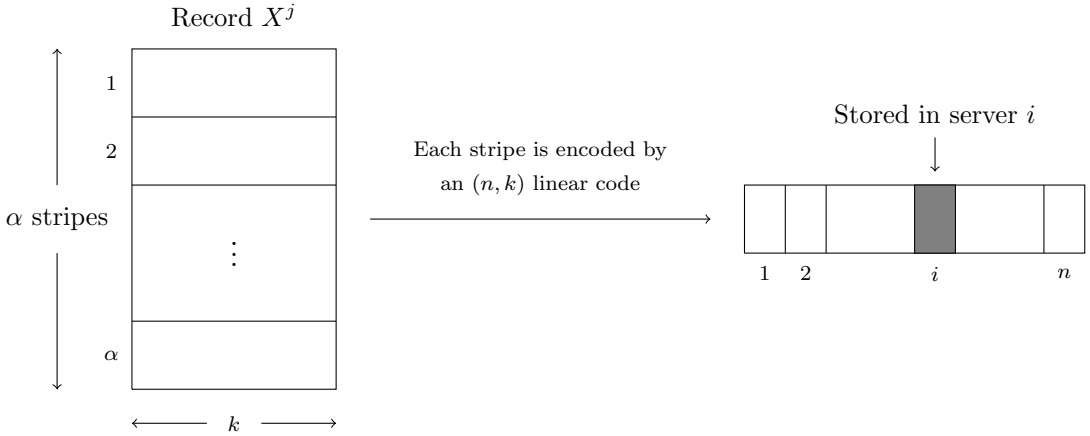

Figure 4.1: Coding process for record X^j for [29]

By studying the conditions under which PIR as well as database reconstruction may be achieved, the following trade-off between SO and cPoP is obtained (see Figure 4.2):

$$1 \leq \text{cPoP}\left(1 - \frac{1}{\text{SO}}\right). \tag{4.1}$$

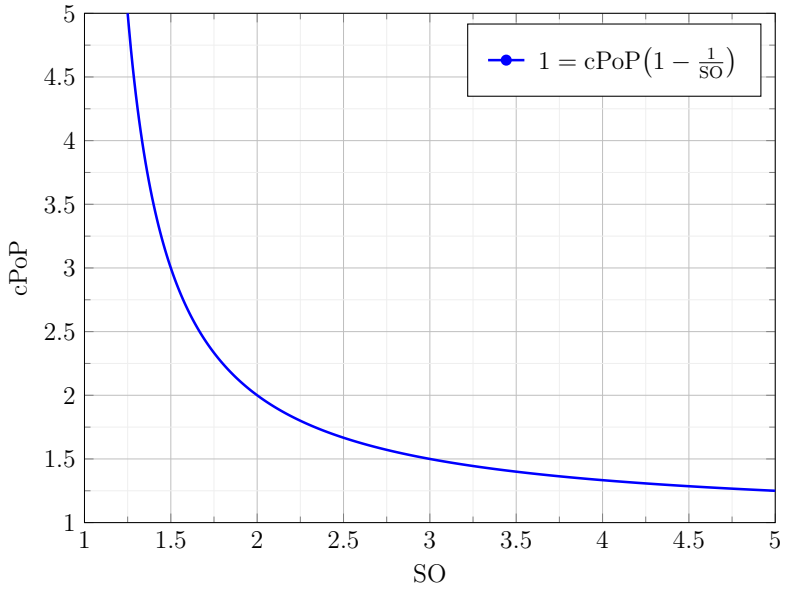

Figure 4.2: The trade-off curve between SO and cPoP derived by Chan et al. [29] for PIR in DSSs using erasure codes under the separate coding architecture.

4.2.1.2 The Work of Tajeddine et al. [30]

Tajeddine et al. present a PIR scheme that attains the trade-off between SO and cPoP (4.1). This is the first explicit PIR scheme that follows the approach proposed by Chan et al. [29] providing an interesting retrieval technique. We describe this scheme in detail since we modify and generalise this retrieval technique to our works in Chapters 5 and 6. The encoding part of the model in [29] is instantiated with an $[n, k]$ MDS code, with the additional requirement that $\alpha = n - k$. Write $\alpha = \lambda k + r$ where λ, r are integers such that $0 \leq r < k$ using the Euclidean division algorithm. The retrieval steps are as follows:

(i) (Initialisation) The user generates a random matrix U of dimensions $k \times m\alpha$ whose elements are chosen independently and uniformly at random over $\mathbb{F}_q$,

(ii) (Query Generation) The query matrix Q^i is defined as

$$Q^i = \begin{cases} U + V^i, & \text{if } i = 1, \ldots, n - r \\ U, & \text{if } i = n - r + 1, \ldots, n \end{cases}$$

where V^i is a deterministic matrix designed to access desired symbols. Recall that server i stores $m\alpha$ symbols. If the entry (j, b) of V^i is 1, then it implies that the b^{th} symbols that stored in server i is privately retrieved by the j^{th} subquery of Q^i. We choose

$$V^1 = \left[\begin{array}{c|c|c|c} \mathbf{0}_{k\times(f-1)\alpha} & \begin{matrix} I_{r\times r} \\ \mathbf{0}_{(k-r)\times r} \end{matrix} & \mathbf{0}_{k\times\lambda k} & \mathbf{0}_{k\times(m-f)\alpha} \end{array} \right],$$

and $V^i, i = 2, \ldots, k$ is defined by a single downward cyclic shift on its row vectors of the matrix V^{i-1}. For $i = sk + 1, \ldots, sk + k, s = 1, \ldots, \lambda$, we define

$$V^i = \left[\begin{array}{c|c|c} \mathbf{0}_{k\times(f-1)\alpha+r+(s-1)k} & I_{k\times k} & \mathbf{0}_{k\times(\lambda-s)k+(m-f)\alpha} \end{array} \right],$$

(iii) (Response Mappings) Each server i returns a vector $A^i = Q^i C_i^T$.

In this scheme, each record consists of $\alpha = \lambda k + r$ stripes. The queries submitted to servers $1, \ldots, k$ are designed to retrieve the first r stripes of the record X^f. The queries submitted to servers $sk+1, \ldots, sk+k$, where $s = 1, \ldots, \lambda$, are designed to retrieve the s^{th} set of k stripes of the record X^f.

For the analysis of this scheme, the storage overhead is $\frac{n}{k}$. Each query is carefully designed to complete the retrieval process, and the resulting cPoP is $\frac{1}{1-R_c}$ where $R_c = \frac{k}{n}$ is the code rate of the $[n, k]$ MDS code. Hence,

$$\text{cPoP}\Big(1 - \frac{1}{\text{SO}}\Big) = \frac{1}{1-R_c}(1 - R_c) = 1,$$

implying that this scheme attains the trade-off curve between SO and cPoP (4.1).

4.2.1.3 The Work of Kumar et al. [31]

Later, Kumar et al. [31] use an arbitrary systematic linear storage code of rate $R_c > 1/2$ in the encoding model by Chan et al. [29]. They provide an algorithm to search for the protocol with optimum cPoP. Interestingly, the numerical results show that the optimal trade-off in [29] can be attained by using locally repairable codes (LRCs) [38] or Pyramid codes [39], which have more efficient repair property, in the storage. Indeed, $[n, k]$ LRCs and Pyramid codes have the locality $c < k$ which means that each code block is a function of at most c other code blocks. Hence, when one server fails, we can connect to only c other surviving servers, and download α symbols from each server in order to reconstruct the failed server. This improves repair ratio from k to be $c < k$. Our research continues along this lines by considering PIR in regenerating codes which emphasises repair ratio.

4.2.2 The Second Direction

The second direction of research focuses on the PIR capacity which is the maximum retrieval rate independent of retrieval techniques. Sun and Jafar [40] first show that the capacity of the replication-based PIR is

$$C_{rep} = \frac{1 - \frac{1}{n}}{1 - \frac{1}{n^m}}. \tag{4.2}$$

The retrieval technique to obtain capacity-achieving is based on the principle that a user retrieves symbols, both desired and undesired, symmetrically across all servers, and uses the undesired symbols as side information to reveal the desired symbols.

In [32], Banawan and Ukulus investigate the capacity of the PIR scheme using MDS codes, which is shown to be

$$C_{MDS} = \frac{1 - \frac{k}{n}}{1 - (\frac{k}{n})^m} = \frac{1 - R_c}{1 - R_c^m} \tag{4.3}$$

where R_c is the rate of the $[n, k]$ code used in the PIR scheme. The encoding in this work is the same as in [30] following the separate coding architecture where each stripe of each record is encoded by an MDS code. We can see that the retrieval rate $1 - R_c$ achieved by [30] asymptotically reaches the MDS capacity (3.3) when m goes to infinity. Note that in the capacity-achieving scheme [32], the length of each record is set to be $\ell = kn^m$, which could be very large and affect the practicality of the scheme. In [34], Xu and Zhang show that the record size of schemes achieving the MDS capacity (3.3) must satisfy $\ell \geq k(n/\gcd(n, k))^{m-1}$ under the assumption that all answers from every server have the same length. They also provide a capacity-achieving scheme with $\ell = k(n/\gcd(n, k))^{m-1}$. Lately, it is shown by Zhou et al. [33] that the minimum length of each record could be as low as $\ell = \text{lcm}(n - k, k)$ for $m > k/\gcd(n, k)$ when answers from each server are of different lengths.

In the multi-message PIR (MPIR) problem, where a user wants to retrieve p records, Banawan and Ulukus [41] analyse the capacity of MPIR schemes with replicated database which is

$$C_{MPIR\text{-}L} = \frac{1}{1 + \frac{m-p}{pn}} \tag{4.4}$$

when $p \geq \frac{m}{2}$ or

$$C_{MPIR\text{-}S} = \frac{1 - \frac{1}{n}}{1 - (\frac{1}{n})^{m/p}} \tag{4.5}$$

when $p \leq \frac{m}{2}$, and $\frac{m}{p} \in \mathbb{N}$. In the example schemes that achieve capacity, there is a requirement of ℓ to be n^2 for the case $p \geq \frac{m}{2}$, but there is no explicit formula for ℓ in case $p \leq \frac{m}{2}$.

Later, Zhang and Ge [35] explore MPIR using $[n, k]$ MDS coded database and prove that when $p \geq \frac{m}{2}$, the exact capacity is

$$C_{MDS\text{-}MPIR} = \frac{1}{1 + \frac{k(m-p)}{pn}} \tag{4.6}$$

which agrees with the result on replicated databases in [41] when $k = 1$. As far as we know, this is the only paper on code-based MPIR capacity.

4.3 Other Related Works

After our early works [42, 43] where we construct PIR and MPIR schemes using PM-MSR and PM-MBR codes (the details of these constructions are in Sections 5.2.3 and 6.3), Lavauzelle et al. [44] propose PIR schemes that also use product-matrix regenerating codes in the single-message PIR scheme. This work improves our result by setting each record to have many stripes, and each stripe is encoded by an PM-MSR or PM-MBR code. They exploit the symmetry of message matrix in the construction from [21] as side information in the retrieval step, while ours

exploit the structure of the regenerating codes designed for efficient repair.

As most existing works on code-based PIR schemes have followed the separate coding architecture introduced by [29], which means that each stripe of each record is assumed to be separately encoded using the same erasure code. Sun and Tian [37] propose PIR schemes following the mixed coding architecture using an MDS code, and demonstrate that the retrieval rate in this scheme can break the capacity on MDS codes [32]. This leads to an open question whether the mixed coding architecture can be applied to other variations of code-based PIR to improve the retrieval rate.

Notice that in the communication cost, following the approach from [29], we only focus on the download cost because we assume that the size of each record is arbitrarily large, and we can divide each record to be many chunks and operate PIR independently with each chunk of all records using the same set of queries, so we can neglect the upload cost. Blackburn et al. [20] propose an averaging technique to applied with replicated databases by setting each record to be large, and instead of processing PIR with the same query set for each chunk, they vary randomness that is used for query generation through all possible random vectors for each chunk. Hence for each server there must be a query which is an all-zero vector, so the server does not need to reply in this case, resulting in better download cost. Note that queries for each chunk can be calculated from just one query, so the upload cost is still low. In Chapter 7, we give an application of the average technique on code-based PIR schemes.

Table 4.1 summarises related works in code-based PIR with non-colluding servers discussed in this chapter.

PIR Schemes	Encoding	Underlying Code	n	α	ℓ	SO	cPoP	RR
One Extra Bit of Download Ensures Perfectly PIR [36]	mixed	-	$(\ell+1)^{m-1}$	ℓ	fixed	$\frac{(\ell+1)^{m-1}}{m}$	$1+\frac{1}{\ell}$	-
	mixed	PM-MBR code with parameters $(n,m,r,r,1,B)$	$n \geq 2r$	r	$\frac{2r-m+1}{2}$	$\frac{2r}{2r-m+1} \cdot \frac{n}{m}$	$\frac{4r}{2r-m+1}$	1
PIR from MDS Coded Data [30]	separate	$[n,k]$ MDS code	fixed	$(n-k)m$	$(n-k)k$	$\frac{n}{k}$	$\frac{1}{1-\frac{n}{k}}$	k
PIR from Arbitrary Linear Code [31]	separate	(n,k) linear code with $R_c > \frac{1}{2}$	fixed	βm	βk	$\frac{n}{k}$	$\frac{n}{\beta}$	k
Capacity of PIR (Replication) [40]	separate	Repetition code	fixed	ℓm	n^m	n	$\frac{1-(1/n)^m}{1-(1/n)}$	1
Capacity of PIR (MDS Coded) [32]	separate	$[n,k]$ MDS code	fixed	mn^m	kn^m	$\frac{n}{k}$	$\frac{1-(k/n)^m}{1-(k/n)}$	k
Capacity of MPIR (Replication) with $p \geq \frac{m}{2}$ [41]	separate	Repetition code	fixed	ℓm	n^2	n	$1+\frac{m-p}{pn}$	1
Capacity of MPIR (Replication) with $p \leq \frac{m}{2}, \frac{m}{p} \in \mathbb{N}$ [41]	separate	Repetition code	fixed	ℓm	implicit	n	$\frac{1-(1/n)^{(m/p)}}{1-(1/n)}$	1
Capacity of MPIR (MDS Coded) with $p \geq \frac{m}{2}$ [35]	separate	$[n,k]$ MDS code	fixed	$m(s+t)\binom{n}{k}$	$k(s+t)\binom{n}{k}$	$\frac{n}{k}$	$1+\frac{k(m-p)}{pn}$	k
				where s,t are the smallest positive integers s.t. $s\binom{n}{k} = (s+t)(\binom{n}{k}-\binom{n-1}{k})$				

Table 4.1: The summary of related works in code-based PIR with non-colluding servers showing an underlying code in the encoding step with parameters n, α and ℓ and metrics SO, cPoP and RR.

Chapter 5

PIR using PM-MSR and PM-MBR codes

In this chapter we are interested to investigate how we can construct PIR schemes using regenerating codes in order to achieve efficient repair. In particular, we use the product-matrix regenerating codes from Rashmi et al. in Sections 2.2.2 and 2.2.3 in our constructions since they use a compact matrix presentation of codes that provides convenience in the encoding step of PIR. In Section 5.1, we apply the technique in the construction of the PIR scheme using PM-MBR codes under the mixed coding architecture from Shah et al. (Section 4.1) to construct a new PIR scheme using PM-MSR codes under the mixed coding architecture. In Section 5.2, we apply the technique in the construction of the PIR scheme using MDS codes from Tajeddine et al. (Section 4.2.1.2) to construct various PIR schemes under the separate coding architecture using both PM-MBR and PM-MSR codes. To the best of our knowledge, these are the first PIR schemes using regenerating codes under the separate coding architecture.

5.1 Schemes with Mixed Coding Architecture

In [36], Shah et al. propose the first PIR scheme using PM-MBR codes from [21] following the mixed coding architecture. In this section, we construct a new PIR scheme using PM-MSR codes from [21] applying their technique. In the encoding step, we use the systematic PM-MSR codes. It has been proved in [21] that every PM-MSR code can be made systematic via the message-symbol remapping. We first give more explicit details on how to do the process with a small example.

5.1.1 The Systematic Version of the PM-MSR Codes [21]

First, we recall the construction of the PM-MSR codes from [21] with parameters

$$(n, k, r, \alpha, \beta, B) = (n, k, 2k-2, k-1, 1, k(k-1)).$$

Let the encoding matrix Ψ be an $(n \times r)$ matrix given by

$$\Psi = \begin{bmatrix} \Phi & \Lambda\Phi \end{bmatrix}$$

where Φ is an $(n \times \alpha)$ matrix and Λ is an $(n \times n)$ diagonal matrix such that (i) any r rows of Ψ are linearly independent, (ii) any α rows of Φ are linearly independent, (iii) the n diagonal elements of Λ are all distinct. Write the ith row of Ψ as $\psi_i = \begin{bmatrix} \phi_i & \lambda_i\phi_i \end{bmatrix}$. Next, the $(r \times \alpha)$ message matrix $\mathcal{M}$ is defined as

$$\mathcal{M} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

where S_1 and S_2 are $(\alpha \times \alpha)$ symmetric matrices constructed such that the $\binom{k}{2}$ entries in the upper-triangular part of each of the two matrices are filled up by $\binom{k}{2}$

distinct message symbols and the entries in the strictly lower-triangular are chosen to make the matrices symmetric. For $i \in [n]$, the $k-1$ symbols stored in server i are

$$\psi_i \mathcal{M} = \begin{bmatrix} \phi_i & \lambda_i \phi_i \end{bmatrix} \mathcal{M} = \phi_i S_1 + \lambda_i \phi_i S_2.$$

Next, for the systematic version of the PM-MSR code, we want to ensure that the first k servers contain all original $B = k\alpha$ symbols. Let

$$\Psi_k = \begin{bmatrix} \Phi_k & \Lambda_k \Phi_k \end{bmatrix}$$

be the $(k \times r)$ submatrix of Ψ with the first k rows of Ψ, thus the $k\alpha$ symbols stored in the first k servers are given by $\Psi_k \mathcal{M}$. Let U be a $(k \times \alpha)$ matrix containing the original $B = k\alpha$ symbols. Hence, we need to solve for the entries of $\mathcal{M}$ in terms of the symbols in U in the equation

$$\Psi_k \mathcal{M} = U$$

in order to use $\mathcal{M}$ to obtain the systematic MSR code $C = \Psi \cdot \mathcal{M}$. Define the matrices P and Q to be

$$P = \Phi_k S_1 \Phi_k^T,$$

and

$$Q = \Phi_k S_2 \Phi_k^T,$$

which are both symmetric as S_1 and S_2 are. Notice that

$$\begin{aligned} P + \Lambda_k Q &= \Phi_k S_1 \Phi_k^T + \Lambda_k \Phi_k S_2 \Phi_k^T \\ &= \left[\Phi_k S_1 + \Lambda_k \Phi_k S_2\right] \Phi_k^T \\ &= \begin{bmatrix} \Phi_k & \Lambda_k \Phi_k \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \Phi_k^T \\ &= \Psi_k \mathcal{M} \Phi_k^T \\ &= U \Phi_k^T, \end{aligned}$$

which we have access to. The $(i,j)^{th}, 1 \leq i,j \leq k$, element of this matrix is

$$P_{ij} + \lambda_i Q_{ij}$$

while the $(j,i)^{th}$ element is

$$P_{ji} + \lambda_j Q_{ji} = P_{ij} + \lambda_j Q_{ij}.$$

As λ_i, λ_j are different, we can solve for P_{ij}, Q_{ij} in terms of symbols in U for every $i \neq j$. Consider first the matrix P. Say

$$\Phi_k = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_{\alpha+1} \end{bmatrix}.$$

It can be seen that the i^{th} row of P excluding the diagonal element is

$$\phi_i S_1 \begin{bmatrix} \phi_1^T & \cdots & \phi_{i-1}^T & \phi_{i+1}^T & \cdots & \phi_{\alpha+1}^T \end{bmatrix}.$$

By the construction of Ψ, the right matrix in the above equation is invertible, thus we can obtain $\phi_i S_1$ for every $i = 1, \ldots, \alpha + 1$. Hence, we have access to

$$\begin{bmatrix} \phi_1 \\ \vdots \\ \phi_\alpha \end{bmatrix} S_1$$

where the left matrix above is again invertible by the construction. We finally get the entries of S_1 in terms of the symbols in U. By repeating the same process with the matrix Q, we are able to solve for the entries of S_2 in terms of the symbols in U and obtain the desired matrix $\mathcal{M}$.

The following example will illustrate how to arrange the message matrix $\mathcal{M}$ to make the PM-MSR code systematic.

Example 5.1. Suppose we have a PM-MSR code with parameters

$$(n, k, r, \alpha, \beta, B) = (6, 3, 4, 2, 1, 6)$$

over $\mathbb{F}_{13}$, and we set

$$S_1 = \begin{bmatrix} s_1 & s_2 \\ s_2 & s_3 \end{bmatrix}, \quad S_2 = \begin{bmatrix} s_4 & s_5 \\ s_5 & s_6 \end{bmatrix}.$$

Assume that the encoding matrix is the Vandermonde matrix

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 1 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \end{bmatrix} \quad \text{where} \quad \Phi = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 1 & & & & & \\ & 4 & & & & \\ & & 9 & & & \\ & & & 3 & & \\ & & & & 12 & \\ & & & & & 10 \end{bmatrix}.$$

Thus we have,

$$\Psi_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 1 \end{bmatrix}.$$

This code can encode $B = 6$ symbols. We write those symbols in the matrix

$$U = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}.$$

Then

$$\begin{aligned} P + \Lambda_3 Q &= \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \\ &= \begin{bmatrix} x_{11} + x_{12} & x_{11} + 2x_{12} & x_{11} + 3x_{12} \\ x_{21} + x_{22} & x_{21} + 2x_{22} & x_{21} + 3x_{22} \\ x_{31} + x_{32} & x_{31} + 2x_{32} & x_{31} + 3x_{32} \end{bmatrix}. \end{aligned}$$

By considering the $(1,2)^{th}$ and $(2,1)^{th}$ elements, we have the following system of equations:

$$P_{12} + Q_{12} = x_{11} + 2x_{12},$$
$$P_{12} + 4Q_{12} = x_{21} + x_{22}.$$

We can solve that

$$P_{12} = 4x_{21} + 4x_{22} + 10x_{11} + 7x_{12},$$
$$Q_{12} = 9x_{21} + 9x_{22} + 4x_{11} + 8x_{12}.$$

By considering the $(1,3)^{th}$ and $(3,1)^{th}$ elements, we have the following system of equations:

$$P_{13} + Q_{13} = x_{11} + 3x_{12},$$
$$P_{13} + 9Q_{13} = x_{31} + x_{32}.$$

We can solve that

$$P_{13} = 8x_{31} + 8x_{32} + 6x_{11} + 5x_{12},$$
$$Q_{13} = 5x_{31} + 5x_{32} + 8x_{11} + 11x_{12}.$$

By considering the $(2,3)^{th}$ and $(3,2)^{th}$ elements, we have the following system of equations:

$$P_{23} + 4Q_{23} = x_{21} + 3x_{22},$$
$$P_{23} + 9Q_{23} = x_{31} + 2x_{32}.$$

We can solve that

$$P_{23} = 7x_{31} + x_{32} + 7x_{21} + 8x_{22},$$
$$Q_{23} = 8x_{31} + 3x_{32} + 5x_{21} + 2x_{22}.$$

Consider the first row of the matrix P excluding the diagonal element,

$$\begin{bmatrix} 4x_{21} + 4x_{22} + 10x_{11} + 7x_{12} & 8x_{31} + 8x_{32} + 6x_{11} + 5x_{12} \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} S_1 \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}.$$

As $\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 3 & 12 \\ 11 & 1 \end{bmatrix}$,

$$\begin{aligned} \begin{bmatrix} 1 & 1 \end{bmatrix} S_1 &= \begin{bmatrix} 3(4x_{21} + 4x_{22} + 10x_{11} + 7x_{12}) + 11(8x_{31} + 8x_{32} + 6x_{11} + 5x_{12}) \\ 12(4x_{21} + 4x_{22} + 10x_{11} + 7x_{12}) + (8x_{31} + 8x_{32} + 6x_{11} + 5x_{12}) \end{bmatrix}^T \\ &= \begin{bmatrix} 5x_{11} + 11x_{12} + 12x_{21} + 12x_{22} + 10x_{31} + 10x_{32} \\ 9x_{11} + 11x_{12} + 9x_{21} + 9x_{22} + 8x_{31} + 8x_{32} \end{bmatrix}^T. \end{aligned}$$

Consider the second row of the matrix P excluding the diagonal element,

$$\begin{bmatrix} 4x_{21} + 4x_{22} + 10x_{11} + 7x_{12} & 7x_{31} + x_{32} + 7x_{21} + 8x_{22} \end{bmatrix} = \begin{bmatrix} 1 & 2 \end{bmatrix} S_1 \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}.$$

As $\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 8 & 6 \\ 6 & 7 \end{bmatrix}$,

$$\begin{aligned} \begin{bmatrix} 1 & 2 \end{bmatrix} S_1 &= \begin{bmatrix} 8(4x_{21} + 4x_{22} + 10x_{11} + 7x_{12}) + 6(7x_{31} + x_{32} + 7x_{21} + 8x_{22}) \\ 6(4x_{21} + 4x_{22} + 10x_{11} + 7x_{12}) + 7(7x_{31} + x_{32} + 7x_{21} + 8x_{22}) \end{bmatrix}^T \\ &= \begin{bmatrix} 2x_{11} + 4x_{12} + 9x_{21} + 2x_{22} + 3x_{31} + 6x_{32} \\ 8x_{11} + 3x_{12} + 8x_{21} + 2x_{22} + 10x_{31} + 7x_{32} \end{bmatrix}^T. \end{aligned}$$

Therefore, we obtain $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} S_1 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} s_1 & s_2 \\ s_2 & s_3 \end{bmatrix}$, which is

$$\begin{bmatrix} 5x_{11} + 11x_{12} + 12x_{21} + 12x_{22} + 10x_{31} + 10x_{32} & 9x_{11} + 11x_{12} + 9x_{21} + 9x_{22} + 8x_{31} + 8x_{32} \\ 2x_{11} + 4x_{12} + 9x_{21} + 2x_{22} + 3x_{31} + 6x_{32} & 8x_{11} + 3x_{12} + 8x_{21} + 2x_{22} + 10x_{31} + 7x_{32} \end{bmatrix}.$$

Since $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & 12 \\ 12 & 1 \end{bmatrix}$, we can calculate that

$$\begin{aligned} s_1 &= 8x_{11} + 5x_{12} + 2x_{21} + 9x_{22} + 4x_{31} + x_{32}, \\ s_2 &= 10x_{11} + 6x_{12} + 10x_{21} + 3x_{22} + 6x_{31} + 9x_{32}, \\ s_3 &= 12x_{11} + 5x_{12} + 12x_{21} + 6x_{22} + 2x_{31} + 12x_{32}. \end{aligned}$$

And we can repeat the process with the matrix Q to get s_4, s_5, s_6.

5.1.2 Construction 1 (PM-MSR-mixed)

The technique in Shah's PM-MBR-mixed PIR scheme [36] is to exploit the repair process of the regenerating codes. Here we construct a new PIR scheme with PM-MSR codes, so we first discuss the repair process for PM-MSR codes [21] which have parameters

$$(n, k, r, \alpha, \beta, B) = (n, k, 2k-2, k-1, 1, k(k-1)).$$

When a server f fails, we contact $r = 2k-2$ helper servers, say $h_1, \ldots, h_{2k-2}$ and request for

$$\psi_{h_1}\mathcal{M}\phi_f^T, \ldots, \psi_{h_{2m-2}}\mathcal{M}\phi_f^T,$$

so we obtain

$$\begin{bmatrix} \text{——} & \psi_{h_1} & \text{——} \\ & \vdots & \\ \text{——} & \psi_{h_{2k-2}} & \text{——} \end{bmatrix} \mathcal{M}\phi_f^T.$$

The left matrix is invertible by the construction, so we get

$$\mathcal{M}\phi_f^T = \begin{bmatrix} S_1\phi_f^T \\ S_2\phi_f^T \end{bmatrix}.$$

As S_1, S_2 are symmetric, we can get $\phi_f S_1 + \lambda_f \phi_f S_2 = \psi_f \mathcal{M}$ which contains the lost α symbols in server f.

Since a PM-MSR code can store $k(k-1)$ symbols, we let the number of records $m = k$, each of length $\ell = \alpha = k-1$. We need a PM-MSR code to be systematic, so we can have that record i is stored in server i in uncoded form for all $i \in [k]$. Therefore, we can use the process of repair to retrieve the desired record. The technique for hiding the identity of the desired record is to mix the vector ϕ_f in the repair process with a random vector. Next we present the detail of the construction.

Encoding Step

Suppose we have a database consisting of m records $X^1, X^2, \ldots, X^m \in \mathbb{F}_q^\ell$, each of length $\ell = m - 1$. The database is stored using a PM-MSR code with parameters

$$(n, k, r, \alpha, \beta, B) = (n, m, 2m-2, m-1, 1, m(m-1)),$$

and the encoding matrix Ψ, where $n \geq 4m - 4$. To ensure that the code is systematic, we write all records in a matrix

$$U = \begin{bmatrix} — & X^1 & — \\ & \vdots & \\ — & X^m & — \end{bmatrix},$$

and use the process in Section 5.1.1 to obtain the message matrix $\mathcal{M}$ that makes the code systematic. With this arrangement of U, we also have that server i stores X^i in uncoded form for $i \in [m]$. Hence the encoding function in this scheme is

$$E_{m,n,m-1}(X^1, \ldots, X^m) = \Psi \cdot \mathcal{M}.$$

Note that if we have longer records, we can divide each record to be many chunks, each of length $m - 1$, and then operate them independently and identically.

Retrieval Step

Suppose that X^f where $f \in [m]$ is the desired record.

(i) Generate a column vector $u \in \mathbb{F}_q^{m-1}$ uniformly at random.

(ii) Connect to some $4m - 4$ arbitrary servers. Recall that ϕ_f is the f^{th} row of the matrix Φ. Then, pass the vector u to any $2m - 2$ of these servers, and pass the vector $u + \phi_f^T$ to any $2m - 2$ remaining servers.

(iii) For the first group of $2m-2$ servers, they are requested to return

$$\psi_{h_1}\mathcal{M}u, \ldots, \psi_{h_{2m-2}}\mathcal{M}u,$$

and for the other $2m-2$ servers, they are requested to return

$$\psi_{h_{2m-1}}\mathcal{M}(u+\phi_f^T), \ldots, \psi_{h_{4m-4}}\mathcal{M}(u+\phi_f^T).$$

To prove decodability, since any $r = 2m-2$ rows of Ψ are linearly independent, we obtain $\mathcal{M}u$ and $\mathcal{M}(u+\phi_f^T)$ respectively, and hence obtain

$$\mathcal{M}\phi_f^T = \begin{bmatrix} S_1\phi_f^T \\ S_2\phi_f^T \end{bmatrix}.$$

As in the repair process, since S_1, S_2 are symmetric, we finally obtain

$$\phi_f S_1 + \lambda_f \phi_f S_2 = \psi_f \mathcal{M}$$

which is the desired record X^f.

Analysis

The storage overhead is

$$\text{SO} = \frac{n\alpha}{m\ell} = \frac{n(m-1)}{m(m-1)} = \frac{n}{m}.$$

In particular, when $n = 2r = 4m-4$, $\text{SO} = \frac{4m-4}{m} < 4$. During the retrieval, we download $4m-4$ symbols to get a record of size $m-1$, thus

$$\text{cPoP} = \frac{4m-4}{m-1} = 4.$$

Table 5.1 shows the comparison between SO and cPoP of Construction 1 (PM-MSR-mixed) and Shah's scheme [36] using PM-MBR codes. Both constructions use the repair property of the regenerating codes in the retrieval, and require the connection to $n \geq 2r$ servers. Therefore, we compare the highest efficiency case when $n = 2r$. Recall that the Shah's scheme [36] has

$$\text{SO} = \frac{2r}{2r - m + 1} \cdot \frac{2r}{m}$$

and

$$\text{cPoP} = \frac{4r}{2r - m + 1} < 4$$

where $m \leq r \leq n - 1$. From Table 5.1, we can see that for the Shah's scheme [36], when r is the lowest possible ($r = m$), the scheme has lower SO for fixed m but it is still higher than the SO of Construction 1 (PM-MSR-mixed). However, the Shah's scheme [36] when $r = m$ has higher cPoP for fixed m but it is still lower than the cPoP of Construction 1 (PM-MSR-mixed). The repair ratio is

$$\frac{r}{\alpha} = \frac{2m - 2}{m - 1} = 2,$$

which is worse than the Shah's scheme [36] that has the least repair ratio RR = 1.

	Construction 1 (PM-MSR-mixed)		Shah's scheme (PM-MBR-mixed) [36]			
	when $r = 2m - 2$		when $r = m$		when $r = m + 1$	
m	SO	cPoP	SO	cPoP	SO	cPoP
2	2	4	2.67	2.67	3.6	2.4
3	2.67	4	3	3	3.56	2.67
4	3	4	3.2	3.2	3.57	2.86
5	3.2	4	3.33	3.33	3.6	3
6	3.33	4	3.43	3.43	3.63	3.11

Table 5.1: The comparison between SO and cPoP of Construction 1 and Shah's scheme [36].

5.2 Schemes with Separate Coding Architecture

Chan et al. initiate the work on code-based PIR with the separate coding architecture in [29] by proposing a general model for encoding a database where each record is divided into α stripes, each of length k, and then each stripe is encoded with an (n, k) linear code. Tajeddine et al. [30] subsequently present a PIR scheme that uses MDS codes in the encoding (Section 4.2.1.2). The retrieval technique exploits the recovery property of $[n, k]$ MDS codes; that is, any k out of n symbols can be used to reconstruct the original k symbols.

We consider PIR schemes using product-matrix regenerating codes with the separate coding architecture in this section. Indeed, for a database consisting of m records $X^1, \ldots, X^m$, each record X^j of size B is encoded via the $(n, k, r, \alpha, \beta, B)$ regenerating code by the product $\Psi \cdot \mathcal{M}^j$ where $\mathcal{M}^j$ is the corresponding message matrix of X^j. Hence the encoding function is

$$E_{m,n,m\alpha}(X^1, \ldots, X^m) = \Psi \cdot \mathcal{M},$$

where $\mathcal{M} = \begin{bmatrix} \mathcal{M}^1 & \cdots & \mathcal{M}^m \end{bmatrix}$. Thus, each server i stores the vector $\psi_i \cdot \mathcal{M}$ which has $m\alpha$ symbols in total. We apply the idea of retrieval technique in Section 4.2.1.2 by using the recovery property of the regenerating codes; that is, the original B symbols can be reconstructed from any k servers, each with α symbols. The difference is that in the MDS scheme, for α stripes of the desired record, we can secretly retrieve k symbols of each stripe from any different set of k servers. However, in the case of our schemes using regenerating codes, we need to secretly retrieve α symbols of the desired record that are stored in each server from one set of k servers.

Suppose we have a random column vector $u \in \mathbb{F}_q^{m\alpha}$. A common technique in PIR for generating a query to have access to the t^{th} symbol stored in server i, denoted by C_{it}, is to mix the random vector u with a unit vector e_t of length $m\alpha$

with 1 at the t^{th} position. Notice that if we project the vector u to server i_1, we get a symbol

$$(\psi_{i_1} \cdot \mathcal{M}) \cdot u,$$

and if we project the vector $u + e_t$ to server i_2, we get a symbol

$$(\psi_{i_2} \cdot \mathcal{M}) \cdot (u + e_t) = (\psi_{i_2} \cdot \mathcal{M}) \cdot u + C_{i_2,t}.$$

We know that any r rows of Ψ are linearly independent. Therefore, for a set of queries that is sent to n servers, the number of servers that receive only the random vector u needs to be at least r servers, so we have $(\psi_{i_1} \cdot \mathcal{M}) \cdot u, \ldots, (\psi_{i_r} \cdot \mathcal{M}) \cdot u$ to solve for $\mathcal{M} \cdot u$. Hence, we can have at most $n - r$ servers that receive a query as the sum of the random vector u and some unit vector, which means that we can access at most $n - r$ desired symbols per one set of queries.

In this section, we propose three PIR constructions. The first two constructions use PM-MBR and PM-MSR codes from Sections 2.2.2 and 2.2.3 respectively with $n = k + r$. Since we can access at most $n - r = k$ desired symbols per set of queries, we need α sets of queries. So we can secretly retrieve $k\alpha$ symbols in total to reconstruct the desired record using the recovery property of the regenerating codes. The last construction uses PM-MSR codes from Section 2.2.3 with $n = \alpha + r$, so we can secretly retrieve $k\alpha$ symbols in total by requesting k sets of queries.

Note that in Chapter 6 we propose three multi-message PIR schemes using PM-MSR and PM-MBR codes which are generalisations of the three constructions in this section. We will give a general MPIR model where the product-matrix regenerating codes are used for storage, and derive a trade-off between cPoP and RR. Hence, we omit the trade-off analysis of the following three constructions since it is a special case of the trade-off analysis in Chapter 6 when the number of records being retrieved is $p = 1$.

5.2.1 Construction 2 (PM-MBR-sep)

In this construction, we use a PM-MBR code from Section 2.2.2 with $n = k + r$ over the finite field $\mathbb{F}_q$ with parameters

$$(n, k, r, \alpha, r, B) = \left(k + r, k, r, r, 1, \frac{k(2r - k + 1)}{2}\right)$$

to store each record of size $\ell = \frac{k(2r-k+1)}{2}$. Note that if the size of records are longer than $\frac{k(2r-k+1)}{2}$, we divide each record of size ℓ to be chunks of size $\frac{k(2r-k+1)}{2}$, and we can operate each chunk independently and identically as discussed in Section 3.3. In this scheme, the queries submitted to the first k servers are designed to access symbols of the desired record that are stored in those k servers, so the desired record can be reconstructed by the property of the regenerating codes. The following example will illustrate how this scheme works.

Example 5.2. Suppose that we have 3 records, each of length 5. We write

$$\begin{aligned} X^1 &= \{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}, \\ X^2 &= \{x_{21}, x_{22}, x_{23}, x_{24}, x_{25}\}, \\ X^3 &= \{x_{31}, x_{32}, x_{33}, x_{34}, x_{35}\}. \end{aligned}$$

Each record is encoded by a $(5, 2, 3, 3, 1, 5)$ MBR code over $\mathbb{F}_7$ where the encoding matrix Ψ is the Vandermonde matrix

$$\Psi = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 2 \\ 1 & 4 & 2 \\ 1 & 5 & 4 \end{bmatrix},$$

server 1	server 2	server 3	server 4	server 5
$x_{11}+x_{12}+x_{14}$	$x_{11}+2x_{12}+4x_{14}$	$x_{11}+3x_{12}+2x_{14}$	$x_{11}+4x_{12}+2x_{14}$	$x_{11}+5x_{12}+4x_{14}$
$x_{12}+x_{13}+x_{15}$	$x_{12}+2x_{13}+4x_{15}$	$x_{12}+3x_{13}+2x_{15}$	$x_{12}+4x_{13}+2x_{15}$	$x_{12}+5x_{13}+4x_{15}$
$x_{14}+x_{15}$	$x_{14}+2x_{15}$	$x_{14}+3x_{15}$	$x_{14}+4x_{15}$	$x_{14}+5x_{15}$
$x_{21}+x_{22}+x_{24}$	$x_{21}+2x_{22}+4x_{24}$	$x_{21}+3x_{22}+2x_{24}$	$x_{21}+4x_{22}+2x_{24}$	$x_{21}+5x_{22}+4x_{24}$
$x_{22}+x_{23}+x_{25}$	$x_{22}+2x_{23}+4x_{25}$	$x_{22}+3x_{23}+2x_{25}$	$x_{22}+4x_{23}+2x_{25}$	$x_{22}+5x_{23}+4x_{25}$
$x_{24}+x_{25}$	$x_{24}+2x_{25}$	$x_{24}+3x_{25}$	$x_{24}+4x_{25}$	$x_{24}+5x_{25}$
$x_{31}+x_{32}+x_{34}$	$x_{31}+2x_{32}+4x_{34}$	$x_{31}+3x_{32}+2x_{34}$	$x_{31}+4x_{32}+2x_{34}$	$x_{31}+5x_{32}+4x_{34}$
$x_{32}+x_{33}+x_{35}$	$x_{32}+2x_{33}+4x_{35}$	$x_{32}+3x_{33}+2x_{35}$	$x_{32}+4x_{33}+2x_{35}$	$x_{32}+5x_{33}+4x_{35}$
$x_{34}+x_{35}$	$x_{34}+2x_{35}$	$x_{34}+3x_{35}$	$x_{34}+4x_{35}$	$x_{34}+5x_{35}$

Table 5.2: The storage of records X^1, X^2, X^3 in the servers

and the message matrix $\mathcal{M}^j$ for the record $j, j \in \{1,2,3\}$ is as described in Section 2.2.2

$$\mathcal{M}^j = \begin{bmatrix} x_{j1} & x_{j2} & x_{j4} \\ x_{j2} & x_{j3} & x_{j5} \\ x_{j4} & x_{j5} & 0 \end{bmatrix}.$$

Hence, each server stores $\Psi \cdot \mathcal{M}$ as shown in Table 5.2. Denote C_{ib}^j to be the b^{th} symbol stored in server i of the record j.

In the retrieval step, suppose the user wants record X^1. The query Q^i is an $(\alpha \times m\alpha) = (3 \times 9)$ matrix which we can interpret as 3 subqueries submitted to server i for each $i \in [5]$. To form the query matrices, the user generates a (3×9) random matrix $U = [u_{ij}]$ whose elements are chosen uniformly at random from $\mathbb{F}_7$. Let

$$V^1 = V^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For server $i = 1, 2$, the query matrix is $Q^i = U + V^i$ and $Q^3 = Q^4 = Q^5 = U$. Then each server computes and returns the length-3 vector $A^i = Q^i C_i^T$ where C_i^T is a

length-9 vector of symbols stored in server i. Write $A^i = (A_1^i, A_2^i, A_3^i)^T$. Let

$$\mathcal{M}_1 = (x_{11}, x_{12}, x_{14}, x_{21}, x_{22}, x_{24}, x_{31}, x_{32}, x_{34}),$$
$$\mathcal{M}_2 = (x_{12}, x_{13}, x_{15}, x_{22}, x_{23}, x_{25}, x_{32}, x_{33}, x_{35}),$$
$$\mathcal{M}_3 = (x_{14}, x_{15}, 0, x_{24}, x_{25}, 0, x_{34}, x_{35}, 0).$$

Consider first the subquery 1; we obtain

$$C_{11}^1 + I_1^1 + I_2^1 + I_3^1 = A_1^1, \tag{1}$$
$$C_{21}^1 + I_1^1 + 2I_2^1 + 4I_3^1 = A_1^2, \tag{2}$$
$$I_1^1 + 3I_2^1 + 2I_3^1 = A_1^3, \tag{3}$$
$$I_1^1 + 4I_2^1 + 2I_3^1 = A_1^4, \tag{4}$$
$$I_1^1 + 5I_2^1 + 4I_3^1 = A_1^5, \tag{5}$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T, h = 1, 2, 3$, and U_1 is the first row of U. The user can solve for I_1^1, I_2^1, I_3^1 from $(3), (4), (5)$ as they form the equation

$$\begin{bmatrix} 1 & 3 & 2 \\ 1 & 4 & 2 \\ 1 & 5 & 4 \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ I_3^1 \end{bmatrix} = \begin{bmatrix} A_1^3 \\ A_1^4 \\ A_1^5 \end{bmatrix},$$

where the left matrix is a (3×3) submatrix of Ψ which is invertible. Therefore, the user gets C_{11}^1, C_{21}^1, which are all the symbols with label 1 in Table 5.3. Similarly, from subqueries 2 and 3, the user obtains C_{12}^1, C_{22}^1 and C_{13}^1, C_{23}^1, respectively. Hence, the user has all the symbols of X^1 which are stored in the first 2 servers. From the property of the regenerating codes, the user can reconstruct X^1 as desired.

Next we formally give the details of this construction.

Encoding Step

For Construction 2, we use a $\left(k+r, k, r, r, 1, \frac{k(2r-k+1)}{2}\right)$ MBR code over $\mathbb{F}_q$ to store each record $X^1, \ldots, X^m$. This means that the encoding function is

$$E_{1,k+r,r}(X^j) = \Psi \cdot \mathcal{M}^j,$$

where $\mathcal{M}^j$ is the message matrix corresponding to X^j as described in Section 2.2.2. Write

$$\mathcal{M} = \begin{bmatrix} \mathcal{M}^1 & \cdots & \mathcal{M}^m \end{bmatrix},$$

and denote by $\mathcal{M}_i$ the row i of $\mathcal{M}$, so

$$E_{m,k+r,mr}(X^1, \ldots, X^m) = \begin{bmatrix} \Psi \cdot \mathcal{M}^1 & \cdots & \Psi \cdot \mathcal{M}^m \end{bmatrix} = \Psi \cdot \mathcal{M}.$$

We denote by C_i the i^{th} row of $E_{m,k+r,mr}(X^1, \ldots, X^m)$ which consists of all $m\alpha$ symbols stored in server i.

Retrieval Step

Suppose that the user wants record X^f. In the retrieval step, the user sends an $(\alpha \times m\alpha)$ query matrix Q^i, which we can interpret as α subqueries, to each server $i, i = 1, \ldots, n$. To form the query matrices, the user generates an $(\alpha \times m\alpha)$ random matrix $U = [u_{ij}]$ whose elements are chosen uniformly at random from $\mathbb{F}_q$. Then

server 1	server 2	server 3	server 4	server 5
1	1			
2	2			
3	3			

Table 5.3: Retrieval pattern for a (5,2,3,3,1,5) MBR code. The 3×5 entries correspond to the first three rows in Table 5.2, which are symbols of X^1 stored in the system. The entries labelled by the same number, say d, are privately retrieved by subquery d.

the query matrices are generated as $Q^i = U + V^i$ when $i = 1, \ldots, k$ and $Q^i = U$ when $i = k+1, \ldots, n$, where V^i is a deterministic binary matrix. The matrix V^i is designed to have access to all the symbols of the requested record X^f that are stored in the first k servers. We choose

$$V^i = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha\times(f-1)\alpha} & I_{\alpha\times\alpha} & \mathbf{0}_{\alpha\times(m-f)\alpha} \end{array}\right]$$

for $i = 1, \ldots, k$. Then, each server returns the length-r vector $A^i = Q^i C_i^T$, and we write $A^i = (A_1^i, A_2^i, \ldots, A_\alpha^i)^T$.

Theorem 5.3. Construction 2 (PM-MBR-sep) is information-theoretically perfect.

Proof. Decodability: We can see that for $i = 1, \ldots, n$,

$$\begin{aligned} C_i &= \psi_i \cdot \mathcal{M} \\ &= \psi_i \cdot \begin{bmatrix} \text{—} & \mathcal{M}_1 & \text{—} \\ \text{—} & \mathcal{M}_2 & \text{—} \\ & \vdots & \\ \text{—} & \mathcal{M}_r & \text{—} \end{bmatrix} = \sum_{h=1}^{r} \psi_{ih} \mathcal{M}_h. \end{aligned}$$

Thus,

$$C_i^T = \sum_{h=1}^{r} \psi_{ih} \mathcal{M}_h^T.$$

Denote e_t to be the length-$m\alpha$ binary unit vector with 1 at the t^{th} position. Consider

first the subquery 1; we obtain

$$C_{11}^f + \sum_{h=1}^{r} \psi_{1h} I_h^1 = (U_1 + e_{(f-1)\alpha+1}) C_1^T = A_1^1, \tag{1}$$

$$C_{21}^f + \sum_{h=1}^{r} \psi_{2h} I_h^1 = (U_1 + e_{(f-1)\alpha+1}) C_2^T = A_1^2, \tag{2}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$C_{k,1}^f + \sum_{h=1}^{r} \psi_{k,h} I_h^1 = (U_1 + e_{(f-1)\alpha+1}) C_k^T = A_1^k, \tag{k}$$

$$\sum_{h=1}^{r} \psi_{k+1,h} I_h^1 = U_1 C_{k+1}^T \qquad = A_1^{k+1}, \tag{k+1}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$\sum_{h=1}^{r} \psi_{n,h} I_h^1 = U_1 C_n^T \qquad = A_1^n, \tag{n}$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T$, $h = 1, 2, \ldots, r$, and U_1 is the first row of U. The user can solve for $I_1^1, \ldots, I_r^1$ from $(k+1), \ldots, (n)$ as they form the equation

$$\begin{bmatrix} \text{—} & \psi_{k+1} & \text{—} \\ & \vdots & \\ \text{—} & \psi_n & \text{—} \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ \vdots \\ I_r^1 \end{bmatrix} = \begin{bmatrix} A_1^{k+1} \\ \vdots \\ A_1^n \end{bmatrix},$$

where, since $n = k + r$, the left matrix is an $(r \times r)$ square submatrix of Ψ which is invertible. Therefore, the user gets $C_{11}^1, C_{21}^1, \ldots, C_{k,1}^1$, which are all the symbols with label 1 in Table 5.4. Similarly, from subqueries $h = 2, \ldots, \alpha$, the user obtains $C_{1h}^1, C_{2h}^1, \ldots, C_{k,h}^1$. Hence, the user has all the $k\alpha$ symbols of X^f which are stored in the first k servers. From the property of the regenerating codes, the user can reconstruct X^f as desired.

Privacy: To prove the privacy of the scheme, as we construct the query matrices

server 1	server 2	server 3	$\cdots$	server k	server $k+1$	$\cdots$	server n
1	1	1	$\cdots$	1			
2	2	2	$\cdots$	2			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
α	α	α	$\cdots$	α			

Table 5.4: Retrieval pattern for a $(k+r, k, r, r, 1, \frac{k(2r-k+1)}{2})$ MBR code. The $\alpha \times n$ entries correspond to the symbols of X^f stored in the system. The entries labelled by the same number, say d, are privately retrieved by subquery d.

Q^i via the random matrix U, Q^i is independent from f which implies that this scheme achieves perfect privacy.

□

Analysis

The storage overhead is

$$\frac{n(m\alpha)}{mB} = \frac{(k+r)r}{\frac{(k+1)k}{2} + k(r-k)},$$

and the cPoP is equal to

$$\frac{n\alpha}{B} = \frac{(k+r)r}{\frac{(k+1)k}{2} + k(r-k)}.$$

The smallest SO and cPoP occur when $r = k$ which are both equal to

$$\frac{4k}{k+1} < 4$$

which are exactly the same as the SO and cPoP of the Shah's scheme (PM-MBR-mixed) in [36] when $n = 2r = 2m$ (as the number of records m is fixed to be k). However, the advantage of this construction is that the number of records m is independent from the parameter k.

As the capacity of PIR using regenerating codes is unknown, we now compare the retrieval rate of our schemes with the capacity of the PIR using MDS codes in order to identify the gap. Recall that the capacity of PIR with MDS coded database is

$$C_{MDS} = \frac{1 - \frac{k}{n}}{1 - (\frac{k}{n})^m}.$$

Table 5.5 shows the comparison between retrieval rate of Construction 2 (PM-MBR-sep) and the capacity of PIR using MDS codes for the same parameters m, k, n. We can see that our retrieval rate is not as good as the MDS capacity, but it is closer to the capacity when the number of records m is larger. For example, when $k = 3$, the percentage difference is asymptotically 33.33% as m approaches infinity. The advantage of the use of PM-MBR codes is that the repair ratio of this scheme is

$$\frac{mr}{m\alpha} = 1$$

which is the lowest possible, while the MDS schemes give a repair ratio of $k > 1$.

	k	n	R_{PIR} of Construction 2	C_{MDS} (4.3)	% difference
$m = 3$	3	6	0.3333	0.5714	41.6667
	5	10	0.3000	0.5714	47.5
	8	16	0.2813	0.5714	50.7813
	10	20	0.2750	0.5714	51.8750
$m = 5$	3	6	0.3333	0.5161	35.4167
	5	10	0.3000	0.5161	41.8750
	8	16	0.2813	0.5161	45.5078
	10	20	0.2750	0.5161	46.7188
$m = 7$	3	6	0.3333	0.5039	33.8542
	5	10	0.3000	0.5039	40.4688
	8	16	0.2813	0.5039	44.1895
	10	20	0.2750	0.5039	45.4297

Table 5.5: The comparison between retrieval rate of Construction 2 and the capacity of PIR using MDS codes

5.2.2 Construction 3 (PM-MSR-sep)

Our idea of this construction is to modify Construction 2 (PM-MBR-sep) by using the PM-MSR code from Section 2.2.3 with $n = k + r$ over the finite field $\mathbb{F}_q$ with parameters

$$(3k-2, k, 2k-2, k-1, 1, k(k-1))$$

to store each record of size $\ell = k(k-1)$. In this scheme, the queries submitted to the first k servers are designed to access symbols of the desired record that are stored in those k servers, so the desired record can be reconstructed by the property of the regenerating codes. The retrieval pattern in this scheme is similar to Construction 2 (PM-MBR-sep). The following example will give an illustration of this scheme.

Example 5.4. Suppose that we have 3 records, each of length 6. We write

$$\begin{aligned} X^1 &= \{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}\}, \\ X^2 &= \{x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}\}, \\ X^3 &= \{x_{31}, x_{32}, x_{33}, x_{34}, x_{35}, x_{36}\}. \end{aligned}$$

Each record is encoded by a $(7, 3, 4, 2, 1, 6)$ MSR code over $\mathbb{F}_{13}$ where the encoding matrix Ψ is the Vandermonde matrix

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 1 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \\ 1 & 7 & 10 & 5 \end{bmatrix},$$

server 1	server 2	server 3	server 4
$x_{11}+x_{12}+x_{14}+x_{15}$	$x_{11}+2x_{12}+4x_{14}+8x_{15}$	$x_{11}+3x_{12}+9x_{14}+x_{15}$	$x_{11}+4x_{12}+3x_{14}+12x_{15}$
$x_{12}+x_{13}+x_{15}+x_{16}$	$x_{12}+2x_{13}+4x_{15}+8x_{16}$	$x_{12}+3x_{13}+9x_{15}+x_{16}$	$x_{12}+4x_{13}+3x_{15}+12x_{16}$
$x_{21}+x_{22}+x_{24}+x_{25}$	$x_{21}+2x_{22}+4x_{24}+8x_{25}$	$x_{21}+3x_{22}+9x_{24}+x_{25}$	$x_{21}+4x_{22}+3x_{24}+12x_{25}$
$x_{22}+x_{23}+x_{25}+x_{26}$	$x_{22}+2x_{23}+4x_{25}+8x_{26}$	$x_{22}+3x_{23}+9x_{25}+x_{26}$	$x_{22}+4x_{23}+3x_{25}+12x_{26}$
$x_{31}+x_{32}+x_{34}+x_{35}$	$x_{31}+2x_{32}+4x_{34}+8x_{35}$	$x_{31}+3x_{32}+9x_{34}+x_{35}$	$x_{31}+4x_{32}+3x_{34}+12x_{35}$
$x_{32}+x_{33}+x_{35}+x_{36}$	$x_{32}+2x_{33}+4x_{35}+8x_{36}$	$x_{32}+3x_{33}+9x_{35}+x_{36}$	$x_{32}+4x_{33}+3x_{35}+12x_{36}$

server 5	server 6	server 7
$x_{11}+5x_{12}+12x_{14}+8x_{15}$	$x_{11}+6x_{12}+10x_{14}+8x_{15}$	$x_{11}+7x_{12}+10x_{14}+5x_{15}$
$x_{12}+5x_{13}+12x_{15}+8x_{16}$	$x_{12}+6x_{13}+10x_{15}+8x_{16}$	$x_{12}+7x_{13}+10x_{15}+5x_{16}$
$x_{21}+5x_{22}+12x_{24}+8x_{25}$	$x_{21}+6x_{22}+10x_{24}+8x_{25}$	$x_{21}+7x_{22}+10x_{24}+5x_{25}$
$x_{22}+5x_{23}+12x_{25}+8x_{26}$	$x_{22}+6x_{23}+10x_{25}+8x_{26}$	$x_{22}+7x_{23}+10x_{25}+5x_{26}$
$x_{31}+5x_{32}+12x_{34}+8x_{35}$	$x_{31}+6x_{32}+10x_{34}+8x_{35}$	$x_{31}+7x_{32}+10x_{34}+5x_{35}$
$x_{32}+5x_{33}+12x_{35}+8x_{36}$	$x_{32}+6x_{33}+10x_{35}+8x_{36}$	$x_{32}+7x_{33}+10x_{35}+5x_{36}$

Table 5.6: The storage of records X^1, X^2, X^3 in the servers

and the message matrix $\mathcal{M}^j$ for the record $j, j \in \{1,2,3\}$ is as described in Section 2.2.3

$$\mathcal{M}^j = \begin{bmatrix} x_{j1} & x_{j2} \\ x_{j2} & x_{j3} \\ x_{j4} & x_{j5} \\ x_{j5} & x_{j6} \end{bmatrix}.$$

Hence, each server stores $\Psi \cdot \mathcal{M}$ as shown in Table 5.6. Denote C_{ib}^j to be the b^{th} symbol stored in server i of the record j.

In the retrieval step, suppose the user wants record X^1. The query Q^i is an $(\alpha \times m\alpha) = (2 \times 6)$ matrix which we can interpret as 2 subqueries submitted to server i for each $i \in [7]$. To form the query matrices, the user generates a (2×6) random matrix $U = [u_{ij}]$ whose elements are chosen uniformly at a random from $\mathbb{F}_{13}$. Let

$$V^1 = V^2 = V^3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For server $i = 1, 2, 3$, the query matrix is $Q^i = U + V^i$ and $Q^4 = Q^5 = Q^6 = Q^7 = U$.

Then each server computes and returns the length-2 vector $A^i = Q^i C_i^T$ where C_i^T is a length-6 vector of symbols stored in server i. Write $A^i = (A_1^i, A_2^i)^T$. Let

$$\mathcal{M}_1 = (x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32}),$$
$$\mathcal{M}_2 = (x_{12}, x_{13}, x_{22}, x_{23}, x_{32}, x_{33}),$$
$$\mathcal{M}_3 = (x_{14}, x_{15}, x_{24}, x_{25}, x_{34}, x_{35}),$$
$$\mathcal{M}_4 = (x_{15}, x_{16}, x_{25}, x_{26}, x_{35}, x_{36}).$$

Consider first the subquery 1; we obtain

$$C_{11}^1 + I_1^1 + I_2^1 + I_3^1 + I_4^1 = A_1^1, \tag{1}$$
$$C_{21}^1 + I_1^1 + 2I_2^1 + 4I_3^1 + 8I_4^1 = A_1^2, \tag{2}$$
$$C_{31}^1 + I_1^1 + 3I_2^1 + 9I_3^1 + I_4^1 = A_1^3, \tag{3}$$
$$I_1^1 + 4I_2^1 + 3I_3^1 + 12I_4^1 = A_1^4, \tag{4}$$
$$I_1^1 + 5I_2^1 + 12I_3^1 + 8I_4^1 = A_1^5, \tag{5}$$
$$I_1^1 + 6I_2^1 + 10I_3^1 + 8I_4^1 = A_1^6, \tag{6}$$
$$I_1^1 + 7I_2^1 + 10I_3^1 + 5I_4^1 = A_1^7, \tag{7}$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T, h = 1, 2, 3, 4$, and U_1 is the first row of U. The user can solve for $I_1^1, I_2^1, I_3^1, I_4^1$ from $(4), (5), (6), (7)$ as they form the equation

$$\begin{bmatrix} 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \\ 1 & 7 & 10 & 5 \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ I_3^1 \\ I_4^1 \end{bmatrix} = \begin{bmatrix} A_1^4 \\ A_1^5 \\ A_1^6 \\ A_1^7 \end{bmatrix},$$

where the left matrix is a (4×4) submatrix of Ψ which is invertible. Therefore,

the user gets $C_{11}^1, C_{21}^1, C_{31}^1$, which are all the symbols with label 1 in Table 5.7. Similarly, from subquery 2, the user obtains $C_{12}^1, C_{22}^1, C_{32}^1$. Hence, the user has all the symbols of X^1 which are stored in the first 3 servers. From the property of the regenerating codes, the user can reconstruct X^1 as desired.

server 1	server 2	server 3	server 4	server 5	server 6	server 7
1	1	1				
2	2	2				

Table 5.7: Retrieval pattern for a (7,3,4,2,1,6) MSR code. The 2×7 entries correspond to the first two rows in Table 5.6, which are symbols of X^1 stored in the system. The entries labelled by the same number, say d, are privately retrieved by subquery d.

Next we formally give the details of this construction.

Encoding Step

For Construction 3, we use a $(3k-2, k, 2k-2, k-1, 1, k(k-1))$ MSR code over $\mathbb{F}_q$ to store each record $X^1, \ldots, X^m$. This means that the encoding function is

$$E_{1,3k-2,k-1}(X^j) = \Psi \cdot \mathcal{M}^j,$$

where $\mathcal{M}^j$ is the message matrix corresponding to X^j as described in Section 2.2.3. Write

$$\mathcal{M} = \begin{bmatrix} \mathcal{M}^1 & \cdots & \mathcal{M}^m \end{bmatrix},$$

and denote by $\mathcal{M}_i$ the row i of $\mathcal{M}$, so

$$E_{m,3k-2,m(k-1)}(X^1, \ldots, X^m) = \begin{bmatrix} \Psi \cdot \mathcal{M}^1 & \cdots & \Psi \cdot \mathcal{M}^m \end{bmatrix} = \Psi \cdot \mathcal{M}.$$

We denote by C_i the i^{th} row of $E_{m,3k-2,m(k-1)}(X^1, \ldots, X^m)$ which consists of all $m\alpha$ symbols stored in server i.

Retrieval Step

Suppose that the user wants record X^f. In the retrieval step, the user sends an $(\alpha \times m\alpha)$ query matrix Q^i, which we can interpret as $\alpha = k-1$ subqueries, to each server $i, i = 1, \ldots, n$. To form the query matrices, the user generates an $(\alpha \times m\alpha)$ random matrix $U = [u_{ij}]$ whose elements are chosen uniformly at random from $\mathbb{F}_q$. Then the query matrices are generated as $Q^i = U + V^i$ when $i = 1, \ldots, k$ and $Q^i = U$ when $i = k+1, \ldots, n$, where V^i is a deterministic binary matrix. The matrix V^i is designed to have access to all the symbols of the requested record X^f that are stored in the first k servers. We choose

$$V^i = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha \times (f-1)\alpha} & I_{\alpha \times \alpha} & \mathbf{0}_{\alpha \times (m-f)\alpha} \end{array} \right]$$

for $i = 1, \ldots, k$. Then, each server returns the length-$(k-1)$ vector $A^i = Q^i C_i^T$, and we write $A^i = (A_1^i, A_2^i, \ldots, A_\alpha^i)^T$.

Theorem 5.5. Construction 3 (PM-MSR-sep) is information-theoretically perfect.

Proof. Decodability: We can see that for $i = 1, \ldots, n$,

$$\begin{aligned} C_i &= \psi_i \cdot \mathcal{M} \\ &= \psi_i \cdot \begin{bmatrix} \text{—} & \mathcal{M}_1 & \text{—} \\ \text{—} & \mathcal{M}_2 & \text{—} \\ & \vdots & \\ \text{—} & \mathcal{M}_{2k-2} & \text{—} \end{bmatrix} = \sum_{h=1}^{2k-2} \psi_{ih} \mathcal{M}_h. \end{aligned}$$

Thus,

$$C_i^T = \sum_{h=1}^{2k-2} \psi_{ih} \mathcal{M}_h^T.$$

Denote e_t to be the length-$m\alpha$ binary unit vector with 1 at the t^{th} position. Consider

first the subquery 1; we obtain

$$
\begin{aligned}
C_{11}^f + \sum_{h=1}^{2k-2} \psi_{1h} I_h^1 &= (U_1 + e_{(f-1)\alpha+1}) C_1^T = A_1^1, && (1)\\
C_{21}^f + \sum_{h=1}^{2k-2} \psi_{2h} I_h^1 &= (U_1 + e_{(f-1)\alpha+1}) C_2^T = A_1^2, && (2)\\
&\vdots \qquad\qquad\qquad \vdots \\
C_{k,1}^f + \sum_{h=1}^{2k-2} \psi_{k,h} I_h^1 &= (U_1 + e_{(f-1)\alpha+1}) C_k^T = A_1^k, && (\text{k})\\
\sum_{h=1}^{2k-2} \psi_{k+1,h} I_h^1 &= U_1 C_{k+1}^T \qquad\qquad = A_1^{k+1}, && (\text{k+1})\\
&\vdots \qquad\qquad\qquad \vdots \\
\sum_{h=1}^{2k-2} \psi_{n,h} I_h^1 &= U_1 C_n^T \qquad\qquad = A_1^n, && (\text{n})
\end{aligned}
$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T$, $h = 1, 2, \ldots, 2k-2$, and U_1 is the first row of U. The user can solve for $I_1^1, \ldots, I_{2k-2}^1$ from $(k+1), \ldots, (n)$ as they form the equation

$$
\begin{bmatrix} \text{—} & \psi_{k+1} & \text{—} \\ & \vdots & \\ \text{—} & \psi_n & \text{—} \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ \vdots \\ I_{2k-2}^1 \end{bmatrix} = \begin{bmatrix} A_1^{k+1} \\ \vdots \\ A_1^n \end{bmatrix},
$$

where, since $n = 3k-2$, the left matrix is a $((2k-2) \times (2k-2))$ square submatrix of Ψ which is invertible. Therefore, the user gets $C_{11}^1, C_{21}^1, \ldots, C_{k,1}^1$, i.e., get all the symbols with label 1 in Table 5.8. Similarly, from subqueries $h = 2, \ldots, \alpha$, the user obtains $C_{1h}^1, C_{2h}^1, \ldots, C_{k,h}^1$. Hence, the user has all the $k\alpha$ symbols of X^f which are stored in the first k servers. From the property of the regenerating codes, the user can reconstruct X^f as desired.

Privacy: To prove the privacy of the scheme, as we construct the query matrices

server 1	server 2	server 3	$\cdots$	server k	server $k+1$	$\cdots$	server n
1	1	1	$\cdots$	1			
2	2	2	$\cdots$	2			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
α	α	α	$\cdots$	α			

Table 5.8: Retrieval pattern for a $(3k-2, k, 2k-2, k-1, 1, k(k-1))$ MSR code. The $\alpha \times n$ entries correspond to the symbols of X^f stored in the system. The entries labelled by the same number, say d, are privately retrieved by subquery d.

Q^i via the random matrix U, Q^i is independent from f which implies that this scheme achieves perfect privacy.

□

Analysis

In this scheme, storage overhead is

$$\frac{n(m\alpha)}{mB} = \frac{(3k-2)(k-1)}{k(k-1)} = \frac{3k-2}{k} < 3,$$

and cPoP equals

$$\frac{n\alpha}{B} = \frac{(3k-2)(k-1)}{k(k-1)} = \frac{3k-2}{k} < 3,$$

which are both better than Construction 1 (PM-MSR-mixed), and Construction 2 (PM-MBR-sep).

	k	n	R_{PIR} of Construction 3	C_{MDS} (4.3)	% difference
$m = 3$	3	7	0.4286	0.6203	30.9038
	5	13	0.3846	0.6525	41.0560
	8	22	0.3636	0.6685	45.6048
	10	28	0.3571	0.6735	46.9752
$m = 5$	3	7	0.4286	0.5798	26.0844
	5	13	0.3846	0.6206	38.0260
	8	22	0.3636	0.6404	43.2205
	10	28	0.3571	0.6466	44.7672
$m = 7$	3	7	0.4286	0.5730	25.1992
	5	13	0.3846	0.6162	37.5778
	8	22	0.3636	0.6369	42.9052
	10	28	0.3571	0.6433	44.4856

Table 5.9: The comparison between the retrieval rate of Construction 3 and the capacity of PIR using MDS codes.

Table 5.9 shows the comparison between the retrieval rate of Construction 3 (PM-MSR-sep) and the capacity of PIR using MDS codes for the same parameters m, k, n. The retrieval rate of Construction 3 is closer to the MDS capacity when the number of records m is larger. For example, when $k = 3$, the percentage difference is asymptotically 25% as m approaches infinity. The repair ratio for this scheme is

$$\frac{mr}{m\alpha} = \frac{(2k-2)}{k-1} = 2,$$

which is higher than Construction 2 (PM-MBR-sep). However it is still smaller than PIR schemes using MDS codes (for example, in Section 4.2.1.2), where the repair ratio is k, when $k > 2$.

5.2.3 Construction 4 (PM-MSR-sep)

In this construction, we also use the PM-MSR codes from Section 2.2.3. Here we assume that the number of servers is $n = \alpha + r$ over the finite field $\mathbb{F}_q$, so the parameters of the PM-MSR codes are

$$(n, k, r, \alpha, \beta, B) = (3k - 3, k, 2k - 2, k - 1, 1, k(k - 1)).$$

In this scheme, the queries submitted to the first k servers are designed to access symbols of the desired record that are stored in those k servers. However, the retrieval pattern in this construction is different from Constructions 2 and 3, and the number of subqueries is k instead of α. We first start with an example to motivate our scheme.

Example 5.6. Suppose that we have 3 records, each of length 6. We write

$$\begin{aligned} X^1 &= \{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}\}, \\ X^2 &= \{x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}\}, \\ X^3 &= \{x_{31}, x_{32}, x_{33}, x_{34}, x_{35}, x_{36}\}. \end{aligned}$$

Each record is encoded by a $(6, 3, 4, 2, 1, 6)$ MSR code over $\mathbb{F}_{13}$ where the encoding matrix Ψ is the Vandermonde matrix

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 1 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \end{bmatrix},$$

server 1	server 2	server 3
$x_{11}+x_{12}+x_{14}+x_{15}$	$x_{11}+2x_{12}+4x_{14}+8x_{15}$	$x_{11}+3x_{12}+9x_{14}+x_{15}$
$x_{12}+x_{13}+x_{15}+x_{16}$	$x_{12}+2x_{13}+4x_{15}+8x_{16}$	$x_{12}+3x_{13}+9x_{15}+x_{16}$
$x_{21}+x_{22}+x_{24}+x_{25}$	$x_{21}+2x_{22}+4x_{24}+8x_{25}$	$x_{21}+3x_{22}+9x_{24}+x_{25}$
$x_{22}+x_{23}+x_{25}+x_{26}$	$x_{22}+2x_{23}+4x_{25}+8x_{26}$	$x_{22}+3x_{23}+9x_{25}+x_{26}$
$x_{31}+x_{32}+x_{34}+x_{35}$	$x_{31}+2x_{32}+4x_{34}+8x_{35}$	$x_{31}+3x_{32}+9x_{34}+x_{35}$
$x_{32}+x_{33}+x_{35}+x_{36}$	$x_{32}+2x_{33}+4x_{35}+8x_{36}$	$x_{32}+3x_{33}+9x_{35}+x_{36}$

server 4	server 5	server 6
$x_{11}+4x_{12}+3x_{14}+12x_{15}$	$x_{11}+5x_{12}+12x_{14}+8x_{15}$	$x_{11}+6x_{12}+10x_{14}+8x_{15}$
$x_{12}+4x_{13}+3x_{15}+12x_{16}$	$x_{12}+5x_{13}+12x_{15}+8x_{16}$	$x_{12}+6x_{13}+10x_{15}+8x_{16}$
$x_{21}+4x_{22}+3x_{24}+12x_{25}$	$x_{21}+5x_{22}+12x_{24}+8x_{25}$	$x_{21}+6x_{22}+10x_{24}+8x_{25}$
$x_{22}+4x_{23}+3x_{25}+12x_{26}$	$x_{22}+5x_{23}+12x_{25}+8x_{26}$	$x_{22}+6x_{23}+10x_{25}+8x_{26}$
$x_{31}+4x_{32}+3x_{34}+12x_{35}$	$x_{31}+5x_{32}+12x_{34}+8x_{35}$	$x_{31}+6x_{32}+10x_{34}+8x_{35}$
$x_{32}+4x_{33}+3x_{35}+12x_{36}$	$x_{32}+5x_{33}+12x_{35}+8x_{36}$	$x_{32}+6x_{33}+10x_{35}+8x_{36}$

Table 5.10: The storage of records X^1, X^2, X^3 in the servers

and the message matrix $\mathcal{M}^j$ for the record $j, j \in \{1,2,3\}$ is as described in Section 2.2.3

$$\mathcal{M}^j = \begin{bmatrix} x_{j1} & x_{j2} \\ x_{j2} & x_{j3} \\ x_{j4} & x_{j5} \\ x_{j5} & x_{j6} \end{bmatrix}.$$

Hence, each server stores $\Psi \cdot \mathcal{M}$ as shown in Table 5.10. Denote C_{ib}^j to be the b^{th} symbol stored in server i of the record j.

In the retrieval step, suppose the user wants record X^1. The query Q^i is a $(k \times m\alpha) = (3 \times 6)$ matrix which we can interpret as $k = 3$ subqueries submitted to server i for each $i \in [6]$. To form the query matrices, the user generates a (3×6) random matrix $U = [u_{ij}]$ whose elements are chosen uniformly at random from $\mathbb{F}_{13}$. Let

$$V^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad V^2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$V^3 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For server $i = 1, 2, 3$, the query matrix is $Q^i = U + V^i$, and $Q^4 = Q^5 = Q^6 = U$. Then each server computes and returns the length-3 vector $A^i = Q^i C_i^T$ where C_i^T is a length-6 vector of symbols stored in server i. Write $A^i = (A_1^i, A_2^i, A_3^i)^T$. Let

$$\mathcal{M}_1 = (x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32})^T,$$
$$\mathcal{M}_2 = (x_{12}, x_{13}, x_{22}, x_{23}, x_{32}, x_{33})^T,$$
$$\mathcal{M}_3 = (x_{14}, x_{15}, x_{24}, x_{25}, x_{34}, x_{35})^T,$$
$$\mathcal{M}_4 = (x_{15}, x_{16}, x_{25}, x_{26}, x_{35}, x_{36})^T.$$

Consider first the subquery 1; we obtain

$$C_{11}^1 + I_1^1 + I_2^1 + I_3^1 + I_4^1 = A_1^1, \tag{1}$$
$$I_1^1 + 2I_2^1 + 4I_3^1 + 8I_4^1 = A_1^2, \tag{2}$$
$$C_{32}^1 + I_1^1 + 3I_2^1 + 9I_3^1 + I_4^1 = A_1^3, \tag{3}$$
$$I_1^1 + 4I_2^1 + 3I_3^1 + 12I_4^1 = A_1^4, \tag{4}$$
$$I_1^1 + 5I_2^1 + 12I_3^1 + 8I_4^1 = A_1^5, \tag{5}$$
$$I_1^1 + 6I_2^1 + 10I_3^1 + 8I_4^1 = A_1^6, \tag{6}$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T, h = 1, 2, 3, 4$, and U_1 is the first row of U. The user can solve

for $I_1^1, I_2^1, I_3^1, I_4^1$ from (2), (4), (5), (6) as they form the equation

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ I_3^1 \\ I_4^1 \end{bmatrix} = \begin{bmatrix} A_1^2 \\ A_1^4 \\ A_1^5 \\ A_1^6 \end{bmatrix},$$

where the left matrix is a (4×4) submatrix of Ψ which is invertible. Therefore, the user gets C_{11}^1 and C_{32}^1, which are all the symbols with label 1 in Table 5.11. Similarly, from subqueries 2, 3, the user obtains C_{12}^1, C_{21}^1 and C_{22}^1, C_{31}^1, respectively. Hence, the user has all the symbols of X^1 which are stored in the first 3 servers. From the property of the regenerating codes, the user can reconstruct X^1 as desired.

server 1	server 2	server 3	server 4	server 5	server 6
1	2	3			
2	3	1			

Table 5.11: Retrieval pattern for a (6,3,4,2,1,6) MSR code. The 2×6 entries correspond to the first two rows in Table 5.10, which are symbols of X^1 stored in the system. The entries labelled by the same number, say d, are privately retrieved by subquery d.

Next we formally give the details of this construction.

Encoding Step

Recall that we use a $(3k-3, k, 2k-2, k-1, 1, k(k-1))$ MSR code over $\mathbb{F}_q$ to store each record $X^1, \ldots, X^m$. This means that the encoding function is

$$E_{1,3k-3,k-1}(X^j) = \Psi \cdot \mathcal{M}^j,$$

where $\mathcal{M}^j$ is the message matrix corresponding to X^j as described in Section 2.2.3. Write

$$\mathcal{M} = \begin{bmatrix} \mathcal{M}^1 & \cdots & \mathcal{M}^m \end{bmatrix},$$

and denote by $\mathcal{M}_i$ the row i of $\mathcal{M}$, so

$$E_{m,3k-3,m(k-1)}(X^1,\ldots,X^m) = \left[\Psi\cdot\mathcal{M}^1 \quad \cdots \quad \Psi\cdot\mathcal{M}^m\right] = \Psi\cdot\mathcal{M}.$$

We denote by C_i the i^{th} row of $E_{m,3k-3,m(k-1)}(X^1,\ldots,X^m)$ which consists of all $m\alpha$ symbols stored in server i.

Retrieval Step

Suppose that the user wants record X^f. In the retrieval step, the user sends a $(k\times m\alpha)$ query matrix Q^i, which we can interpret as k subqueries, to each server $i, i=1,\ldots,n$. To form the query matrices, the user generates a $(k\times m\alpha)$ random matrix $U=[u_{ij}]$ whose elements are chosen uniformly at a random from $\mathbb{F}_q$. Then the query matrices are generated as $Q^i = U+V^i$ when $i=1,\ldots,k$ and $Q^i=U$ when $i=k+1,\ldots,n$, where V^i is a deterministic binary matrix. The matrix V^i is designed to have access to all the symbols of the requested record X^f that are stored in the first k servers. We choose

$$V^1 = \left[\begin{array}{c|c|c} \mathbf{0}_{k\times(f-1)\alpha} & \begin{array}{c} I_{(k-1)\times(k-1)} \\ \mathbf{0}_{1\times(k-1)} \end{array} & \mathbf{0}_{k\times(m-f)\alpha} \end{array}\right]$$

and $V^i, i=2,\ldots,k$ is obtained from the matrix V^{i-1} by a single downward cyclic shift of its row vectors. Then, each server computes and returns the length-k vector $A^i = Q^iC_i^T$, and we write $A^i=(A_1^i,A_2^i,\ldots,A_k^i)^T$.

Theorem 5.7. Construction 4 (PM-MSR-sep) is information-theoretically perfect.

Proof. Decodability: We can see that for $i = 1, \ldots, n$,

$$\begin{aligned} C_i &= \psi_i \cdot \mathcal{M} \\ &= \psi_i \cdot \begin{bmatrix} \text{—} & \mathcal{M}_1 & \text{—} \\ \text{—} & \mathcal{M}_2 & \text{—} \\ & \vdots & \\ \text{—} & \mathcal{M}_{2k-2} & \text{—} \end{bmatrix} = \sum_{h=1}^{2k-2} \psi_{ih} \mathcal{M}_h. \end{aligned}$$

Thus,

$$C_i^T = \sum_{h=1}^{2k-2} \psi_{ih} \mathcal{M}_h^T.$$

Denote e_t to be the length-$m\alpha$ binary unit vector with 1 at the t^{th} position. Consider

first the subquery 1; we obtain

$$
\begin{aligned}
C_{11}^f + \sum_{h=1}^{2k-2} \psi_{1h} I_h^1 &= (U_1 + e_{(f-1)\alpha+1}) C_1^T \quad &= A_1^1, \quad &(1)\\
\sum_{h=1}^{2k-2} \psi_{2h} I_h^1 &= U_1 C_2^T \quad &= A_1^2, \quad &(2)\\
C_{3,k-1}^f + \sum_{h=1}^{2k-2} \psi_{3h} I_h^1 &= (U_1 + e_{(f-1)\alpha+(k-1)}) C_3^T \quad &= A_1^3, \quad &(3)\\
C_{4,k-2}^f + \sum_{h=1}^{2k-2} \psi_{4h} I_h^1 &= (U_1 + e_{(f-1)\alpha+(k-2)}) C_4^T \quad &= A_1^4, \quad &(4)\\
&\vdots \quad &\vdots \quad &\\
C_{k,2}^f + \sum_{h=1}^{2k-2} \psi_{k,h} I_h^1 &= (U_1 + e_{(f-1)\alpha+2}) C_k^T \quad &= A_1^k, \quad &(\mathrm{k})\\
\sum_{h=1}^{2k-2} \psi_{k+1,h} I_h^1 &= U_1 C_{k+1}^T \quad &= A_1^{k+1}, \quad &(\mathrm{k{+}1})\\
&\vdots \quad &\vdots \quad &\\
\sum_{h=1}^{2k-2} \psi_{n,h} I_h^1 &= U_1 C_n^T \quad &= A_1^n, \quad &(\mathrm{n})
\end{aligned}
$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T$, $h = 1, 2, \ldots, 2k-2$, and U_1 is the first row of U. The user can solve for $I_1^1, \ldots, I_{2k-2}^1$ from $(2), (k+1), \ldots, (n)$ as they form the equation

$$
\begin{bmatrix} \text{—} & \psi_2 & \text{—} \\ \text{—} & \psi_{k+1} & \text{—} \\ & \vdots & \\ \text{—} & \psi_n & \text{—} \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ \vdots \\ I_{2k-2}^1 \end{bmatrix} = \begin{bmatrix} A_1^2 \\ A_1^{k+1} \\ \vdots \\ A_1^n \end{bmatrix},
$$

where, since $n = 3k-3$, the left matrix is a $((2k-2) \times (2k-2))$ square submatrix of Ψ which is invertible. Therefore, the user gets $C_{11}^1, C_{3,k-1}^1, C_{4,k-2}^1, \ldots, C_{k,2}^1$, i.e., get

server 1	server 2	server 3	$\cdots$	server $k-1$	server k	server $k+1$	$\cdots$	server n
1	2	3	$\cdots$	$k-1$	k			
2	3	4	$\cdots$	k	1			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$k-2$	$k-1$	k	$\cdots$	$k-4$	$k-3$			
$k-1$	k	1	$\cdots$	$k-3$	$k-2$			

Table 5.12: Retrieval pattern for a $(3k-3, k, 2k-2, k-1, 1, k(k-1))$ MSR code. The $\alpha \times n$ entries correspond to the symbols of X^f stored in the system. The entries labelled by the same number, say d, are privately retrieved by subquery d.

all the symbols with label 1 in Table 5.12. Similarly, from subqueries $h = 2, \ldots, k$, the user obtains all the symbols with label h in Table 5.12. Hence, the user has all the symbols of X^f which are stored in the first k servers. From the property of the regenerating codes, the user can reconstruct X^f as desired.

Privacy: To prove the privacy of the scheme, as we construct the query matrices Q^i via the random matrix U, Q^i is independent from f which implies that this scheme achieves perfect privacy. □

Analysis

The storage overhead is

$$\frac{n(m\alpha)}{mB} = \frac{(3k-3)(k-1)}{k(k-1)} = \frac{3k-3}{k} < 3,$$

and cPoP is equal to

$$\frac{nk}{B} = \frac{(3k-3)k}{k(k-1)} = 3,$$

which are both better than Construction 1 (PM-MSR-mixed), and Construction 2 (PM-MBR-sep) when $k \geq 3$. With different retrieval pattern between Constructions 3 and 4 (PM-MSR-sep), we can see that in this construction, storage overhead is lower than Construction 3 while cPoP is slightly higher.

	k	n	R_{PIR} of Construction 4	C_{MDS} (4.3)	% difference
$m = 3$	3	6	0.3333	0.5714	41.6667
	5	12	0.3333	0.6288	46.9907
	8	21	0.3333	0.6553	49.1308
	10	27	0.3333	0.6633	49.7485
$m = 5$	3	6	0.3333	0.5161	35.4167
	5	12	0.3333	0.5908	43.5748
	8	21	0.3333	0.6241	46.5859
	10	27	0.3333	0.6340	47.4278
$m = 7$	3	6	0.3333	0.5039	33.8542
	5	12	0.3333	0.5846	42.9817
	8	21	0.3333	0.6198	46.2165
	10	27	0.3333	0.6302	47.1094

Table 5.13: The comparison between the retrieval rate of Construction 4 and the capacity of PIR using MDS codes.

Table 5.13 shows the comparison between the retrieval rate of Construction 4 (PM-MSR-sep) and the capacity of PIR using MDS codes for the same parameters m, k, n. Similar to Construction 3 (PM-MSR-sep), the repair ratio is RR $= 2$, and the retrieval rate of Construction 4 (PM-MSR-sep) is closer to the capacity for MDS codes when the number of records m is larger. For example, when $k = 3$, the percentage difference is asymptotically 33.33% as m approaches infinity.

Chapter Summary

To sum up, we apply the technique from Shah's scheme (PM-MBR-mixed) to build Construction 1 (PM-MSR-mixed), which has better SO, but worse cPoP and RR. Afterwards, we construct three PIR schemes under the separate coding architecture. Construction 2 (PM-MBR-sep) has the same SO, cPoP and RR with Shah's scheme (PM-MBR-mixed) with more flexibility that allows the number of records m being independent from the parameter k of the underlying code. Constructions 3 and 4 (PM-MSR-sep) have better SO and cPoP, but worse RR than Construction 2 (PM-MBR-sep). Compared with Construction 1 that uses the same PM-MSR codes,

Constructions 3 and 4 also have better SO and cPoP. Hence, our schemes under the separate coding architecture (Constructions 2, 3, and 4) are more efficient than schemes under the mixed coding architecture (Shah's scheme and Construction 1), so we continue to investigate further on PIR using product-matrix regenerating codes under the separate coding architecture in Chapters 6 and 7.

Chapter 6

Multi-Message PIR using PM-MSR and PM-MBR codes

We discuss a multi-message scenario in this chapter, where a user wants to retrieve more than one record. Trivially, a single-message PIR scheme can be repeatedly operated to retrieve multiple records. However, we wish for a more efficient way. We notice that during the retrieval step of PIR, we obtain some undesired symbols in order to obtain the desired record. Thus, we investigate how we can exploit those undesired symbols to retrieve more records. In Section 6.1, we propose the storage model using product-matrix regenerating codes under the separate coding architecture, and the retrieval scheme of MPIR under this storage model. In Section 6.2, we derive decodability condition of MPIR under this model. We also analyse relationships between SO, cPoP and RR, and obtain a trade-off curve between cPoP and RR. In Section 6.3, we modify Constructions 2, 3, and 4 to be able to retrieve multiple records, and show that these constructions attain the trade-off curve between cPoP and RR. To the best of our knowledge, these are the first MPIR schemes using regenerating codes.

6.1 System Model

In this section, we formally present the storage model and its retrieval scheme of MPIR using product-matrix regenerating codes under the separate coding architecture. Suppose there are n non-communicating servers in the system that store a database X which consists of m records, each of length ℓ, denoted by $X^1, X^2, \ldots, X^m \in \mathbb{F}_q^\ell$. Each record is encoded and distributed across n servers by the same product-matrix regenerating code with parameters $(n, k, r, \alpha, \beta, \ell)$ with the encoding matrix Ψ. Hence,

$$E_{1,n,\alpha}(X^j) = \Psi_{n\times r} \cdot \mathcal{M}^j_{r\times\alpha}$$

where $\mathcal{M}^j$ is the corresponding message matrix of X^j. We denote $E_{1,n,\alpha}(X^j)$ by C^j. Note that the constraint of q here is the constraint of the underlying code. Write

$$\mathcal{M} = \begin{bmatrix} \mathcal{M}^1 & \cdots & \mathcal{M}^m \end{bmatrix},$$

and denote by $\mathcal{M}_i$ the row i of $\mathcal{M}$. Hence, we can see the entire system as

$$E_{m,n,m\alpha}(X^1, \ldots, X^m) = \begin{bmatrix} \Psi \cdot \mathcal{M}^1 & \cdots & \Psi \cdot \mathcal{M}^m \end{bmatrix} = \Psi \cdot \mathcal{M},$$

and each server stores $m\alpha$ symbols in total. We simply write $E_{m,n,m\alpha}(X^1, \ldots, X^m)$ as C. We denote by C_i the row i of C which is all symbols stored in server i, and C_i^j the row i of C^j which is all symbols of X^j stored in server i.

We assume that in the retrieval step the user wants to download p records when $p \leq m$, denoted by $X^{f_1}, \ldots, X^{f_p}$. The user submits a $d \times m\alpha$ query matrix Q^i over $\mathbb{F}_q$ to server i. We can interpret d rows of Q^i as d subqueries, which depend on how we design the query to access desired symbols. Finally, server i computes and responds with an answer $A^i = Q^i C_i^T$. The retrieval steps are as follows:

(i) (Initialisation) The user generates a $d \times m\alpha$ matrix U whose elements are chosen uniformly at random over $\mathbb{F}_q$. Let U_j be row j of U.

(ii) (Query Generation) The query matrix Q^i is defined by $d \times \alpha$ binary matrices

$$V^{i_{f_1}}, \ldots, V^{i_{f_p}},$$

as

$$Q^i = U + V^{i_{f_1}} E^{f_1} + \cdots + V^{i_{f_p}} E^{f_p}$$

where

$$E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha \times (f_u - 1)\alpha} & I_{\alpha \times \alpha} & \mathbf{0}_{\alpha \times (m - f_u)\alpha} \end{array} \right].$$

In other words, E^{f_u} is an $\alpha \times m\alpha$ matrix such that $C_i(E^{f_u})^T = C_i^{f_u}$ which is a coded data piece of a desired record X^{f_u} stored in server i. If the entry (j, b) of $V^{i_{f_u}}$ is 1, then it implies that the entry $C_{ib}^{f_u}$ is privately retrieved by the j^{th} subquery of Q^i.

(iii) (Response Mappings) Each server i returns a $d \times 1$ column vector $A^i = Q^i C_i^T$.

This extends the general framework in [11, 14, 29–31] to retrieve multiple records. An MPIR scheme is said to be *information-theoretically perfect* if

(i) (privacy) $H(f_1, \ldots, f_p | Q^i) = H(f_1, \ldots, f_p)$ for every $i \in [n]$;

(ii) (decodability) $H(X^{f_1}, \ldots, X^{f_p} | A^1, \ldots, A^n) = 0$.

According to our definition, (i) implies that a server i does not obtain any information about which records are being downloaded by the user, and (ii) ensures that the user can recover the desired records $X^{f_1}, \ldots, X^{f_p}$ with no errors from all responses A^i for $i \in [n]$. In this model, three metrics to measure the

efficiency of the scheme can be calculated as follows: first, storage overhead equals

$$\text{SO} = n(m\alpha)/m\ell = n\alpha/\ell,$$

then cPoP is

$$\text{cPoP} = dn/p\ell,$$

and lastly, repair ratio is equal to

$$\text{RR} = mr\beta/m\alpha = r\beta/\alpha.$$

6.2 Decodability Condition and Trade-off Analysis

From the retrieval scheme, the response from server i can be written as a $1 \times d$ row vector

$$\begin{aligned}(A^i)^T &= C_i(Q^i)^T \\ &= C_i[U^T + (E^{f_1})^T(V^{if_1})^T + \cdots + (E^{f_p})^T(V^{if_p})^T] \\ &= C_iU^T + C_i(E^{f_1})^T(V^{if_1})^T + \cdots + C_i(E^{f_p})^T(V^{if_p})^T \\ &= C_iU^T + (C_i^{f_1})(V^{if_1})^T + \cdots + (C_i^{f_p})(V^{if_p})^T.\end{aligned}$$

Then, the j^{th} response in A^i is $A^i_j = C_i(U_j)^T + (C_i^{f_1})(V_j^{if_1})^T + \cdots + (C_i^{f_p})(V_j^{if_p})^T$ where $V_j^{if_u}$ is the row j of $V^{if_u}, u \in [p]$. Hence, records $X^{f_1}, \ldots, X^{f_p}$ should be decoded by solving the system of linear equations

$$A^i_j = C_i(U_j)^T + (C_i^{f_1})(V_j^{if_1})^T + \cdots + (C_i^{f_p})(V_j^{if_p})^T,$$

for all $i \in [n], j \in [d]$ where the unknowns are

$$\left\{C_i(U_j)^T : i \in [n], j \in [d]\right\} \text{ and } \left\{C_{ib}^{f_u} : u \in [p], i \in [n], b \in [\alpha]\right\}.$$

Consider first the set of unknowns $\left\{C_i(U_j)^T, i \in [n], j \in [d]\right\}$, for each $j \in [d]$,

$$\begin{aligned} C(U_j)^T &= \Psi \cdot \mathcal{M} \cdot (U_j)^T \\ &= \Psi \cdot \begin{bmatrix} I_1^j & \cdots & I_r^j \end{bmatrix}^T \end{aligned}$$

where $I_h^j = \mathcal{M}_h \cdot (U_j)^T$, for every row $h \in [r]$ of the matrix $\mathcal{M}$. For the set of unknowns $\left\{C_{ib}^{f_u}, u \in [p], i \in [n], b \in [\alpha]\right\}$, we know that

$$C_{ib}^{f_u} = \text{the entry } (i,b) \text{ of } \Psi \cdot \mathcal{M}^{f_u}, \ \forall u \in [p], i \in [n], b \in [\alpha].$$

Hence, the retrieval scheme is decodable if the following system of linear equations

$$\begin{cases} A_j^i = C_i(U_j)^T + (C_i^{f_1})(V_j^{i_{f_1}})^T + \cdots + (C_i^{f_p})(V_j^{i_{f_p}})^T, & i \in [n], j \in [d] \\ C_i(U_j)^T = \Psi_i \cdot \begin{bmatrix} I_1^j & \cdots & I_r^j \end{bmatrix}^T, & i \in [n], j \in [d] \\ C_{ib}^{f_u} = \text{the entry } (i,b) \text{ of } C^{f_u}, & u \in [p], i \in [n], b \in [\alpha] \end{cases}$$

has a unique solution, where the unknowns are

$$\left\{C_i(U_j)^T : i \in [n], j \in [d]\right\} \text{ and } \left\{C_{ib}^{f_u} : u \in [p], i \in [n], b \in [\alpha]\right\}.$$

This condition is called *decodability condition.*

Next, we analyse the relationship between storage overhead, cPoP, and repair ratio. First, we count the number of unknowns in the system of linear equations in the decodability condition which is equal to $nd + pn\alpha$. Next, we count the number

of linearly independent equations in the system. Consider

$$C_i(U_j)^T = \Psi_i \cdot \begin{bmatrix} I_1^j & \cdots & I_r^j \end{bmatrix}^T, \forall i \in [n], j \in [d],$$

so we have, for each $j \in [d]$,

$$C(U_j)^T = \Psi \cdot \begin{bmatrix} I_1^j & \cdots & I_r^j \end{bmatrix}^T.$$

Since Ψ is of rank r, it has a parity check matrix P of rank $n-r$ such that $P \cdot \Psi = 0$. So we have

$$P \cdot C(U_j)^T = P \cdot \Psi \cdot \begin{bmatrix} I_1^j & \cdots & I_r^j \end{bmatrix}^T = 0.$$

This gives us $n - r$ linearly independent equations for each $j \in [d]$. Then, for

$$C_{ib}^{f_u} = \text{the entry } (i, b) \text{ of } C^{f_u}, \forall u \in [p], i \in [n], b \in [\alpha],$$

since any k rows of C^{f_u} would give us $\mathcal{M}^{f_u}$, the remaining $n - k$ rows must be able to be written in terms of linear combinations of those k rows of C^{f_u}. This give us $(n - k)\alpha$ equations in $C_{ib}^{f_u}$. Hence, there are at most $nd + (n - r)d + p(n - k)\alpha$ linearly independent equations in the system. If the retrieval scheme meets the decodability condition, then

$$nd + pn\alpha \leq nd + (n - r)d + p(n - k)\alpha,$$

which implies that

$$pk\alpha \leq (n - r)d.$$

Therefore,

$$1 \leq \frac{dn}{p\ell} \cdot \frac{\ell}{n\alpha} \cdot \frac{n - r}{k}.$$

In terms of storage overhead and cPoP we have

$$1 \leq \text{cPoP}\left(\frac{n-r}{k(\text{SO})}\right). \tag{6.1}$$

This shows that cPoP is bounded below by storage overhead.

Next, we recall that the trade-off between storage overhead and repair bandwidth of regenerating codes is given by [23] which is

$$\sum_{i=0}^{k-1} \min\{\alpha, (r-i)\beta\} \geq \ell,$$

for the regenerating code with parameters $(n, k, r, \alpha, \beta, \ell)$. So, we have

$$\sum_{i=0}^{k-1} \min\left\{1, \left(1-\frac{i}{r}\right)\frac{r\beta}{\alpha}\right\} \geq \frac{\ell}{\alpha},$$

which can be written in terms of storage overhead and repair ratio as

$$\sum_{i=0}^{k-1} \min\left\{1, \left(1-\frac{i}{r}\right)\text{RR}\right\} \geq \frac{n}{\text{SO}}.$$

From the inequality (6.1), we have

$$\frac{n}{\text{SO}} \geq \left(\frac{nk}{n-r}\right)\frac{1}{\text{cPoP}},$$

therefore, the relationship between repair ratio and cPoP is obtained as

$$\sum_{i=0}^{k-1} \min\left\{1, \left(1-\frac{i}{r}\right)\text{RR}\right\} \geq \left(\frac{nk}{n-r}\right)\frac{1}{\text{cPoP}}, \tag{6.2}$$

which shows that there is a trade-off between cPoP and RR.

6.3 MPIR Schemes

In this section, we generalise Constructions 2, 3, 4 in Section 5.2 to be able to retrieve multiple records. The idea is that, in those constructions, the number of servers n is fixed to be either $k+r$ or $\alpha+r$. Two possible responses of server i for subquery j are

$$A_j^i = \sum_{h=1}^{r} \psi_{i,h} I_h^j$$

and

$$A_j^i = C_{i,a}^f + \sum_{h=1}^{r} \psi_{i,h} I_h^j,$$

for some $a \in [\alpha]$. The subqueries in those constructions are designed to have r responses in the former possibility, so the undesired symbols $I_1^j, \ldots, I_r^j$ can be solved. Then we can obtain $n-r$ desired symbol from the others $n-r$ responses in the latter possibility. Hence, if we allow the number of servers n to be higher than the fixed value $k+r$ or $\alpha+r$, then we will be able to obtain more desired symbols. From this idea, we modify Constructions 2, 3, 4 to new MPIR schemes, and we can prove that these MPIR schemes attain the trade-off curve between cPoP and RR (6.2) with equality.

6.3.1 Construction 5 (PM-MBR-sep)

This construction is the generalisation of Construction 2. We use the PM-MBR code in Section 2.2.2 over the finite field $\mathbb{F}_q$. Here, we assume that the number of servers is

$$n \geq k+r,$$

so the parameters of the PM-MBR code are

$$(n, k, r, \alpha, \beta, \ell) = \left(n, k, r, r, 1, \frac{k(2r-k+1)}{2}\right).$$

The number of records that can be retrieved in this scheme is

$$p \leq \left\lfloor \frac{n-r}{k} \right\rfloor.$$

Hence, a sufficiently large value of $n \geq k+r$ guarantees that this scheme can retrieve at least 1 record. In this scheme, the queries submitted to servers $(u-1)k+1, \ldots, uk$, where $u \in [p]$, are designed to access symbols of the u^{th} desired record that are stored in those k servers, so the u^{th} desired record can be reconstructed by the property of regenerating codes. We first start with an example to motivate our scheme.

Example 6.1. Suppose that we have 3 records over the finite field $\mathbb{F}_{13}$, each with size 5, which can be written as

$$X^j = \{x_{j1}, x_{j2}, x_{j3}, x_{j4}, x_{j5}\}, \text{ for } j = 1, 2, 3.$$

Each record is encoded by a $(7, 2, 3, 3, 1, 5)$ PM-MBR code over $\mathbb{F}_{13}$ where the encoding matrix Ψ is the Vandermonde matrix

$$\Psi = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 3 \\ 1 & 5 & 12 \\ 1 & 6 & 10 \\ 1 & 7 & 10 \end{bmatrix},$$

server 1	server 2	server 3	server 4
$x_{11}+x_{12}+x_{14}$	$x_{11}+2x_{12}+4x_{14}$	$x_{11}+3x_{12}+9x_{14}$	$x_{11}+4x_{12}+3x_{14}$
$x_{12}+x_{13}+x_{15}$	$x_{12}+2x_{13}+4x_{15}$	$x_{12}+3x_{13}+9x_{15}$	$x_{12}+4x_{13}+3x_{15}$
$x_{14}+x_{15}$	$x_{14}+2x_{15}$	$x_{14}+3x_{15}$	$x_{14}+4x_{15}$
$x_{21}+x_{22}+x_{24}$	$x_{21}+2x_{22}+4x_{24}$	$x_{21}+3x_{22}+9x_{24}$	$x_{21}+4x_{22}+3x_{24}$
$x_{22}+x_{23}+x_{25}$	$x_{22}+2x_{23}+4x_{25}$	$x_{22}+3x_{23}+9x_{25}$	$x_{22}+4x_{23}+3x_{25}$
$x_{24}+x_{25}$	$x_{24}+2x_{25}$	$x_{24}+3x_{25}$	$x_{24}+4x_{25}$
$x_{31}+x_{32}+x_{34}$	$x_{31}+2x_{32}+4x_{34}$	$x_{31}+3x_{32}+9x_{34}$	$x_{31}+4x_{32}+3x_{34}$
$x_{32}+x_{33}+x_{35}$	$x_{32}+2x_{33}+4x_{35}$	$x_{32}+3x_{33}+9x_{35}$	$x_{32}+4x_{33}+3x_{35}$
$x_{34}+x_{35}$	$x_{34}+2x_{35}$	$x_{34}+3x_{35}$	$x_{34}+4x_{35}$

server 5	server 6	server 7
$x_{11}+5x_{12}+12x_{14}$	$x_{11}+6x_{12}+10x_{14}$	$x_{11}+7x_{12}+10x_{14}$
$x_{12}+5x_{13}+12x_{15}$	$x_{12}+6x_{13}+10x_{15}$	$x_{12}+7x_{13}+10x_{15}$
$x_{14}+5x_{15}$	$x_{14}+6x_{15}$	$x_{14}+7x_{15}$
$x_{21}+5x_{22}+12x_{24}$	$x_{21}+6x_{22}+10x_{24}$	$x_{21}+7x_{22}+10x_{24}$
$x_{22}+5x_{23}+12x_{25}$	$x_{22}+6x_{23}+10x_{25}$	$x_{22}+7x_{23}+10x_{25}$
$x_{24}+5x_{25}$	$x_{24}+6x_{25}$	$x_{24}+7x_{25}$
$x_{31}+5x_{32}+12x_{34}$	$x_{31}+6x_{32}+10x_{34}$	$x_{31}+7x_{32}+10x_{34}$
$x_{32}+5x_{33}+12x_{35}$	$x_{32}+6x_{33}+10x_{35}$	$x_{32}+7x_{33}+10x_{35}$
$x_{34}+5x_{35}$	$x_{34}+6x_{35}$	$x_{34}+7x_{35}$

Table 6.1: The storage of records X^1, X^2, X^3 in the servers

and the message matrix $\mathcal{M}^j$ for the record $j, j \in \{1,2,3\}$ is as described in Section 2.2.2

$$\mathcal{M}^j = \begin{bmatrix} x_{j1} & x_{j2} & x_{j4} \\ x_{j2} & x_{j3} & x_{j5} \\ x_{j4} & x_{j5} & 0 \end{bmatrix}.$$

Hence, each server stores $\Psi \cdot \mathcal{M}$ as shown in Table 6.1.

Recall that C_{ib}^j is the b^{th} symbol of record j, stored in server i. Here the entire database can be recovered from the content of any 2 servers, and if any one server failed, it can be repaired by downloading one symbol each from 3 of the remaining servers. In the retrieval step, we see that this scheme can retrieve $p \leq \lfloor \frac{7-3}{2} \rfloor = 2$ records. Suppose the user wants record X^1 and X^2. The query Q^i is a (3×9) matrix which we can interpret as 3 subqueries submitted to server i for each $i \in [7]$. To form the query matrices, the user generates a (3×9) random matrix $U = [u_{ij}]$

whose entries are chosen uniformly at random from $\mathbb{F}_{13}$. Recall that V^{i_j} is a matrix which is part of the query submitted to server i, attempting to retrieve information about record X^j. Choose

$$V^{1_1} = V^{2_1} = V^{3_2} = V^{4_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and

$$V^{1_2} = V^{2_2} = V^{3_1} = V^{4_1} = V^{5_1} = V^{5_2} = V^{6_1} = V^{6_2} = V^{7_1} = V^{7_2} = \mathbf{0}_{3\times 3}.$$

Here, for example, the entry $(3,3)$ of V^{4_2} is 1 which implies that the 3^{rd} symbol of X^2 stored in server 4 is accessed by subquery 3 of Q^4. As

$$E^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$E^2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

we have

$$V^{1_1}E^1 = V^{2_1}E^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$V^{3_2}E^2 = V^{4_2}E^2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

and

$$V^{1_2}E^2 = V^{2_2}E^2 = V^{3_1}E^1 = V^{4_1}E^1 =$$

$$V^{5_1}E^1 = V^{5_2}E^2 = V^{6_1}E^1 = V^{6_2}E^2 = V^{7_1}E^1 = V^{7_2}E^2 = \mathbf{0}_{3\times 9}.$$

The query matrices are $Q^i = U + V^{i_1}E^1 + V^{i_2}E^2, i \in [7]$. Then each server computes and returns the length-3 vector $A^i = Q^i C_i^T$. Write $(A^i)^T = (A_1^i, A_2^i, A_3^i)$. Recall that

$$\begin{aligned} \mathcal{M}_1 &= (x_{11}, x_{12}, x_{14}, x_{21}, x_{22}, x_{24}, x_{31}, x_{32}, x_{34}), \\ \mathcal{M}_2 &= (x_{12}, x_{13}, x_{15}, x_{22}, x_{23}, x_{35}, x_{32}, x_{33}, x_{35}), \\ \mathcal{M}_3 &= (x_{14}, x_{15}, 0, x_{24}, x_{25}, 0, x_{34}, x_{35}, 0). \end{aligned}$$

Consider first subquery 1; we obtain

$$C_{11}^1 + I_1^1 + I_2^1 + I_3^1 = A_1^1, \tag{1.1}$$

$$C_{21}^1 + I_1^1 + 2I_2^1 + 4I_3^1 = A_1^2, \tag{1.2}$$

$$C_{34}^2 + I_1^1 + 3I_2^1 + 9I_3^1 = A_1^3, \tag{1.3}$$

$$C_{44}^2 + I_1^1 + 4I_2^1 + 3I_3^1 = A_1^4, \tag{1.4}$$

$$I_1^1 + 5I_2^1 + 12I_3^1 = A_1^5, \tag{1.5}$$

$$I_1^1 + 6I_2^1 + 10I_3^1 = A_1^6, \tag{1.6}$$

$$I_1^1 + 7I_2^1 + 10I_3^1 = A_1^7, \tag{1.7}$$

where $I_h^1 = \mathcal{M}_h \cdot U_1^T, h = 1, 2, 3$, and U_1 is the first row of U. The user can solve

for I_1^1, I_2^1, I_3^1 from (1.5),(1.6),(1.7) as they form the equation

$$\begin{bmatrix} 1 & 5 & 12 \\ 1 & 6 & 10 \\ 1 & 7 & 10 \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ I_3^1 \end{bmatrix} = \begin{bmatrix} A_1^5 \\ A_1^6 \\ A_1^7 \end{bmatrix},$$

where the left matrix is the (3×3) submatrix of Ψ which is invertible by the design of the encoding matrix. Therefore, the user gets C_{11}^1 and C_{31}^1 for record 1 and C_{34}^2 and C_{44}^2 for record 2, which are all the symbols with label 1 in Table 6.2. Similarly, from subquery 2, the user obtains C_{12}^1 and C_{22}^1 for record 1 and C_{35}^2 and C_{45}^2 for record 2. Lastly, from subquery 3, the user obtains C_{13}^1 and C_{23}^1 for record 1 and C_{36}^2 and C_{46}^2 for record 2. Hence, the user has all the symbols of X^1 which are stored in the server $1, 2$ and all the symbols of X^2 which are stored in the server $3, 4$. From the recovery property of regenerating codes, the user can reconstruct X^1 and X^2 as desired.

server 1	server 2	server 3	server 4	server 5	server 6	server 7
1	1					
2	2					
3	3					
		1	1			
		2	2			
		3	3			

Table 6.2: Retrieval pattern for a $(7, 2, 3, 3, 1, 5)$ MBR code. The $m\alpha \times n$ entries correspond to C^T and the entries labelled by the same number, say d, are privately retrieved by subquery d.

Next we formally give the details of this construction.

Encoding Step

Recall that we use the MBR code with parameters

$$(n, k, r, \alpha, \beta, B) = \left(n, k, r, r, 1, \frac{k(2r-k+1)}{2}\right)$$

over $\mathbb{F}_q$ with $n \geq k + r$ to store each record $X^1, \ldots, X^m$, which means that

$$C^j = \Psi \cdot \mathcal{M}^j$$

where $\mathcal{M}^j$ is the message matrix corresponding to X^j as described in Section 2.2.2, so

$$C = \left[\Psi \cdot \mathcal{M}^1 \quad \cdots \quad \Psi \cdot \mathcal{M}^m\right].$$

Retrieval Step

Suppose that the user wants to retrieve $p \leq \lfloor \frac{n-r}{k} \rfloor$ records $X^{f_1}, X^{f_2}, \ldots, X^{f_p}$. In the retrieval step, the user sends an $(\alpha \times m\alpha)$ query matrix Q^i, which we can interpret as α subqueries, to each server $i, i = 1, \ldots, n$. To form the query matrices, the user generates an $(\alpha \times m\alpha)$ random matrix $U = [u_{ij}]$ whose entries are chosen uniformly at a random from $\mathbb{F}_q$. We choose, for $u \in [p], w \in [k]$,

$$V^{(w+(u-1)k)_{f_u}} = I_{\alpha\times\alpha},$$

and for others V^{s_t} which are not defined above, we choose $V^{s_t} = \mathbf{0}_{\alpha\times\alpha}$. As

$$E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha\times(f_u-1)\alpha} & I_{\alpha\times\alpha} & \mathbf{0}_{\alpha\times(m-f_u)\alpha} \end{array}\right],$$

we have

$$V^{(w+(u-1)k)_{f_u}} E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha\times(f_u-1)\alpha} & I_{\alpha\times\alpha} & \mathbf{0}_{\alpha\times(m-f_u)\alpha} \end{array}\right].$$

For the rest, we have $V^{s_t}E^{f_t} = \mathbf{0}_{\alpha \times m\alpha}$. The query matrices are

$$Q^i = U + V^{i_{f_1}}E^{f_1} + \cdots + V^{i_{f_p}}E^{f_p}, i \in [n].$$

Then, each server computes and returns the length-α $A^i = Q^i C_i^T$, and we write

$$(A^i)^T = (A_1^i, A_2^i, \ldots, A_\alpha^i).$$

Theorem 6.2. Construction 5 (PM-MBR-sep) is information-theoretically perfect.

Proof. Decodability: We can see that for $i = 1, \ldots, n$,

$$\begin{aligned} C_i &= \Psi_i \cdot \mathcal{M} \\ &= \Psi_i \cdot \begin{bmatrix} \text{—} & \mathcal{M}_1 & \text{—} \\ \text{—} & \mathcal{M}_2 & \text{—} \\ & \vdots & \\ \text{—} & \mathcal{M}_r & \text{—} \end{bmatrix} \\ &= \sum_{h=1}^{r} \Psi_{ih}\mathcal{M}_h. \end{aligned}$$

Thus,

$$C_i^T = \sum_{h=1}^{r} \Psi_{ih}\mathcal{M}_h^T.$$

Denote e_t to be the length-$m\alpha$ binary unit vector with 1 at the t^{th} position. Consider

first subquery 1; we obtain

$$
\begin{aligned}
C_{1,1}^{f_1} + \sum_{h=1}^{r} \Psi_{1h} I_h^1 &= (U_1 + e_{(f_1-1)\alpha+1}) C_1^T && = A_1^1, \\
&\vdots && \vdots \\
C_{k,1}^{f_1} + \sum_{h=1}^{r} \Psi_{k,h} I_h^1 &= (U_1 + e_{(f_1-1)\alpha+1}) C_k^T && = A_1^k, \\
&\vdots && \vdots \\
C_{(p-1)k+1,1}^{f_p} + \sum_{h=1}^{r} \Psi_{(p-1)k+1,h} I_h^1 &= (U_1 + e_{(f_p-1)\alpha+1}) C_{(p-1)k+1}^T && = A_1^{(p-1)k+1}, \\
&\vdots && \vdots \\
C_{pk,1}^{f_p} + \sum_{h=1}^{r} \Psi_{pk,h} I_h^1 &= (U_1 + e_{(f_p-1)\alpha+1}) C_{pk}^T && = A_1^{pk}, \\
\sum_{h=1}^{r} \Psi_{pk+1,h}, I_h^1 &= U_1 C_{pk+1}^T && = A_1^{pk+1}, \\
&\vdots && \vdots \\
\sum_{h=1}^{r} \Psi_{n,h}, I_h^1 &= U_1 C_n^T && = A_1^n,
\end{aligned}
$$

where $I_h^1 = U_1 \cdot \mathcal{M}_h^T, h = 1, 2, \ldots, r$, and U_1 is the first row of U. Since $p \leq \lfloor \frac{n-r}{k} \rfloor$, we can see that $n - pk \geq r$, which means we can choose any r responses from server $pk+1$ to server n to solve for $I_1^1, \ldots, I_r^1$. Without loss of generality, we choose responses from server $pk+1$ to server $pk+r$ as they form the equation

$$
\begin{bmatrix} \text{—} & \Psi_{pk+1} & \text{—} \\ & \vdots & \\ \text{—} & \Psi_{pk+r} & \text{—} \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ \vdots \\ I_r^1 \end{bmatrix} = \begin{bmatrix} A_1^{pk+1} \\ \vdots \\ A_1^{pk+r} \end{bmatrix},
$$

where the left matrix is an $(r \times r)$ square submatrix of Ψ which is invertible by the construction. Note that here we in fact make use of the repair property of the code, which requires $r \times r$ submatrices to be invertible. Therefore, the user gets

$$C^{f_u}_{(u-1)k+1,1}, \ldots, C^{f_u}_{uk,1},$$

i.e., all the symbols of record f_u with label 1 in Table 6.3 for every $u \in [p]$. Combined with responses from subqueries $2, \ldots, \alpha$, the user has all the symbols of X^{f_u} which are stored in server $(u-1)k+1$ to server uk for all $u \in [p]$. From the recovery property of the regenerating code, the user can finally reconstruct $X^{f_1}, \ldots, X^{f_p}$ as desired.

Privacy: As we construct the query matrices Q^i via the random matrix U, Q^i is independent from $f_1, \ldots, f_p$ which implies that this scheme achieves perfect privacy. □

server 1	$\cdots$	server $(u-1)k$	server $(u-1)k+1$	$\cdots$	server uk	server $uk+1$	$\cdots$	server n
			1	$\cdots$	1			
			2	$\cdots$	2			
			$\vdots$	$\ddots$	$\vdots$			
			α	$\cdots$	α			

Table 6.3: Retrieval pattern for an $\left(n, k, r, r, 1, \frac{k(2r-k+1)}{2}\right)$ MBR code. The $\alpha \times n$ entries correspond to $(C^{f_u})^T$ and the entries labelled by the same number, say d, are privately retrieved by subquery d.

Analysis

We analyse the efficiency of Construction 5 (PM-MBR-sep). As the capacity of MPIR using product-matrix regenerating codes is still unknown, we compare the best case of our schemes with the capacity of the worst case of MPIR using MDS codes just to give us an idea of the gap and what we might expect the capacity

for MPIR using product-matrix regenerating codes to be. We notice that in our scheme, the number of desired records p is independent from the number of all records m. From the capacity of MPIR with MDS coded database (4.6) [35], we see that when $p = \frac{m}{2}$, we have the worst-case capacity of MDS-MPIR which is

$$C_{MDS\text{-}MPIR} = \frac{1}{1 + \frac{k}{n}}. \tag{6.3}$$

Recall that our scheme can retrieve up to $\lfloor \frac{n-r}{k} \rfloor$ records, however, when we retrieve less than $\lfloor \frac{n-r}{k} \rfloor$ records we have more responses that have not been involved in the reconstruction of desired records. We will give an analysis when $p = \lfloor \frac{n-r}{k} \rfloor$ when the scheme has highest efficiency. In this scheme,

$$\text{SO} = \frac{n\alpha}{\ell} = \frac{nr}{\frac{k(2r-k+1)}{2}} = \frac{2nr}{k(2r-k+1)},$$

and

$$\text{cPoP} = \frac{dn}{p\ell} = \frac{rn}{p\frac{k(2r-k+1)}{2}} = \frac{2nr}{pk(2r-k+1)}.$$

We can see that cPoP is decreasing in p. Hence, the download cost is lower when the user retrieves multiple records compared to the repeated use of our scheme when $p = 1$. We notice that the smallest SO and cPoP occur when $r = k$, and since $p = \lfloor \frac{n-k}{k} \rfloor$, we have $p + 1 > \frac{n-k}{k}$, i.e., $n < (p+2)k$. Then

$$\text{SO} = \frac{2n}{k+1} < \frac{2(p+2)k}{k+1} < 2(p+2)$$

and

$$\text{cPoP} = \frac{2n}{p(k+1)} < \frac{2(p+2)k}{p(k+1)} \approx 2,$$

so the retrieval rate of this scheme when $r = k$ and $p = \lfloor \frac{n-k}{k} \rfloor$ is

$$R_{PIR} = \frac{1}{\text{cPoP}} = \frac{p(k+1)}{2n}. \tag{6.4}$$

For a larger p, storage overhead is higher as it is bounded by $2(p+2)$, however, R_{PIR} is approximately 0.5 which gives flexibility to the scheme for achieving this download cost regardless of parameters of PM-MBR codes used in the system.

Table 6.4 shows the comparison between the retrieval rate of Construction 5 (PM-MBR-sep) when $r = k$ (equation (6.4)) with $p = \frac{n-r}{k}$ and the capacity of MPIR using MDS codes with $p = \frac{m}{2}$ (equation (6.3)) for the same parameters n, k. We can see that our retrieval rate is not as good as capacity of MDS-MPIR. However, the advantage of using MBR codes is that the repair ratio is

$$\frac{r\beta}{\alpha} = 1,$$

which is best possible, while the MDS schemes give a repair ratio of $k > 1$. Note

	k	n	R_{PIR} of Construction 5 (6.4)	$C_{MDS\text{-}MPIR}$ (6.3)	% difference
$p = 2$	3	9	0.4444	0.75	40.7407
	5	15	0.4	0.75	46.6667
	8	24	0.375	0.75	50
	10	30	0.3667	0.75	51.1111
$p = 4$	3	15	0.5333	0.8333	36
	5	25	0.48	0.8333	42.4
	8	40	0.45	0.8333	46
	10	50	0.44	0.8333	47.2
$p = 7$	3	24	0.5833	0.8889	34.375
	5	40	0.525	0.8889	40.9375
	8	64	0.4922	0.8889	44.6289
	10	80	0.4813	0.8889	45.8594

Table 6.4: The comparison between the retrieval rate of Construction 5 when $r = k$ with $p = \frac{n-r}{k}$ and the capacity of MPIR using MDS codes with $p = \frac{m}{2}$.

that when $\mathrm{RR} = 1$, we have

$$\sum_{i=0}^{k-1} \min\left\{1, \left(1 - \frac{i}{r}\right)\mathrm{RR}\right\} = \sum_{i=0}^{k-1}\left(1 - \frac{i}{r}\right) = k - \frac{k(k-1)}{2r} = \frac{k(2r-k+1)}{2r},$$

and if $p = \frac{n-r}{k} \in \mathbb{N}$, then

$$\left(\frac{nk}{n-r}\right)\frac{1}{\mathrm{cPoP}} = \frac{n}{p} \cdot \frac{pk(2r-k+1)}{2nr} = \frac{k(2r-k+1)}{2r},$$

which implies that this scheme lies on the trade-off curve between cPoP and RR (the inequality (6.2)) derived in Section 6.2 when $k|(n-r)$ and $p = \frac{n-r}{k}$. Note that if $p < \frac{n-r}{k}$ then we download more than we need in the retrieval step, therefore the relationships (6.1) and (6.2) are strict inequalities.

6.3.2 Construction 6 (PM-MSR-sep)

This construction is the generalisation of Construction 3. We use the PM-MSR codes in Section 2.2.3 over $\mathbb{F}_q$ to store each record. Here, we assume that the number of servers is

$$n \geq k + r = 3k - 2.$$

This scheme can retrieve up to

$$p \leq \lfloor \frac{n-r}{k} \rfloor = \lfloor \frac{n-2k+2}{k} \rfloor$$

records, and the sufficient condition that $n \geq 3k-2$ ensures that this scheme can retrieve at least 1 record. In this scheme, the queries submitted to servers $(u-1)k+1, \ldots, uk$, where $u \in [p]$, are designed to access symbols of the u^{th} desired record that are stored in those k servers, so the u^{th} desired record can be reconstructed by the property of regenerating codes.

Encoding Step

We construct an MPIR scheme using a PM-MSR code with parameters

$$(n, k, r, \alpha, \beta, \ell) = (n, k, 2k-2, k-1, 1, k(k-1))$$

over $\mathbb{F}_q$ with $n \geq 3k-2$ to store each record $X^1, \ldots, X^m$, i.e., $C^j = \Psi \cdot \mathcal{M}^j$ where $\mathcal{M}^j$ is the message matrix corresponding to X^j as described in Section 2.2.3, so

$$C = \begin{bmatrix} \Psi \cdot \mathcal{M}^1 & \cdots & \Psi \cdot \mathcal{M}^m \end{bmatrix}.$$

Retrieval Step

Suppose that the user wants to retrieve $p \leq \lfloor \frac{n-2k+2}{k} \rfloor$ records

$$X^{f_1}, X^{f_2}, \ldots, X^{f_p}.$$

In the retrieval step, the user sends an $(\alpha \times m\alpha)$ query matrix Q^i, which we can interpret as α subqueries, to each server $i, i = 1, \ldots, n$. To form the query matrices, the user generates an $(\alpha \times m\alpha)$ random matrix $U = [u_{ij}]$ whose entries are chosen uniformly at a random from $\mathbb{F}_q$. We choose, for $u \in [p], w \in [k]$,

$$V^{(w+(u-1)k)_{f_u}} = I_{\alpha \times \alpha},$$

and for others V^{s_t} which are not defined above, we choose $V^{s_t} = \mathbf{0}_{\alpha \times \alpha}$. As

$$E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha \times (f_u - 1)\alpha} & I_{\alpha \times \alpha} & \mathbf{0}_{\alpha \times (m - f_u)\alpha} \end{array} \right],$$

we have

$$V^{(w+(u-1)k)_{f_u}} E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha \times (f_u - 1)\alpha} & I_{\alpha \times \alpha} & \mathbf{0}_{\alpha \times (m - f_u)\alpha} \end{array} \right].$$

For the rest, we have $V^{s_t}E^{f_t} = \mathbf{0}_{\alpha \times m\alpha}$. The query matrices are

$$Q^i = U + V^{i_{f_1}}E^{f_1} + \cdots + V^{i_{f_p}}E^{f_p}, i \in [n].$$

Then, each server computes and returns the length-α $A^i = Q^i C_i^T$, and we write

$$(A^i)^T = (A_1^i, A_2^i, \ldots, A_\alpha^i).$$

Note that the retrieval pattern in this construction is similar to Construction 5 (PM-MBR-sep), hence the proof of decodability and privacy is omitted here.

Analysis

Similar to the analysis of Construction 5 (PM-MBR-sep), here we only analyse this construction for the case $p = \lfloor \frac{n-2k+2}{k} \rfloor$. When $p = \lfloor \frac{n-2k+2}{k} \rfloor$, we have $p+1 > \frac{n-2k+2}{k}$, that is, $n < pk + 3k - 2$. Hence, storage overhead is

$$\text{SO} = \frac{n\alpha}{\ell} = \frac{n(k-1)}{k(k-1)} = \frac{n}{k} < p + 3 - \frac{2}{k} < p + 3,$$

and

$$\text{cPoP} = \frac{dn}{p\ell} = \frac{(k-1)n}{pk(k-1)} = \frac{n}{pk} < 1 + \frac{3k-2}{pk},$$

so the retrieval rate is

$$R_{PIR} = \frac{pk}{n}. \tag{6.5}$$

The cPoP here is also decreasing in p for fixed k. Therefore, the download cost when the user retrieves multiple records is lower compared to the repeated use of our scheme when $p = 1$.

Table 6.5 shows the comparison between the retrieval rate of Construction 6 (PM-MSR-sep) (equation (6.5)) with $p = \frac{n-2k+2}{k}$ and the capacity of MPIR using MDS codes with $p = \frac{m}{2}$ (equation (6.3)) for the same parameters n, k. We can see that for higher p, retrieval rate in this scheme is getting closer to capacity of MDS-MPIR. Note that repair ratio in this scheme is

$$\frac{r\beta}{\alpha} = \frac{(2k-2)}{k-1} = 2,$$

which is higher than Construction 5 (PM-MBR-sep). However it is still smaller than PIR schemes that use $[n, k]$ MDS codes (for example, in [35]) where repair ratio is equal to k if $k > 2$. Since RR $= 2$, we have

$$\sum_{i=0}^{k-1} \min\left\{1, \left(1 - \frac{i}{r}\right)\text{RR}\right\} = \sum_{i=0}^{k-1} 1 = k$$

as $r = 2k - 2$ implying that $1 - \frac{i}{r} \geq \frac{1}{2}$ for all $i = 0, \ldots, k-1$. Interestingly, if

	k	n	R_{PIR} of Construction 6 (6.5)	$C_{MDS\text{-}MPIR}$ (6.3)	% difference
$p = 2$	3	10	0.6	0.7692	22
	5	18	0.5556	0.7826	29.0123
	8	30	0.5333	0.7895	32.4444
	10	38	0.5263	0.7917	33.518
$p = 4$	3	16	0.75	0.8421	10.9375
	5	28	0.7143	0.8485	15.8163
	8	46	0.6957	0.8519	18.3365
	10	58	0.6897	0.8529	19.1439
$p = 7$	3	25	0.84	0.8929	5.92
	5	43	0.814	0.8958	9.1401
	8	70	0.8	0.8974	10.8571
	10	88	0.7955	0.898	11.4153

Table 6.5: The comparison between the retrieval rate of Construction 6 with $p = \frac{n-2k+2}{k}$ and the capacity of MPIR using MDS codes with $p = \frac{m}{2}$.

$p = \frac{n-r}{k} \in \mathbb{N}$, then

$$\left(\frac{nk}{n-r}\right)\frac{1}{\text{cPoP}} = \frac{n}{p} \cdot \frac{pk}{n} = k,$$

which implies that this scheme lies on the trade-off curve between cPoP and RR (the inequality (6.2)) derived in Section 6.2 when $k|(n-r)$ and $p = \frac{n-r}{k}$. Notice that when $p = \frac{n-r}{k} \in \mathbb{N}$ the retrieval rate in this scheme is actually

$$\frac{pk}{n} = \frac{1}{1+\frac{r}{n-r}}$$

which is the MDS-MPIR capacity of an $[n-r, r]$ MDS code. We lastly see that cPoP here is lower than our MBR construction while repair ratio is higher which is as expected by the trade-off (6.2).

6.3.3 Construction 7 (PM-MSR-sep)

This construction is the generalisation of Construction 4. We use the PM-MSR codes in Section 2.2.3 over the finite field $\mathbb{F}_q$ to store each record. Here, we assume that the number of servers is

$$n \geq r + \alpha = 3k - 3.$$

This scheme can retrieve up to

$$p \leq \left\lfloor \frac{n-2k+2}{k-1} \right\rfloor$$

records when $n \leq (2k-2)k$, or

$$p \leq \left\lfloor \frac{n}{k} \right\rfloor$$

records when $n > (2k-2)k$. The sufficient condition that $n \geq 3k-3$ ensures that this scheme can retrieve at least 1 record. In this scheme, the queries submitted to

servers $(u-1)k+1, \ldots, uk$, where $u \in [p]$, are designed to access symbols of the u^{th} desired record that are stored in those k servers, so the u^{th} desired record can be reconstructed by the property of regenerating codes. However, the retrieval pattern in this construction is different from Constructions 5 and 6, resulting in lower SO and higher cPoP.

Encoding Step

In this construction, we use the PM-MSR code in Section 2.2.3 over the finite field $\mathbb{F}_q$ with parameters

$$(n, k, r, \alpha, \beta, \ell) = (n, k, 2k-2, k-1, 1, k(k-1)),$$

with $n \geq 3k-3$ to store each record $X^1, \ldots, X^m$, which means that

$$C^i = \Psi \cdot \mathcal{M}^i$$

where $\mathcal{M}^i$ is the message matrix corresponding to X^i as described in Section 2.2.3, so

$$C = \begin{bmatrix} \Psi \cdot \mathcal{M}^1 & \cdots & \Psi \cdot \mathcal{M}^m \end{bmatrix}.$$

Retrieval Step

Suppose that the user wants to retrieve $p \leq \left\lfloor \frac{n-2k+2}{k-1} \right\rfloor$ records when $n \leq (2k-2)k$, or $p \leq \left\lfloor \frac{n}{k} \right\rfloor$ records when $n > (2k-2)k$. Note that it can be verified in both cases that $n \geq pk$. We say that the desired records are $X^{f_1}, X^{f_2}, \ldots, X^{f_p}$. In the retrieval step, the user sends a $(k \times m\alpha)$ query matrix Q^i, which we can interpret as k subqueries, to each server $i, i \in [n]$. To form the query matrices, the user generates a $(k \times m\alpha)$ random matrix $U = [u_{ij}]$ whose entries are chosen uniformly

at a random from $\mathbb{F}_q$. We choose, for $u \in [p]$,

$$V^{(1+(u-1)k)_{f_u}} = \begin{bmatrix} I_{(k-1)\times(k-1)} \\ \mathbf{0}_{1\times(k-1)} \end{bmatrix}$$

and $V^{(w+(u-1)k)_{f_u}}, w = 2, \ldots, k$ is obtained from the matrix $V^{((w-1)+(u-1)k)_{f_u}}$ by a single downward cyclic shift of its row vectors. For any V^{s_t} which is not defined above, we choose $V^{s_t} = \mathbf{0}_{k\times(k-1)}$. As

$$E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{\alpha\times(f_u-1)\alpha} & I_{\alpha\times\alpha} & \mathbf{0}_{\alpha\times(m-f_u)\alpha} \end{array} \right],$$

we have

$$V^{(1+(u-1)k)_{f_u}} E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{k\times(f_u-1)\alpha} & \begin{matrix} I_{(k-1)\times(k-1)} \\ \mathbf{0}_{1\times(k-1)} \end{matrix} & \mathbf{0}_{k\times(m-f_u)\alpha} \end{array} \right]$$

and $V^{(w+(u-1)k)_{f_u}} E^{f_u}, w = 2, \ldots, k$ is obtained from the matrix $V^{((w-1)+(u-1)k)_{f_u}} E^{f_u}$ by a single downward cyclic shift of its row vectors. For the rest, we have

$$V^{s_t} E^{f_t} = \mathbf{0}_{k\times m\alpha}.$$

The query matrices are

$$Q^i = U + V^{i_{f_1}} E^{f_1} + \cdots + V^{i_{f_p}} E^{f_p}, i \in [n].$$

Then, each server computes and returns the length-k $A^i = Q^i C_i^T$, and we write

$$(A^i)^T = (A_1^i, A_2^i, \ldots, A_k^i).$$

Theorem 6.3. Construction 7 (PM-MSR-sep) is information-theoretically perfect.

Proof. Decodability: The following proof will show the decodability of this scheme. We can see that for $i = 1, \ldots, n$,

$$C_i = \Psi_i \cdot \mathcal{M}$$
$$= \Psi_i \cdot \begin{bmatrix} \text{—} & \mathcal{M}_1 & \text{—} \\ \text{—} & \mathcal{M}_2 & \text{—} \\ & \vdots & \\ \text{—} & \mathcal{M}_{2k-2} & \text{—} \end{bmatrix} = \sum_{h=1}^{2k-2} \Psi_{ih} \mathcal{M}_h.$$

Thus, $C_i^T = \sum_{h=1}^{2k-2} \Psi_{ih} \mathcal{M}_h^T$. Consider first subquery 1, we obtain

$$
\begin{aligned}
C_{1,1}^{f_1} + \sum_{h=1}^{2k-2} \Psi_{1h} I_h^1 &= (U_1 + e_{(f_1-1)\alpha+1}) C_1^T && = A_1^1, \\
\sum_{h=1}^{2k-2} \Psi_{2h} I_h^1 &= U_1 C_2^T && = A_1^2, \\
C_{3,k-1}^{f_1} + \sum_{h=1}^{2k-2} \Psi_{3h} I_h^1 &= (U_1 + e_{(f_1-1)\alpha+k-1}) C_3^T && = A_1^3, \\
C_{4,k-2}^{f_1} + \sum_{h=1}^{2k-2} \Psi_{4h} I_h^1 &= (U_1 + e_{(f_1-1)\alpha+k-2}) C_4^T && = A_1^4, \\
&\vdots && \vdots \\
C_{k,2}^{f_1} + \sum_{h=1}^{2k-2} \Psi_{k,h} I_h^1 &= (U_1 + e_{(f_1-1)\alpha+2}) C_k^T && = A_1^k, \\
&\vdots && \vdots \\
C_{(p-1)k+1,1}^{f_p} + \sum_{h=1}^{2k-2} \Psi_{(p-1)k+1,h} I_h^1 &= (U_1 + e_{(f_p-1)\alpha+1}) C_{(p-1)k+1}^T && = A_1^{(p-1)k+1}, \\
\sum_{h=1}^{2k-2} \Psi_{(p-1)k+2,h} I_h^1 &= U_1 C_{(p-1)k+2}^T && = A_1^{(p-1)k+2}, \\
C_{(p-1)k+3,k-1}^{f_p} + \sum_{h=1}^{2k-2} \Psi_{(p-1)k+3,h} I_h^1 &= (U_1 + e_{(f_p-1)\alpha+k-1}) C_{(p-1)k+3}^T && = A_1^{(p-1)k+3}, \\
C_{(p-1)k+4,k-2}^{f_p} + \sum_{h=1}^{2k-2} \Psi_{(p-1)k+4,h} I_h^1 &= (U_1 + e_{(f_p-1)\alpha+k-2}) C_{(p-1)k+4}^T && = A_1^{(p-1)k+4}, \\
&\vdots && \vdots \\
C_{pk,2}^{f_p} + \sum_{h=1}^{2k-2} \Psi_{pk,h} I_h^1 &= (U_1 + e_{(f_p-1)\alpha+2}) C_{pk}^T && = A_1^{pk}, \\
\sum_{h=1}^{2k-2} \Psi_{pk+1,h} I_h^1 &= U_1 C_{pk+1}^T && = A_1^{pk+1}, \\
&\vdots && \vdots \\
\sum_{h=1}^{2k-2} \Psi_{n,h}, I_h^1 &= U_1 C_n^T && = A_1^n,
\end{aligned}
$$

where $I_h^1 = U_1 \cdot \mathcal{M}_h^T, h = 1, 2, \ldots, 2k-2$, and U_1 is the first row of U.

It can be seen that for the first pk servers, we obtain p linearly independent equations from servers $2, k+2, \ldots, (p-1)k+2$ to use for getting rid of the interferences $I_1^1, \ldots, I_{2k-2}^1$.

In case $n \leq (2k-2)k$, since $p \leq \lfloor \frac{n-2k+2}{k-1} \rfloor \leq 2k-2$, we have $0 < (2k-2) - p \leq n - pk$ which means that, apart from p equations we get from the first pk servers, we are able to choose any $(2k-2) - p$ responses from servers $pk+1$ to n to solve for $I_1^1, \ldots, I_{2k-2}^1$. Without loss of generality, we use responses from servers $2, k+2, \ldots, (p-1)k+2$ and $pk+1, \ldots, pk+(2k-2)-p$, which form the equation

$$\begin{bmatrix} \text{—— } \Psi_2 \text{ ——} \\ \text{—— } \Psi_{k+2} \text{ ——} \\ \vdots \\ \text{—— } \Psi_{(p-1)k+2} \text{ ——} \\ \text{—— } \Psi_{pk+1} \text{ ——} \\ \vdots \\ \text{—— } \Psi_{(pk+(2k-2)-p)} \text{ ——} \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ \vdots \\ I_{2k-2}^1 \end{bmatrix} = \begin{bmatrix} A_{21} \\ A_{k+2,1} \\ \vdots \\ A_{(p-1)k+2} \\ A_{pk+1,1} \\ \vdots \\ A_{(pk+(2k-2)-p),1} \end{bmatrix},$$

where the left matrix is a $((2k-2) \times (2k-2))$ square submatrix of Ψ which is invertible by the construction.

In case $n > (2k-2)k$, we have two possibilities: $p \geq 2k-2$ or $p < 2k-2$. If $p \geq 2k-2$, we have enough linearly independent equations from the first pk servers to get rid of the interference $I_1^1, \ldots, I_{2k-2}^1$, that is, we can choose any $2k-2$ responses from servers $2, k+2, \ldots, (p-1)k+2$ to solve for $I_1^1, \ldots, I_{2k-2}^1$. Without loss of generality, we use responses from servers $2, k+2, \ldots, (2k-3)k+2$ which

form the equation

$$\begin{bmatrix} \text{—— } \Psi_2 \text{ ——} \\ \text{—— } \Psi_{k+2} \text{ ——} \\ \vdots \\ \text{—— } \Psi_{(2k-3)k+2} \text{ ——} \end{bmatrix} \cdot \begin{bmatrix} I_1^1 \\ I_2^1 \\ \vdots \\ I_{2k-2}^1 \end{bmatrix} = \begin{bmatrix} A_{21} \\ A_{k+2,1} \\ \vdots \\ A_{(2k-3)k+2,1} \end{bmatrix},$$

where the left matrix is a $((2k-2) \times (2k-2))$ square submatrix of Ψ which is invertible by the construction. However, if $p < 2k-2$, then it is similar to the case $n \leq (2k-2)k$ since we can show that $0 < (2k-2) - p \leq n - pk$. Indeed, when $p = 2k-3$, $n - pk \geq k \geq 1 = (2k-2) - p$, and when $p < 2k-3$, $n - pk \geq 2k > (2k-2) - p$. Therefore, we also have enough linearly independent equations for getting rid of the interferences $I_1^1, \ldots, I_{2k-2}^1$.

Note that here we make use of the repair property of the code, which requires any $r \times r$ submatrices to be invertible. After getting rid of interference, the user obtains

$$C^{f_u}_{(u-1)k+1,1}, C^{f_u}_{(u-1)k+3,k-1}, C^{f_u}_{(u-1)k+4,k-2}, \ldots, C^{f_u}_{uk,2},$$

i.e., all the symbols of record f_u with label 1 in Table 6.6 for all $u \in [p]$. Combined with responses from subqueries $2, \ldots, k$, the user has all the symbols of X^{f_u} which are stored in the server $(u-1)k+1$ to server uk for all $u \in [p]$. From the recovery property of the regenerating code, the user can finally reconstruct $X^{f_1}, \ldots, X^{f_p}$ as desired.

server 1	$\cdots$	server $(u-1)k$	server $(u-1)k+1$	server $(u-1)k+2$	server $(u-1)k+3$	$\cdots$	server $uk-1$	server uk	server $uk+1$	$\cdots$	server n
			1	2	3	$\cdots$	$k-1$	k			
			2	3	4	$\cdots$	k	1			
			$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
			$k-2$	$k-1$	k	$\cdots$	$k-4$	$k-3$			
			$k-1$	k	1	$\cdots$	$k-3$	$k-2$			

Table 6.6: Retrieval pattern for an $(n, k, 2k-2, k-1, 1, k(k-1))$ MSR code. The $\alpha \times n$ entries correspond to $(C^{f_u})^T$ and the entries labelled by the same number, say d, are privately retrieved by subquery d.

Privacy: As we construct the query matrices Q^i via the random matrix U, Q^i is independent from $f_1, \ldots, f_p$. So this scheme achieves perfect privacy.

□

Analysis

Here we only analyse the scheme with highest efficiency when $p = \lfloor \frac{n-2k+2}{k-1} \rfloor$, so we have $p+1 > \frac{n-2k+2}{k-1}$, hence

$$n < (p+3)(k-1).$$

In this scheme, storage overhead is

$$\text{SO} = \frac{n\alpha}{\ell} = \frac{n(k-1)}{k(k-1)} = \frac{n}{k} < \frac{(p+3)(k-1)}{k} < p+3,$$

and

$$\text{cPoP} = \frac{dn}{p\ell} = \frac{kn}{pk(k-1)} = \frac{n}{p(k-1)} < \frac{(p+3)(k-1)}{p(k-1)} = 1 + \frac{3}{p},$$

so retrieval rate in this scheme is

$$R_{PIR} = \frac{p(k-1)}{n}. \tag{6.6}$$

Table 6.7 indicates the comparison between the retrieval rate of Construction 7 (PM-MSR-sep) (equation (6.6)) with $p = \frac{n-r}{k-1}$ and the capacity of MPIR using an MDS code with $p = \frac{m}{2}$ (equation (6.3)) for the same parameters n, k. Similar to Construction 6 (PM-MSR-sep), the retrieval rate here is getting closer to capacity of MDS-MPIR when p is higher. The different is the retrieval rate and cPoP here only depend on the parameter p when $n < (2k-2)k$ and $p = \frac{n-r}{k-1} \in \mathbb{N}$. The download cost when the user retrieves multiple records here is lower compared to the repeated use of our scheme when $p = 1$ as cPoP in this scheme is decreasing in

	k	n	R_{PIR} of Construction 7 (6.6)	$C_{MDS\text{-}MPIR}$ (6.3)	% difference
$p=2$	3	8	0.5	0.7273	31.25
	5	16	0.5	0.7619	34.375
	8	28	0.5	0.7778	35.7143
	10	36	0.5	0.7826	36.1111
$p=4$	3	12	0.6667	0.8	16.6667
	5	24	0.6667	0.8276	19.4444
	8	42	0.6667	0.84	20.6349
	10	54	0.6667	0.8438	20.9877
$p=7$	3	18	0.7778	0.8571	9.2593
	5	36	0.7778	0.878	11.4198
	8	63	0.7778	0.8873	12.3457
	10	81	0.7778	0.8901	12.62

Table 6.7: The comparison between the retrieval rate of Construction 7 with $p = \frac{n-r}{k-1}$ and the capacity of MPIR using an MDS code with $p = \frac{m}{2}$.

p for fixed k. Since RR $= 2$, we have

$$\sum_{i=0}^{k-1} \min\left\{1, \left(1-\frac{i}{r}\right)\mathrm{RR}\right\} = \sum_{i=0}^{k-1} 1 = k$$

as $r = 2k-2$ implying that $1-\frac{i}{r} \geq \frac{1}{2}$ for all $i = 0, \ldots, k-1$. In case $n < (2k-2)k$, if $p = \frac{n-r}{k-1} \in \mathbb{N}$, then

$$\left(\frac{nk}{n-r}\right)\frac{1}{\mathrm{cPoP}} = \frac{nk}{p(k-1)} \cdot \frac{p(k-1)}{n} = k,$$

which implies that this scheme lies on the trade-off curve between cPoP and RR (the inequality (6.2)) derived in Section 6.2 when $(k-1)|(n-r)$ and $p = \frac{n-r}{k-1}$.

Chapter Summary

In this chapter, we consider MPIR schemes using product-matrix regenerating codes. We propose the system model for MPIR and discover a trade-off between cPoP and RR under this model. Then, we generalise our single-message PIR schemes in Section 5.2 to new MPIR schemes, which attain the trade-off between cPoP and RR for some parameters. The cPoP of Construction 5 (PM-MBR-sep) is approximately 2 with the smallest possible RR $= 1$. The cPoP of Constructions 6 and 7 (PM-MSR-sep) are lower than Construction 5 (PM-MBR-sep) with the higher RR $= 2$. This is as expected by the trade-off curve between cPoP and RR.

Chapter 7

An Averaging Technique

An averaging technique is invented by Blackburn et al. [20] for replication-based PIR. In this chapter, we contribute an application of the averaging technique to code-based PIR. The idea behind the technique is that when a query submitted to a server is the sum between a random vector and a deterministic vector designed to access desired symbols stored in that server, there is a possibility that the query is all-zero vector, so the server do not need to reply in this case. Usually when records are long, we can divide each record to be many chunks and each chunk is processed independently and identically by the same PIR scheme. In the averaging technique, instead of using a similar random vector in the query generation, we vary randomness through all possibilities of the random vector to generate queries for each chunk. Thus, there must be queries which contain some all-zero rows where the server does not need to respond, so the retrieval rate is improved. In Section 7.1, we apply the averaging technique to Construction 5. Furthermore, we discuss how to apply the averaging technique to existing code-based PIR schemes in general, and calculate the improvement factor for the retrieval rate. In Section 7.2, we construct a new PIR scheme using $[2k, k]$ MDS codes, which, after applying the averaging technique, achieves the highest improvement factor.

7.1 An Application on Code-Based PIR

First, we apply the averaging technique to Construction 5 (PM-MBR-sep) in order to improve their retrieval rate. Notice that queries in our schemes are matrices where each row in the matrix can be interpreted as a subquery. When a row in the matrix is all-zero, a server do not need to reply to that subquery.

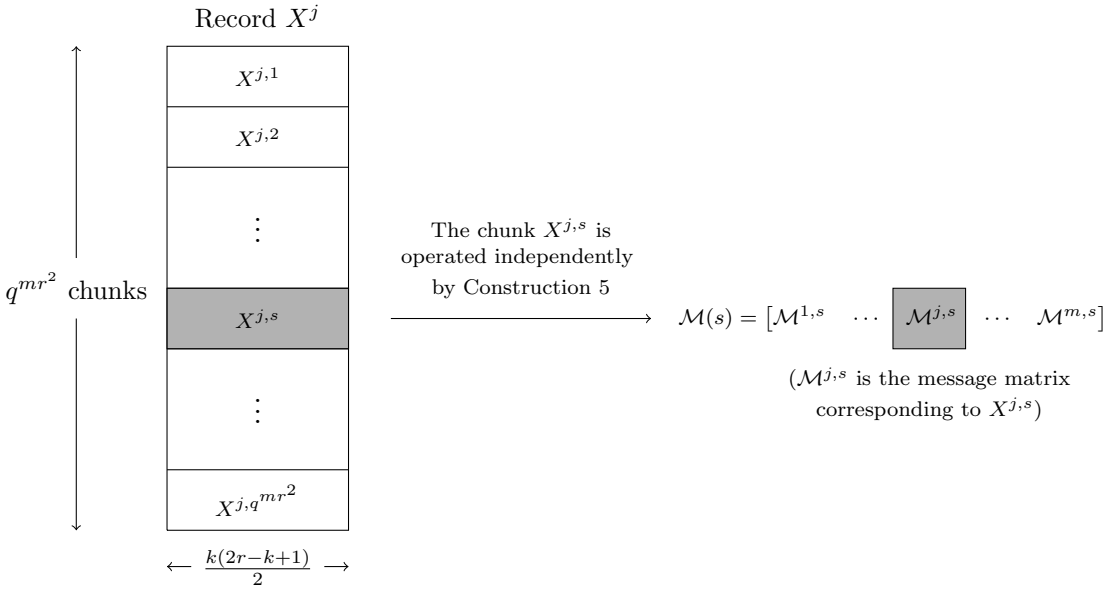

Figure 7.1: Coding process for record X^j

7.1.1 Construction 8 (PM-MBR-sep)

We suppose that each record X^j is of length $q^{mr^2}\left(\frac{k(2r-k+1)}{2}\right)$ over $\mathbb{F}_q$ for all $j \in [m]$. Each record is then divided into q^{mr^2} chunks of length $\frac{k(2r-k+1)}{2}$. Each chunk is operated independently and identically using Construction 5 (PM-MBR-sep). The technique is to vary randomness that is used for query generation through all possible q^{mr^2} random matrices for each chunk. Hence, for each server there must be queries which contain some all-zero rows, so the server does not need to reply those subqueries, resulting in good download complexity in the worst case.

Note that queries for each chunk can be calculated from just one query which means upload complexity here is still low. Each chunk of X^j is indexed by $X^{j,s}$ for $j \in [m], s \in \mathbb{M}_{r\times mr}(\mathbb{F}_q)$, and encoded as

$$E_{1,n,r}(X^{j,s}) = \Psi \cdot \mathcal{M}^{j,s},$$

where $\mathcal{M}^{j,s}$ is the message matrix corresponding to $X^{j,s}$ as described in Section 2.2.2. We denote $E_{1,n,r}(X^{j,s})$ by $C^{j,s}$. Write

$$\mathcal{M}(s) = \begin{bmatrix} \mathcal{M}^{1,s} & \cdots & \mathcal{M}^{m,s} \end{bmatrix},$$

and denote by $\mathcal{M}(s)_i$ the row i of $\mathcal{M}(s)$. Hence, we can see the entire system as

$$E_{m,n,mr}(X^{1,s}, \ldots, X^{m,s}) = \begin{bmatrix} \Psi \cdot \mathcal{M}^{1,s} & \cdots & \Psi \cdot \mathcal{M}^{m,s} \end{bmatrix} = \Psi \cdot \mathcal{M}(s),$$

We simply write $E_{m,n,mr}(X^{1,s}, \ldots, X^{m,s})$ as $C(s)$. We denote by $C(s)_i$ the row i of $C(s)$ which is all symbols of the s^{th} chunk of the database X stored in server i, $C(s)_i^j$ the row i of $C^{j,s}$ which is all symbols of $X^{j,s}$ stored in server i, and $C(s)_{i,a}^j$ the a^{th} symbol of $X^{j,s}$ that stored in server i. Note that each server stores $q^{mr^2}mr$ symbols in total. Suppose that the user wants to retrieve $p \leq \lfloor \frac{n-r}{k} \rfloor$ records

$$X^{f_1}, X^{f_2}, \ldots, X^{f_p}.$$

The retrieval steps are as follows:

(i) (Initialisation) The user generates a random matrix U of dimensions $r \times mr$ whose elements are chosen independently and uniformly at random over $\mathbb{F}_q$.

(ii) (Query Generation) The query matrix Q^i is defined as

$$Q^i = U + V^{i_{f_1}} E^{f_1} + \cdots + V^{i_{f_p}} E^{f_p},$$

where

$$E^{f_u} = \left[\begin{array}{c|c|c} \mathbf{0}_{r\times(f_u-1)r} & I_{r\times r} & \mathbf{0}_{r\times(m-f_u)r} \end{array} \right]$$

and

$$V^{i_{f_1}}, \ldots, V^{i_{f_p}},$$

are $r \times r$ deterministic binary matrices defined as follows: for $u \in [p], w \in [k]$,

$$V^{(w+(u-1)k)_{f_u}} = I_{r\times r},$$

and for others V^{s_t} which are not defined above, we choose $V^{s_t} = \mathbf{0}_{r\times r}$.

(iii) (Response Mappings) For $i \in [n]$, server i returns a symbol

$$A^i(s) = C(s)_i (Q^i + s)^T$$

for every $s \in \mathbb{M}_{r\times mr}(\mathbb{F}_q)$.

Theorem 7.1. Construction 8 (PM-MBR-sep) is information-theoretically perfect with retrieval rate $\left(\frac{q^{mr}}{q^{mr}-1}\right)\left(\frac{pk(2r-k+1)}{2nr}\right)$.

Proof. Decodability: As each chunk is processed independently using the MPIR scheme in Construction 5 (PM-MBR-sep), from responses $A^i(s), i \in [n]$ we obtain $X^{f_1,s}, \ldots, X^{f_p,s}$ for $s \in \mathbb{M}_{r\times mr}(\mathbb{F}_q)$. Hence a user get $X^{f_1}, X^{f_2}, \ldots, X^{f_p}$ as desired.

Privacy: Since server i gets a uniformly distributed matrix $Q^i \in \mathbb{M}_{r\times mr}(\mathbb{F}_q)$ in all circumstances for every $i \in [n]$, and the distribution of Q^i does not depend on $f_1, \ldots, f_p$, server i obtains no information about the desired indices.

Analysis: We give an analysis when $p = \lfloor \frac{n-r}{k} \rfloor$ when the scheme has highest efficiency. First, we count the total amount of downloaded data. Since we vary query matrices submitted to each server $i, i \in [n]$ through all possibilities in $\mathbb{M}_{r\times mr}(\mathbb{F}_q)$, the number of elements in $\mathbb{M}_{r\times mr}(\mathbb{F}_q)$ that has exactly h all-zero rows is

$$\binom{r}{h}(q^{mr}-1)^{r-h}$$

as the number of possible rows which are not an all-zero row is $q^{mr}-1$. Hence, server i does not need to reply

$$\sum_{h=1}^{r} h\binom{r}{h}(q^{mr}-1)^{r-h} = r(q^{mr})^{r-1}$$

symbols, and the total download is

$$q^{mr^2}nr - nr(q^{mr})^{r-1}.$$

The size of desired records is $pq^{mr^2}\left(\frac{k(2r-k+1)}{2}\right)$, so retrieval rate for the scheme is

$$\begin{aligned}\frac{pq^{mr^2}\left(\frac{k(2r-k+1)}{2}\right)}{nr(q^{mr^2}-(q^{mr})^{r-1})} &= \left(\frac{q^{mr^2}}{q^{mr^2}-(q^{mr})^{r-1}}\right)\left(\frac{pk(2r-k+1)}{2nr}\right)\\ &= \left(\frac{q^{mr}}{q^{mr}-1}\right)\left(\frac{pk(2r-k+1)}{2nr}\right),\end{aligned}$$

where $\frac{pk(2r-k+1)}{2nr}$ is the retrieval rate of Construction 5 (PM-MBR-sep), so the retrieval rate is improved by a factor $\frac{q^{mr}}{q^{mr}-1}$. □

7.1.2 Applying the Averaging Technique to Existing Code-Based PIR Schemes

Remark that the averaging technique can be applied to many existing code-based PIR schemes where the queries are the sum of random matrices and deterministic matrices including Construction 1 in Chapter 5 which follows the mixed coding architecture, and Constructions 2-4 in Chapter 5, Constructions 6, 7 in Chapter 6, and Tajeddine and El Rouayheb's scheme using MDS codes in Section 4.2.1.2 which follow the separate coding architecture. The following discussion presents the application of the averaging technique on code-based PIR schemes in general.

Consider that we apply the averaging technique to an existing code-based PIR scheme. Suppose that each server stores A symbols. Remark that for PIR schemes following the separate coding architecture, each record is encoded independently where each server stores α symbols of each record, so $\mathrm{A} = m\alpha$ in this case. Suppose that the existing scheme can retrieve records over $\mathbb{F}_q$ of length $\bar{\ell}$. Assume that this scheme use a random matrix in $\mathbb{M}_{d\times \mathrm{A}}(\mathbb{F}_q)$ to generate queries where d is the number of subqueries. Hence, we can modify this scheme to a new scheme that can retrieve records of length $\ell = q^{d\mathrm{A}}\bar{\ell}$ in the same way as in Construction 8 (PM-MBR-sep). The number of elements in $\mathbb{M}_{d\times \mathrm{A}}(\mathbb{F}_q)$ that has exactly h all-zero rows is $\binom{d}{h}(q^{\mathrm{A}}-1)^{d-h}$. Hence, server i does not need to reply

$$\sum_{h=1}^{d} h\binom{d}{h}(q^{\mathrm{A}}-1)^{d-h} = d(q^{\mathrm{A}})^{d-1}$$

symbols, and the total download is

$$q^{d\mathrm{A}}nd - nd(q^{\mathrm{A}})^{d-1}.$$

Therefore, retrieval rate for the new scheme is

$$\begin{aligned}\frac{pq^{d\mathrm{A}}\bar{\ell}}{nd(q^{d\mathrm{A}}-(q^{\mathrm{A}})^{d-1})} &= \left(\frac{q^{d\mathrm{A}}}{q^{d\mathrm{A}}-(q^{\mathrm{A}})^{d-1}}\right)\left(\frac{p\bar{\ell}}{nd}\right)\\ &= \left(\frac{q^{\mathrm{A}}}{q^{\mathrm{A}}-1}\right)\left(\frac{p\bar{\ell}}{nd}\right),\end{aligned}$$

where $\frac{p\bar{\ell}}{nd}$ is the retrieval rate of the existing PIR scheme. Hence, the improvement factor in general is $\frac{q^{\mathrm{A}}}{q^{\mathrm{A}}-1}$ when we apply an averaging technique to existing code-based PIR schemes in this way. Note that in the separate coding architecture the improvement factor is

$$\frac{q^{m\alpha}}{q^{m\alpha}-1}. \tag{7.1}$$

7.2 PIR Schemes using $[2k, k]$ MDS codes

In this section, we construct a PIR scheme using $[2k, k]$ MDS codes over $\mathbb{F}_q$. In this scheme we assume that each record of length k is encoded by a $[2k, k]$ MDS code, so each server only stores one symbol of each record. Consequently, it achieves the best improvement factor (7.1) for fixed parameters q and m after applying the averaging technique. The idea of this construction is to exploit the structure of the parity check matrix in a standard form of $[n, k]$ linear codes. The parity check matrix provides $n-k$ parity-check equations. With the design of queries, these $n-k$ equations can be used to get access for $n-k$ symbols of the desired record. Since we use $[2k, k]$ MDS codes, we are able to reconstruct the desired record from k symbols by the property of the MDS codes.

7.2.1 Construction 9 (MDS-sep)

Suppose there are $2k$ non-communicating servers in the system that store a database X which consists of m records, each of length k, denoted by $X^1, X^2, \ldots, X^m \in \mathbb{F}_q^k$. Each record is encoded and distributed across $2k$ servers by the same $[2k, k]$ MDS code which can be written as

$$E_{1,2k,1}(X^j) = G_{2k\times k} \cdot X^j_{k\times 1},$$

where G is a generator matrix of the $[2k, k]$ MDS code. We simply write $E_{1,2k,1}(X^j)$ as C^j. We denote by C^j_i the i^{th} position of C^j. Since every non-systematic linear code can be transformed into a systematic code, we assume that our code is systematic. Then G can be written in the form

$$G = \begin{bmatrix} I_k \\ A \end{bmatrix},$$

where I_k is the identity matrix of size k, and $A \in \mathbb{F}_q^{k\times k}$. Write

$$X = \begin{bmatrix} X^1 & X^2 & \cdots & X^m \end{bmatrix},$$

we can see the entire system as

$$E_{m,2k,m}(X^1, \ldots, X^m) = \begin{bmatrix} C^1 & C^2 & \cdots & C^m \end{bmatrix} = G \cdot X,$$

and each server stores m symbols in total. We write $E_{m,2k,m}(X^1, \ldots, X^m)$ as C, and denote by C_i the row i of C which is all symbols stored in server i.

We assume that in the retrieval step the user wants to download X^f. The user submits a vector Q^i of length m over $\mathbb{F}_q$ to server i. Finally, server i computes and responds with an answer $A^i = Q^i C_i^T$. The retrieval steps are as follows:

(i) (Initialisation) The user generates a vector u of length m whose elements are chosen independently and uniformly at random over $\mathbb{F}_q$.

(ii) (Query Generation) The query vector Q^i is defined as

$$Q^i = \begin{cases} u, & \text{if } i = 1, \ldots, k \\ u + e_f, & \text{if } i = k+1, \ldots, 2k \end{cases}$$

where e_f is a the i^{th} unit vector of length m.

(iii) (Response Mappings) Each server i returns a symbol $A^i = Q^i C_i^T$.

(iv) (Recovery) Write a vector $A = (A^1, \ldots, A^{2k})$. Let H be a parity check matrix of the $[2k, k]$ MDS code in a standard form. The user computes $A \cdot H$.

Theorem 7.2. Construction 9 (MDS-sep) is an information-theoretically perfect PIR scheme with retrieve rate $\frac{1}{2}$ achieving asymptotic capacity for MDS codes.

Proof. We have that $e_f \cdot C_i^T = C_i^f$ when $i \in [2k]$. As H is a parity check matrix in a standard form, we can write

$$H = \begin{bmatrix} B \\ I_k \end{bmatrix},$$

where I_k is the identity matrix of size k, and $B \in \mathbb{F}_q^{k \times k}$. Hence,

$$\begin{aligned} A \cdot H &= (A^1, \ldots, A^{2k}) \cdot H \\ &= \left(u \cdot C_1^T, \ldots, u \cdot C_k^T, (u + e_f) \cdot C_{k+1}^T, \ldots, (u + e_f) \cdot C_{2k}^T\right) \cdot H \\ &= u \cdot C^T \cdot H + \begin{bmatrix} 0 & \cdots & 0 & C_{k+1}^f & C_{k+2}^f & \cdots & C_{2k}^f \end{bmatrix} \cdot H \\ &= 0 + \left(C_{k+1}^f, C_{k+2}^f, \cdots, C_{2k}^f\right) \\ &= \left(C_{k+1}^f, C_{k+2}^f, \cdots, C_{2k}^f\right). \end{aligned}$$

By the property of the MDS codes, a user can reconstruct X^f as desired.

To prove privacy, since server i gets a uniformly distributed vector $Q^i \in \mathbb{F}_q^m$ in all circumstances for every $i \in [2k]$, and the distribution of Q^i does not depend on f, server i obtains no information about the index f.

The total amount of downloaded data is $2k$ symbols, and the size of the desired record is k, so retrieval rate for Construction 9 (MDS-sep) is $\frac{1}{2} = \left(1 - \frac{k}{2k}\right)$ which means that this scheme achieves asymptotic capacity for MDS codes. □

7.2.2 Applying the Averaging Technique to Construction 9

Here we apply the averaging technique to Construction 9 (MDS-sep).

7.2.2.1 Construction 10 (MDS-sep)

We suppose a record X^j is of length $q^m k$ over $\mathbb{F}_q$ for all $j \in [m]$. Each record is then divided into q^m chunks of length k. Each chunk is encoded using the same systematic $[2k, k]$ MDS code and distributed across $2k$ non-communicating servers. The technique is to vary randomness that is used for query generation through all possible q^m random vectors for each chunk. Hence, for each server there must be a query which is an all-zero vector, so the server does not need to reply in this case, resulting in good download complexity in the worst case. Note that queries for each chunk can be calculated from just one query which means upload complexity here is still low. Indeed, each chunk of X^j is indexed by $X^{j,s}$ for $j \in [m], s \in \mathbb{F}_q^m$, and is encoded as

$$E_{1,2k,1}(X^{j,s}) = G \cdot X^{j,s},$$

where

$$G = \begin{bmatrix} I_k \\ A \end{bmatrix}$$

is a generator matrix in standard form of the MDS code. We write $E_{1,2k,1}(X^{j,s})$ as $C^{j,s}$, and denote by $C_i^{j,s}$ the i^{th} position of $C^{j,s}$. For each chunk $s \in \mathbb{F}_q^m$, we write

$$X(s) = \begin{bmatrix} X^{1,s} & X^{2,s} & \cdots & X^{m,s} \end{bmatrix},$$

so

$$E_{m,n,m}(X^{1,s}, \ldots, X^{m,s}) = \begin{bmatrix} C^{1,s} & C^{2,s} & \cdots & C^{m,s} \end{bmatrix} = G \cdot X(s),$$

and each server stores $q^m m$ symbols in total. We write $E_{m,2k,m}(X^{1,s}, \ldots, X^{m,s})$ as $C(s)$, and denote by $C(s)_i$ the row i of $C(s)$ which is all symbols of $X(s)$ stored in server i. We assume that the record X^f is demanded. The retrieval steps are

(i) (Initialisation) The user generates a vector u of length m whose elements are chosen independently and uniformly at random over $\mathbb{F}_q$.

(ii) (Query Generation) The query vector Q^i is defined as

$$Q^i = \begin{cases} u, & \text{if } i = 1, \ldots, k \\ u + e_f, & \text{if } i = k+1, \ldots, 2k \end{cases}$$

where e_f is a the i^{th} unit vector of length m.

(iii) (Response Mappings) For $i \in [2k]$, server i returns a symbol

$$A^i(s) = (Q^i + s) \cdot C(s)_i^T$$

for every $s \in \mathbb{F}_q^m$.

(iv) (Recovery) For $s \in \mathbb{F}_q^m$, write a vector $A(s) = \big(A^1(s), \ldots, A^{2k}(s)\big)$. Let H be a parity check matrix of the $[2k, k]$ MDS code in a standard form. The user computes $A(s) \cdot H$ for every $s \in \mathbb{F}_q^m$.

Theorem 7.3. Construction 10 (MDS-sep) is an information-theoretically perfect PIR scheme with retrieve rate $\frac{1-\frac{1}{2}}{1-(\frac{1}{q})^m}$ achieving capacity for MDS codes when $q = 2$.

Proof. We have that $e_f \cdot C(s)_i^T = C_i^{f,s}$ when $i \in [2k], s \in \mathbb{F}_q^m$. As H is a parity check matrix in a standard form, we can write

$$H = \begin{bmatrix} B \\ I_k \end{bmatrix},$$

where I_k is the identity matrix of size k, and $B \in \mathbb{F}_q^{k \times k}$. Hence,

$$\begin{aligned}
A(s) \cdot H &= \left(A^1(s), \ldots, A^{2k}(s)\right) \cdot H \\
&= ((Q^1 + s)C(s)_1^T, \ldots, (Q^n + s)C(s)_{2k}^T) \cdot H \\
&= ((u + s)C(s)_1^T, \ldots, (u + s)C(s)_k^T, \\
&\qquad (u + s + e_f)C(s)_{k+1}^T, \ldots, (u + s + e_f)C(s)_{2k}^T) \cdot H \\
&= (u + s)C(s)^T \cdot H + \begin{bmatrix} 0 & \cdots & 0 & C_{k+1}^{f,s} & C_{k+2}^{f,s} & \cdots & C_{2k}^{f,s} \end{bmatrix} \cdot H \\
&= 0 + \left(C_{k+1}^{f,s}, C_{k+2}^{f,s}, \ldots, C_{2k}^{f,s}\right).
\end{aligned}$$

Hence, the user gets

$$C_{k+1}^{f,s}, C_{k+2}^{f,s}, \ldots, C_{2k}^{f,s}$$

for every $s \in \mathbb{F}_q^m$. By the property of the MDS code, a user can reconstruct the chunk $X^{f,s}$ for every $s \in \mathbb{F}_q^m$, and hence obtains X^f as desired.

To prove privacy, since server i gets a uniformly distributed vector $Q^i \in \mathbb{F}_q^m$ in all circumstances for every $i \in [2k]$, and the distribution of Q^i does not depend on f, server i obtains no information about the index f.

The total amount of downloaded data is $2k(q^m - 1)$ symbols, and the size of the desired record is $q^m k$, so the retrieval rate for Construction 10 (MDS-sep) is

$$\frac{q^m k}{2k(q^m - 1)} = \frac{1 - \frac{1}{2}}{1 - (\frac{1}{q})^m},$$

which means that this scheme could achieve the capacity for $[2k, k]$ MDS codes when $q = 2$. However, from Theorem 2.19, the only existing $[2k, k]$ MDS code with $q = 2$ is the trivial $[2, 1]$ code.

As discussed in Section 7.1.2, the improvement factor when applying the averaging technique this way to an existing PIR scheme is $\frac{q^{m\alpha}}{q^{m\alpha}-1}$ and here we apply the technique to Construction 9 (MDS-sep) which has $\alpha = 1$, therefore this gives the most improvement for the separate coding architecture case. □

Chapter Summary

To sum up, we introduce the first application of the averaging technique on code-based PIR in this chapter. We start from applying the technique to transform Construction 5 into a new MPIR scheme with improved retrieval rate. Then we derive the improvement factor of the retrieval rate when we apply the averaging technique to any existing code-based PIR schemes where the queries are the sum of random matrices and deterministic matrices. Subsequently, we propose a new PIR scheme using $[2k, k]$ MDS codes, and show that it could achieve the highest improvement factor after applying the averaging technique.

Chapter 8

Conclusions

8.1 Summary of Contributions

In this thesis, we focus on the construction of PIR and MPIR schemes using product-matrix regenerating codes in order to minimise repair ratio in the system. Before our work, the only PIR scheme that uses regenerating codes is proposed by Shah et al. [36]. We give a general model to explain the encoding in code-based PIR falling into two classes of encoding - separate coding and mixed coding architecture. We propose a new metric to measure the efficiency of the repair, namely the repair ratio. We present various schemes for single-message PIR following both mixed coding and separate coding architecture in the storage.

Then we initiate the work on the multi-message problem with regenerating codes. We propose an MPIR model where each record is separately encoded by product-matrix regenerating codes and analyse the relationship between three metrics under this model. We derive a trade-off between cPoP and RR under this model, and then exhibit MPIR schemes that lie on the trade-off curve between cPoP and RR. The MPIR scheme using PM-MBR codes has the smallest possible repair ratio which benefits the situation where server failures should be taken into

account. While the retrieval rates of our schemes do not appear to be very good, they are the first schemes to achieve MPIR in regenerating codes and they give us a lower bound of what is achievable.

We also present a corresponding scheme using an averaging technique from [20] to improve the retrieval rate. This is the first application of the averaging technique on coded databases. We also give the improvement rate in general when applying the averaging technique to existing code-based PIR schemes in a similar way, and lastly provide a new PIR scheme using $[2k, k]$ MDS codes following the separate coding architecture that could achieve the best possible improvement rate after applying the averaging technique.

Lastly, Table 8.1 indicates the summary of our constructions in PIR/MPIR showing an underlying code in the encoding step with parameters n, α and ℓ and the efficiency metrics SO, cPoP and RR.

PIR Schemes	Encoding	Underlying Code	n	α	ℓ	SO	cPoP	RR
Construction 1	mixed	PM-MSR	$n \geq 4m-4$	$m-1$	$m-1$	$\frac{n}{m}$	4	2
Construction 2	separate	PM-MBR	$k+r$	mr	$\frac{k(2r-k+1)}{2}$	$\frac{2r(k+r)}{k(2r-k+1)}$	$\frac{2r(k+r)}{k(2r-k+1)}$	1
Construction 3	separate	PM-MSR	$3k-2$	$m(k-1)$	$k(k-1)$	$3-\frac{2}{k}$	$3-\frac{2}{k}$	2
Construction 4	separate	PM-MSR	$3k-3$	$m(k-1)$	$k(k-1)$	$3-\frac{3}{k}$	3	2
Construction 5	separate	PM-MBR	$n \geq k+r$	mr	$\frac{k(2r-k+1)}{2}$	$\frac{2nr}{k(2r-k+1)}$	$\frac{2nr}{k(2r-k+1)}$	1
Construction 6	separate	PM-MSR	$n \geq 3k-2$	$m(k-1)$	$k(k-1)$	$\frac{n}{k}$	$\frac{n}{pk}$	2
Construction 7	separate	PM-MSR	$n \geq 3k-3$	$m(k-1)$	$k(k-1)$	$\frac{n}{k}$	$\frac{n}{p(k-1)}$	2
Construction 8	separate	PM-MBR	$n \geq k+r$	$q^{mr^2}mr$	$q^{mr^2}\left(\frac{k(2r-k+1)}{2}\right)$	$\frac{2nr}{k(2r-k+1)}$	$\left(\frac{q^{mr}-1}{q^{mr}}\right)\left(\frac{2nr}{pk(2r-k+1)}\right)$	1
Construction 9	separate	$[2k,k]$ MDS	$2k$	m	k	2	2	k
Construction 10	separate	$[2k,k]$ MDS	$2k$	$q^m m$	$q^m k$	2	$2\left(\frac{q^m-1}{q^m}\right)$	k

Table 8.1: The summary of our works in PIR/MPIR showing an underlying code in the encoding step with parameters n, α and ℓ and metrics SO, cPoP and RR.

8.2 Future Works

There are several research problems arising from this thesis.

1. All existing schemes with regenerating codes (including [36],[42–44]) store a database by the product-matrix constructions from Rashmi et al. [21]. Can we construct PIR or MPIR schemes with other types of regenerating codes (for example [24–27])?

2. Since MSR codes are MDS codes but MBR codes are not MDS codes, the capacity results on PIR using MDS codes [32], and MPIR using MDS codes [35] can only be applied to PIR and MPIR using MSR codes, respectively. We propose PIR and MPIR schemes using MBR codes, and Lavauzelle et al. [44] propose a PIR scheme using MBR codes that has better retrieval rate. However, the capacity of MBR codes is still unknown. Is it possible to derive the capacity of PIR or MPIR using PM-MBR codes (or even more general MBR codes)?

3. In this thesis, Construction 1 in Chapter 5 is the only PIR scheme which follows the mixed coding architecture. Also, most existing code-based PIR schemes assume the separate coding architecture including the capacity result on MDS coded database. Sun and Tian [37] present a scheme that can break the MDS capacity using the mixed coding architecture. Can we apply their technique to construct better schemes using the mixed coding architecture?

4. All existing works on PIR or MPIR using regenerating codes assume that the servers are non-colluding. Can we construct schemes using regenerating codes in other variations of PIR such as PIR with colluding servers, or symmetric PIR?

Bibliography

[1] B. Chor, O. Goldreigh, E. Kushilevitz, and M. Sudan. "Private Information Retrieval". In: *Journal ACM* (Nov. 1998), pp. 965–981.

[2] B. Chor and N. Gilboa. "Computationally Private Information Retrieval (Extended Abstract)". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. STOC '97. New York, NY, USA, 1997, pp. 304–313.

[3] E. Kushilevitz and R. Ostrovsky. "Replication is not needed: single database, computationally-private information retrieval". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. 1997, pp. 364–373.

[4] J. P. Stern. "A New Efficient All-Or-Nothing Disclosure of Secrets Protocol". In: *Advances in Cryptology - ASIACRYPT '98*. Vol. 1514. 1998, pp. 357–371.

[5] C. Cachin, S. Micali, and M. Stadler. "Computationally Private Information Retrieval with Polylogarithmic Communication". In: *Advances in Cryptology — EUROCRYPT '99*. 1999, pp. 402–414.

[6] C. Aguilar Melchor and P. Gaborit. "Lattice-Based Computationally-Efficient Private Information Retrieval Protocol". In: *WEWORC 2007*. 2007.

[7] H. Lipmaa. "Oblivious Transfer Protocol with Log-Squared Communication". In: *Information Security*. 2005, pp. 314–328.

[8] J. Trostle and A. Parrish. “Efficient Computationally Private Information Retrieval from Anonymity or Trapdoor Groups”. In: *Information Security*. 2011, pp. 114–128.

[9] F. Olumofin and I. Goldberg. “Revisiting the Computational Practicality of Private Information Retrieval”. In: *Financial Cryptography and Data Security*. 2012, pp. 158–172.

[10] W. Gasarch. “A Survey on Private Information Retrieval”. In: *Bulletin of the EATCS* 82 (2004), pp. 72–107.

[11] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, and D. A. Karpuk. “Private Information Retrieval from Coded Databases with Colluding Servers”. In: *SIAM Journal on Applied Algebra and Geometry* 1.1 (2017), pp. 647–664.

[12] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, C. Hollanti, and S. E. Rouayheb. “Private Information Retrieval Schemes for Coded Data with Arbitrary Collusion Patterns”. In: *2017 IEEE International Symposium on Information Theory (ISIT)*. 2017, pp. 1908–1912.

[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.” In: *IEEE Transactions on Information Theory* 64.2 (2018), pp. 1000–1022.

[14] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, A. Horlemann-Trautmann, D. Karpuk, and I. Kubjas. “t-Private Information Retrieval Schemes Using Transitive Codes”. In: *IEEE Transactions on Information Theory* 65.4 (2019), pp. 2107–2118.

[15] H. Sun and S. A. Jafar. “The Capacity of Symmetric Private Information Retrieval”. In: *IEEE Transactions on Information Theory* 65.1 (2019), pp. 322–329.

[16] Q. Wang and M. Skoglund. “Symmetric Private Information Retrieval from MDS Coded Distributed Storage with Non-Colluding and Colluding Servers”. In: *IEEE Transactions on Information Theory* 65.8 (2019), pp. 5160–5175.

[17] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran. “A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster”. In: *Proc. 5th USENIX Workshop on Hot Topics in Storage and File Systems* (2013).

[18] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. “Network Coding for Distributed Storage Systems”. In: *IEEE Transactions on Information Theory* 56.9 (2010), pp. 4539–4551.

[19] S. Balaji, M. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. Kumar. “Erasure Coding for Distributed Storage: An Overview”. In: *Science China Information Sciences* (2018).

[20] S. R. Blackburn, T. Etzion, and M. B. Paterson. “PIR Schemes With Small Download Complexity and Low Storage Requirements”. In: *IEEE Transactions on Information Theory* 66.1 (2020), pp. 557–571.

[21] 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”. In: *IEEE Transactions on Information Theory* 57.8 (2011), pp. 5227–5239.

[22] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library. North-Holland Pub. Co., 1977.

[23] Y. Wu, A. G. Dimakis, and K. Ramchandran. “Deterministic Regenerating Codes for Distributed Storage”. In: *Proc. 45th Annual Allerton Conference on Control, Computing, and Communication*. Sept. 2007.

[24] C. Suh and K. Ramchandran. "Exact-Repair MDS Codes for Distributed Storage Using Interference Alignment". In: *2010 IEEE International Symposium on Information Theory*. 2010, pp. 161–165.

[25] Z. Wang, I. Tamo, and J. Bruck. "Long MDS Codes for Optimal Repair Bandwidth". In: *2012 IEEE International Symposium on Information Theory Proceedings*. 2012, pp. 1182–1186.

[26] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe. "Repair Optimal Erasure Codes Through Hadamard Designs". In: *IEEE Transactions on Information Theory* 59.5 (2013), pp. 3021–3037.

[27] N. Raviv, N. Silberstein, and T. Etzion. "Constructions of High-Rate Minimum Storage Regenerating Codes over Small Fields". In: *IEEE Transactions on Information Theory* 63.4 (2017), pp. 2015–2038.

[28] D. S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory*. Princeton University Press, 2005.

[29] T. H. Chan, S. Ho, and H. Yamamoto. "Private Information Retrieval for Coded Storage". In: *2015 IEEE International Symposium on Information Theory (ISIT)*. 2015, pp. 2842–2846.

[30] R. Tajeddine and S. El Rouayheb. "Private Information Retrieval from MDS Coded Data in Distributed Storage Systems". In: *2016 IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 1411–1415.

[31] S. Kumar, H. Lin, E. Rosnes, and A. Graell i Amat. "Achieving Maximum Distance Separable Private Information Retrieval Capacity with Linear Codes". In: *IEEE Transactions on Information Theory* 65.7 (2019), pp. 4243–4273.

[32] K. Banawan and S. Ulukus. "The Capacity of Private Information Retrieval From Coded Databases". In: *IEEE Transactions on Information Theory* 64.3 (2018), pp. 1945–1956.

[33] R. Zhou, C. Tian, T. Liu, and H. Sun. "Capacity-Achieving Private Information Retrieval Codes from MDS-Coded Databases with Minimum Message Size". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 2019, pp. 370–374.

[34] J. Xu and Z. Zhang. "On Sub-Packetization and Access Number of Capacity-Achieving PIR Schemes for MDS Coded Non-Colluding Servers". In: *Science China Information Sciences* 61 (2018), 100306:1–100306:16.

[35] Y. Zhang and G. Ge. *Private Information Retrieval from MDS Coded Databases with Colluding Servers under Several Variant Models*. 2017. arXiv: 1705.03186 [cs.IT].

[36] N. B. Shah, K. V. Rashmi, and K. Ramchandran. "One Extra Bit of Download Ensures Perfectly Private Information Retrieval". In: *IEEE International Symposium on Information Theory* (June 2014), pp. 856–860.

[37] H. Sun and C. Tian. "Breaking the MDS-PIR Capacity Barrier via Joint Storage Coding". In: *Information* 10.9 (2019).

[38] 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* (Mar. 2013), pp. 325–336.

[39] C. Huang, M. Chen, and J. Li. "Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems". In: *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*. 2007, pp. 79–86.

[40] H. Sun and S. A. Jafar. "The Capacity of Private Information Retrieval". In: *IEEE Transactions on Information Theory* 63.7 (2017), pp. 4075–4088.

[41] K. Banawan and S. Ulukus. "Multi-Message Private Information Retrieval: Capacity Results and Near-Optimal Schemes". In: *IEEE Transactions on Information Theory* 64.10 (2018), pp. 6842–6862.

[42] C. Dorkson and S. Ng. *Private Information Retrieval Using Product-Matrix Minimum Storage Regenerating Codes*. 2018. arXiv: 1805.07190 [cs.IT].

[43] C. Dorkson and S. Ng. *Multi-Message Private Information Retrieval Using Product-Matrix MSR and MBR Codes*. 2018. arXiv: 1808.02023 [cs.IT].

[44] J. Lavauzelle, R. Tajeddine, R. Freij-Hollanti, and C. Hollanti. "Private Information Retrieval Schemes With Product-Matrix MBR Codes". In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 441–450.