Fault tolerant capacity achieving PIR protocols based on combinatorial designs

A Project Report

submitted by

MOHIT SHRIVASTAVA

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR AND MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2021

THESIS CERTIFICATE

This is to certify that the thesis titled **Private Information Retrieval**, submitted by **Mohit Shrivastava**, to the Indian Institute of Technology, Madras, for the award of the degree of **BTech. + M.Tech**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Pradeep Sarvepalli
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my research guide Dr. Pradeep Sarvepalli for his continuous guidance and support. His motivation, positivity and and enthusiasm throughout the duration of the research kept me going. His insights on the topic, immense knowledge and numerous discussions helped me explore and enjoy the field. I would be ever grateful to him for all the support and guidance.

Secondly, I would like to thank SAMSUNG IITM PRAVARTAK TECHNOLOGIES FOUNDATION for the financial support towards the later end of the work.

Finally, I would like to thank my friends and family to keep me motivated and strong through these tough times of fighting the novel coronavirus. Without their continuous support and care it wouldn't had been possible to smoothly carry out the research and learn so much.

ABSTRACT

In this work we study the problem of private information retrieval where a user seeks to retrieve one of the F files from a cluster of N non-colluding servers without revealing the identity of the requested file. In our setting the servers are storage constrained in that they can only store a fraction $\mu = t/N$ of each file. Furthermore, we assume that the files are stored in an uncoded fashion. The rate of a PIR protocol is defined as the ratio of the file size to the total number of bits downloaded. The maximum achievable rate is referred to as capacity. It was previously shown that there are capacity achieving PIR protocols when the file size is N^F and complete files were stored on all the servers. These results were further extended for the case when servers store only a fraction of each file. We propose a novel uncoded PIR protocol based on combinatorial designs that are also capacity achieving when the file size is $N/\gcd(N,t) \times t^{F-1}$. Secondly, we also analyse the scenario when one of the servers fails during the retrieval phase. Our protocol achieves capacity for the one residual system after one server has failed, and can also distribute the load uniformly on the remaining servers if BIBD's are used for the storage scheme. Furthermore, if symmetric BIBD's are used for the storage scheme, the rate of PIR protocol for two residual system is fixed and does not depend on which two servers fail. In the proposed PIR protocol, the given system is projected to multiple instances of reduced systems with replicated servers having full storage capacity. The subfiles stored in these various instances are separately retrieved and lifted to solve the PIR problem for the original system.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

PIR	Private Information Retrieval
FS-PIR	Full Storage - Private Information Retrieval
SC-PIR	Storage Constrained - Private Information Retrieval
BIBD	Balanced Incomplete Block Design

CHAPTER 1

INTRODUCTION

With the commencement of Information age the amount of stored data has grown immensely everyday. The increasing availability of internet-accessible data sources has lead to a transition in the way these service providers and database owners collect and use personal information about users. While this sensitive captured information is useful in providing personalized and improved service to the users, it can potentially also be used to harass, disrepute or financially trouble the targeted users. This captured user information if sold to third party increases the risk of exploitation manifold. Selling of user's activity and query information brings large monetary benefits for the database owners and huge privacy threats to the user.

In recent years, the field of Private Information Retrieval has gained a lot of attention owing to the fact that more and more Internet users are now beginning to care about privacy of their online activities. PIR systems allow a user to query a cluster of servers and retrieve the desired file without revealing any information about the desired file index to any of the individual servers.

PIR systems are particularly important to protect users from surveillance, monitoring and profiling. A few services where PIR systems are of particular interest include location based services, patent database search, real-time stock quotes etc. In such a scenario it becomes important that the querying of servers by the user does not leak any information about the user's requirements.

1.1 Problem Background

The notion of Private Information Retrieval (PIR) was first introduced by Chor *et al.* in [1], and since then, the field has immensely grown [2, 3, 4, 5, 6, 7, 8]. Besides its direct application, PIR has interesting connections with other problems such as network coding [9], index coding [10, 11] and secret-sharing schemes [12, 13] .

The basic setting of PIR considers the servers to be non-colluding i.e. the servers cannot cooperate with each other to recover information about the query. Trivially, one can achieve privacy by downloading all the files. Clearly, this is an inefficient method to achieve privacy. A chief problem of PIR protocols is to achieve privacy efficiently. Efficiency is typically measured in terms of the rate of the protocol. The ratio of size of retrieved file to the amount to data downloaded is referred to as the PIR rate. A higher PIR rate signifies higher efficiency, and the highest achievable PIR rate is referred to as the PIR capacity. The reciprocal of PIR rate is referred to as download cost per bit and is a measure of number of bits that need to be downloaded to retrieve one bit of the desired file. The optimal download cost per bit is the lowest achievable download cost per bit.

PIR protocols can store the files in a coded form or an uncoded form. In the recent years there has been a growing interest in uncoded PIR due to the simplicity of encoding and decoding the files and also because they can offer competitive performances compared to coded protocols [4, 8, 14]. Uncoded PIR systems consist of a collection of N servers and F files which are either stored fully or partially on each of the servers. The placement of these F files on the servers constitutes the storage scheme. Now, any file that the user desires is characterized by its index and to retrieve the desired file the users generates N query sets and sends one to each of the servers. Servers on receiving the queries respond by returning answers based on the query and server contents. The user must be able to recover the desired file from the answers obtained from the servers and additionally the queries must not leak any information about the index of user's desired file. The generation of queries and recovery of desired file form the retrieval scheme for any PIR protocol.

1.2 Contributions

In this work we propose a novel PIR protocol for uncoded systems storing a fraction of each file. While the storage capacity is identical for the servers, they need not store the same content. The proposed PIR protocol has two main components: a storage scheme and a retrieval scheme. The storage scheme is based on combinatorial designs while the retrieval scheme works by projecting the given system to multiple instances of smaller systems with replicated servers having full storage capacity. It uses the retrieval scheme of full storage PIR systems for these smaller instances. The idea of projecting storage constrained systems to Full storage PIR systems has been common to storage constrained PIR protocols and has previously been used by Tandon *et al.* [15], Zhang *et al.* [14] and Woolsey *et al.* [16].

Following are the key contributions of our work:

1. Proposal of a capacity achieving novel PIR protocol with low subpacketization for uncoded storage constrained PIR systems.
2. Robust proofs of privacy and correctness, which have often been missing in previous literature in the field [15, 16, 17].
3. Fault tolerant PIR protocol that achieves capacity for the new system after any of the one server has failed (one residual system).
4. Uniform load distribution even after failure of one server.
5. Fixed rate independent of which two servers fail for the two residual system.

Our scheme is flexible and can easily accommodate a range of system parameters. As our schemes are more generally based on tactical configuration, we also propose a way for constructing different tactical configurations given a set of parameters.

Compared to the existing work on storage constrained PIR by Tandon *et al.* [15] where authors require a subpacketization of $\binom{N}{t}t^F$, our protocol requires a way lesser subpacketization as low as $\frac{N}{\gcd(N,t)}(t-1)$. Although Woolsey *et al.* in [16] also achieve low subpacketization, our storage constrained PIR protocols offers subpacketization lesser than theirs by a factor of $\gcd(N,t)$ and has extra benefits of fault tolerance. It is able to achieve a uniform load distribution for the one residual systems which the protocol of [16] is unable to achieve. Also, for the two residual system the rate of PIR protocol of [16] depends on which two servers fail whereas our protocol has a fixed

rate irrespective of which two servers fail. Also, in [16] authors provide two different protocols one for $t \in \mathbb{Z}$ and one for $t \notin \mathbb{Z}$, whereas ours is a unified protocol. The protocol of [16] can be considered as a special case of our protocol.

1.3 Thesis organisation

The rest of the thesis is organized as follows. In Chapter 2 we briefly review the different PIR schemes available for retrieval of data from uncoded full storage PIR systems where each of the F files is fully stored on all N servers. Since our retrieval scheme is based on such full storage PIR protocols we understand these schemes with examples and note out the key points behind their working and the constraints required to be able to apply them. Then we move on to uncoded homogeneous storage constrained PIR systems and review the foundation work in the field. Also, we review the information theoretic bounds on achievable rates for both the full storage and storage constrained PIR systems. Finally, since the storage scheme of our proposed protocol is based on combinatorial designs, we review different classes of combinatorial designs and their properties.

In Chapter 3 we propose a capacity achieving PIR scheme for Homogeneous storage constrained PIR systems and provide a robust proof of its privacy and correctness. We then illustrate the scheme through examples and quantify the minimum file size required for our protocol under the usage of different full storage retrieval schemes. Interestingly, we are able to achieve minimum file size possible for any capacity achieving PIR protocol with equally sized subfiles. This is in accordance to a recent independent work on homogeneous PIR systems.

In Chapter 4 we study the scenario where one of the servers fails and the user has to retrieve the desired file from the set of available $N-1$ servers. We show that our protocol is capacity achieving for the new system. Also, our protocol is able to uniformly distribute the excess load due to server failure on rest of the available servers. Then we analyze the scenario where two servers fail. We observe that irrespective of which two servers fail, we always lose the same fraction of capacity which means that the achievable rate does not depend on which two servers fail. The achievable rate goes to capacity as $N \to \infty$.

Finally, in Chapter 5 we briefly discuss directions where our work can be possibly extended. These include the setting of Heterogeneous storage constrained PIR and Weakly PIR.

1.4 Notation

For a positive integer n, we use the notation $[n]$ to denote all the natural numbers upto n, i.e. $[n] = \{1, 2, ..., n\}$. For $n_1, n_2 \in \mathbb{Z}$, we use the notation $A_{n_1:n_2}$ to denote the set $\{A_{n_1}, A_{n_1+1}, ..., A_{n_2}\}$ if $n_1 \leq n_2$, and null set otherwise.

CHAPTER 2

BACKGROUND

In this chapter we review two important setups for Private information retrieval and summarise existing results. The general setup for PIR consists of a system of N non-colluding servers storing F files in parts or completely. The user wishes to retrieve the θ-th file without leaking any information about θ to any of the servers. The user generates N query sets and sends one to each of the servers. The servers on receiving these query sets respond with answers, from which the user must be able to retrieve the desired file. Also, the query sets must not leak any information about θ to any of the servers. We first review the setup of Full Storage PIR often known as FS-PIR or Classical PIR.

2.1 Full Storage Private Information Retrieval (FS-PIR)

Consider a system with N non-colluding servers $S_1, S_2, \ldots, S_N$ and F independent files $W_1, W_2, \ldots, W_F$, each L bit long and stored fully on all the servers. A user privately generates index $\theta \in [F]$ and wants to retrieve file W_θ without leaking any information about θ to any of the individual servers. In order to do so, it generates query sets $Q_1^{[\theta]}, Q_2^{[\theta]}, \ldots Q_N^{[\theta]}$, where $Q_n^{[\theta]}$ is sent to server S_n.

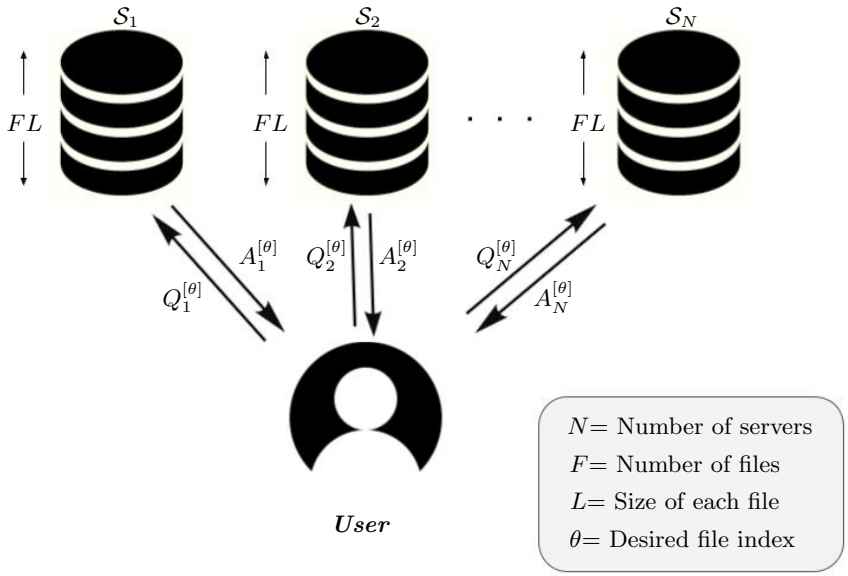


Figure 2.1: FS-PIR setup

2.1.1 Requirements for PIR

Since the queries are generated without any apriory information about file data,

$$I(W_1, W_2, \ldots, W_F; Q_1^{[\theta]}, Q_2^{[\theta]}, \ldots, Q_N^{[\theta]}) = 0 \quad \text{for all } \theta \in [F]. \tag{1}$$

Server S_n on receiving query $Q_n^{[\theta]}$ responds with answer $A_n^{[\theta]}$ which is a deterministic function of $Q_n^{[\theta]}$ and $W_1, W_2, \ldots, W_F$ i.e.

$$H(A_n^{[\theta]}|Q_n^{[\theta]}, W_1, W_2, \ldots, W_F) = 0 \text{ for all } n \in [N], \theta \in [F] \tag{2}$$

The desired file W_θ is then retrieved from all the answers i.e

$$H(W_\theta|A_1^{[\theta]}, A_2^{[\theta]}, \ldots, A_N^{[\theta]}) = 0 \text{ for all } \theta \in N \qquad \text{(Recoverability)}$$

To ensure that the user request is private the following privacy constraint must be satisfied

$$I(\theta; Q_n^{[\theta]}, A_n^{[\theta]}, W_1, W_2, \ldots, W_F)) = 0 \text{ for all } \theta \in [F], n \in [N] \qquad \text{(Privacy)}$$

or equivalently

$$(Q_n^{[1]}, A_n^{[1]}, W_1, W_2, \ldots, W_F) \sim (Q_n^{[\theta]}, A_n^{[\theta]}, W_1, W_2, \ldots, W_F) \text{ for all } \theta \in [F], n \in [N]$$

2.1.2 Capacity

For any PIR protocol the retrieval rate R is defined as the ratio of number of bits retrieved to the total number of bits downloaded i.e. $R = L/D$, where D is the total number of bits downloaded. The maximum achievable retrieval rate for any PIR system is termed as its PIR-capacity (usually called just capacity).

For the Full Storage PIR system, capacity was determined by Sun and Jafar in 2017 [4].

Theorem 1 *For a system of N servers storing F files completely, the maximum achievable rate i.e. capacity is given by*

$$C = \big(1 + \frac{1}{N} + \frac{1}{N^2} + \cdots + \frac{1}{N^{F-1}}\big)^{-1} \tag{2.1}$$

2.1.3 Capacity achieving Protocols

Now, we review the different capacity achieving FS-PIR protocols in the existing literature. Since the retrieval scheme of of our SC-PIR protocol is based on FS-PIR schemes, we review these in detail.

Capacity achieving Protocol 1 (FS-PIR protocol 1)

We first review the capacity achieving protocol proposed by Sun and Jafar in [4].

The protocol assumes file size to be $L = N^F$ for all files and is based on 3 key principles:

(i) Enforcing symmetry across servers

(ii) Enforcing file Symmetry within the query to each server

(iii) Exploiting side information of undesired files to retrieve new desired information

Let us first try to understand the protocol through an example.

Example 1 *Consider 2 files a and b each of length $L = 4$ bits both stored on 2 servers S_1 and S_2. We represent bits of file a and b by a_i and b_i, $i \in [4]$.*

Suppose the user wants to retrieve file a. So he starts by querying S_1 by first bit of file a i.e a_1. Next, to make his query symmetric wrt files at S_1 (done so as to confuse the server) he also queries S_1 for bit b_1. Similarly, he queries S_2 for bits a_2 and b_2 which enforces symmetry across servers. At this point the user has 2 desired bits - a_1, a_2 and 2 side information bits -b_1, b_2. Now he can mix a unknown desired bit with known side information and queries from the servers. So he queries $a_3 + b_2$ from server S_1 and $a_4 + b_1$ from server S_2. This way he can exploit side information of undesired file while keeping the query structure symmetric wrt files and also symmetric across servers.

S_1	S_2
a_1	a_2
b_1	b_2
$a_3 + b_2$	$a_4 + b_1$

From point of view of the servers, each server has received query for 1 bit of file a, 1 bit of file b and a linear combination of 1 bit of file a and 1 bit of file b. So, there's a discrepancy in what the user actually wants to retrieve.

Similarly file b can be retrieved as:

S_1	S_2
a_1	a_2
b_1	b_2
$a_2 + b_3$	$a_1 + b_4$

Note that 4 bits of desired file are retrieved by by downloading a total of 6 bits, so rate $R = \frac{4}{6} = \frac{2}{3}$, which matches the capacity for this case ($C = (1 + \frac{1}{2})^{-1} = \frac{2}{3}$ by eq. (2.1)).

Next, we understand the PIR protocol proposed by Sun and Jafar [4] in detail.

Suppose the user wants to retrieve file W_θ.

For all $i \in [F]$, define vectors

$$U_i = [u_i(1), u_i(2), \ldots, u_i(N^F)]$$

Let the term 'f-sum' be used to denote an expression representing sum of f distinct variables each drawn from a different U_i vector, i.e. $u_{i_1}(j_1) + u_{i_2}(j_2) + \cdots + u_{i_f}(j_f)$, where $i_1, i_2, \ldots i_f \in [F]$ and are all distinct. Furthermore the above f-sum is said to be f-sum of type $\{i_1, i_2, \ldots i_f\}$.

Firstly, a fixed query structure $Q(n, \theta)$ for the query to be sent to server S_n for retrieving file W_θ is generated. Each $Q(n, \theta)$ must be a union of F disjoint subsets called blocks. Block f, $f \in [F]$ must contain only f-sums . There are total $\binom{F}{f}$ possible types of f-sums. Block f, must contain all of them. Also, it is required that each type of f-sum has exactly $(N-1)^{f-1}$ distinct instances.

For each $i \in [F]$, let $new(U_i)$ be a function that returns the "next" variable in U_i, starting from $u_i(1)$. Each block is partitioned into two subsets —a subset $\mathcal{M}$ that contains the f-sums which include a variable from U_θ and a subset $\mathcal{I}$ which contains all the remaining f-sums, which contain no symbols from U_θ.

The scheme for generating query sets $Q(n, \theta)$, $n \in [N]$ from U_i variables is presented in Algorithm 1.

Once the query sets $Q(n, \theta)$ have been generated, a ordered representation is obtained by first ordering the blocks in increasing order of block index, and then within the f^{th} block arranging the "types" of f-sum in increasing lexicographic order. The ordering of query sets $Q(n, \theta)$ is represented by $\vec{Q}(n, \theta)$.

Finally, the U_i variables have to be mapped to the file bits in order to generate the actual query vectors, $Q_n^{[\theta]}$.

Suppose the i^{th} file W_i, $i \in [F]$ is represented as $W_i = [w_i(1), w_i(2), \ldots, w_i(N^F)]$, where $w_i(j)$ represents the j^{th} bit of file W_i. The user privately chooses permutations

Algorithm 1 FS-Capacity achieving Protocol 1 [4]
Input: θ
Output: Query sets $Q(n,\theta), n \in [N]$

Initialize: All query sets as null sets. Also initialize Block $\leftarrow 1$
for $n \in [N]$ **do**

$$Q(n,\theta,Block,\mathcal{M}) \leftarrow new(U_\theta)$$
$$Q(n,\theta,Block,\mathcal{I}) \leftarrow \bigcup_{i\in[F]\setminus\theta} new(U_i)$$

end for
for $Block \in \{2,3,\ldots,F\}$ **do**
  for $n \in [N]$ **do**
    for each $n' \in [N]$ & $n' \neq n$ **do**
      for each $q \in Q(n',\theta,Block-1,\mathcal{I})$ **do**

$$Q(n,\theta,Block,\mathcal{M}) \leftarrow Q(n,\theta,Block,\mathcal{M}) \cup \{new(U_\theta)+q\}$$

      end for
    end for
    for distinct $\{i_1,i_2,\ldots,i_{Block}\} \in [F]\setminus\theta$ **do**
      for $i \in [(N-1)^{Block-1}]$ **do**

$$Q(n,\theta,Block,\mathcal{I}) \leftarrow Q(n,\theta,Block,\mathcal{I}) \bigcup \{new(U_{i_1}) + new(U_{i_2}) + \cdots + new(U_{i_{Block}})\}$$

      end for
    end for
  end for
end for
for $n \in [N]$ **do**

$$Q(n,\theta) \leftarrow \bigcup_{Block\in[F]} \big(Q(n,\theta,Block,\mathcal{I}) \cup Q(n,\theta,Block,\mathcal{M})\big)$$

end for

$\gamma_1, \gamma_2, \ldots, \gamma_F$, uniformly randomly from all possible $(N^F)!$ permutations over the index set $[N^F]$, so that the permutations are independent of each other and also of θ . The U_i variables are mapped to the files W_i through the random permutation γ_i ,$\forall i \in [F]$.

Let Γ denote an operator that replaces every instance of $u_i(j)$ with $w_i(\gamma_i(j)), \forall\ i \in [F],\ j \in [N^F]$. This random mapping, applied to $\vec{Q}(n, \theta)$ produces the actual query vector $Q_n^{[\theta]}$ to be sent to server S_n for retrieving W_θ as

$$Q_n^{[\theta]} = \Gamma(\vec{Q}(n, \theta)) \tag{2.2}$$

Example 2 *Consider 3 files $a,\ b$ and c each of 27 bits stored on 3 servers S_1, S_2 and S_3. To make the notation simpler we use $W_1 = a,\ W_2 = b\ W_3 = c$, $U_1 = [a_1, a_2, \ldots, a_{27}]$, $U_2 = [b_1, b_2, \ldots, b_{27}]$ and $U_3 = [c_1, c_2, \ldots, c_{27}]$.*

The output of Algorithm 1 for $\theta = 1$ i.e retrieving file a is as follows:

S_1	S_2	S_3
a_1	a_2	a_3
b_1	b_2	b_3
c_1	c_2	c_3
$a_4 + b_2$	$a_8 + b_1$	$a_{12} + b_1$
$a_6 + b_3$	$a_{10} + b_3$	$a_{14} + b_2$
$a_5 + c_2$	$a_9 + c_1$	$a_{13} + c_1$
$a_7 + c_3$	$a_{11} + c_3$	$a_{15} + c_2$
$b_4 + c_4$	$b_6 + c_6$	$b_8 + c_8$
$b_5 + c_5$	$b_7 + c_7$	$b_9 + c_9$
$a_{16} + b_6 + c_6$	$a_{20} + b_4 + c_4$	$a_{24} + b_4 + c_4$
$a_{17} + b_7 + c_7$	$a_{21} + b_5 + c_5$	$a_{25} + b_5 + c_5$
$a_{18} + b_8 + c_8$	$a_{22} + b_8 + c_8$	$a_{26} + b_6 + c_6$
$a_{19} + b_9 + c_9$	$a_{23} + b_9 + c_9$	$a_{27} + b_7 + c_7$

The algorithm first starts by generating 1-sums for each server. There are only 3 possible types of 1-sums : a_i, b_i and c_i , where $i \in [9]$. Each 1 sum has to occur $(N-1)^{1-1} = 1$ times in each query set which leads to the Block 1 shown in the query table above. Next, 2-sums are generated. There are total $\binom{3}{2} = 3$ possible 2-sums

and each type of f-sum has to occur $(3-1)^{2-1}=2$ times in each query set. 2-sums of type $a+b$ or $a+c$ correspond to subset $\mathcal{M}$ and should use the side information available from Block 1. 2-sums of type $b+c$ correspond to subset $\mathcal{I}$ and should be used as side information in Block 3. As per this, Block 2 is generated in the above query table. Finally, Block 3 has to be generated using 3-sums. Since $\binom{3}{3}=1$, there is only 1 type of 3-sum and it has to occur $(3-1)^{3-1}=4$ times in each query set. Since, this corresponds to subset $\mathcal{M}$, it should exploit the side information available from Block 2 to retrieve bits of desired file i.e. a. In the tables shown, different blocks are separated by horizontal lines and numbered from 1 to F.

The output of Algorithm 1 for $\theta=2$ i.e retrieving file b is as follows:

S_1	S_2	S_3
a_1	a_2	a_3
b_1	b_2	b_3
c_1	c_2	c_3
a_2+b_4	a_1+b_8	a_1+b_{12}
a_3+b_6	a_3+b_{10}	a_3+b_{14}
a_4+c_4	a_6+c_6	a_8+c_8
a_5+c_5	a_7+c_7	a_9+c_9
b_5+c_2	b_9+c_1	$b_{13}+c_1$
b_7+c_3	$b_{11}+c_3$	$b_{15}+c_3$
$a_6+b_{16}+c_6$	$a_4+b_{20}+c_4$	$a_4+b_{24}+c_4$
$a_7+b_{17}+c_7$	$a_5+b_{21}+c_5$	$a_5+b_{25}+c_5$
$a_8+b_{18}+c_8$	$a_8+b_{22}+c_8$	$a_6+b_{26}+c_6$
$a_9+b_{19}+c_9$	$a_9+b_{23}+c_9$	$a_7+b_{27}+c_7$

The output of Algorithm 1 for $\theta=3$ i.e retrieving file c is as follows:

S_1	S_2	S_3
a_1	a_2	a_3
b_1	b_2	b_3
c_1	c_2	c_3
$a_4 + b_4$	$a_6 + b_6$	$a_8 + b_8$
$a_5 + b_5$	$a_7 + b_7$	$a_9 + b_9$
$a_2 + c_4$	$a_1 + c_8$	$a_1 + c_{12}$
$a_3 + c_6$	$a_3 + c_{10}$	$a_2 + c_{14}$
$b_2 + c_5$	$b_1 + c_9$	$b_1 + c_{13}$
$b_3 + c_7$	$b_3 + c_{11}$	$b_2 + c_{15}$
$a_6 + b_6 + c_{16}$	$a_4 + b_4 + c_{20}$	$a_4 + b_4 + c_{24}$
$a_7 + b_7 + c_{17}$	$a_5 + b_5 + c_{21}$	$a_5 + b_5 + c_{25}$
$a_8 + b_8 + c_{18}$	$a_8 + b_8 + c_{22}$	$a_6 + b_6 + c_{26}$
$a_9 + b_9 + c_{19}$	$a_9 + b_9 + c_{23}$	$a_7 + b_7 + c_{27}$

Note that this construction retrieves 27 desired file bits by downloading a total of 39 bits, so its rate is 27/39 = 9/13, which matches the capacity for this case ($C = (1 + 1/3 + 1/9)^{-1} = 9/13$ by eq. (2.1)).

Key points about the algorithm

1. For all $f \in [F]$, Block f contains exactly $(N-1)^{f-1}$ instances of f-sums of each possible type.
2. No $u_i(j), j \in [N^F]$ variable appears more than once in query set to any given server.
3. Exactly N^{F-1} variables for each $U_i, i \in [F]$ appear in the query set $Q(n, \theta), n \in [N]$.
4. The size of each query set is $N^{F-1} + \frac{1}{N-1}(N^{F-1} - 1)$.

Correctness of the algorithm follows from the fact that each bit of the desired file appears in the query sets either without any interference (i.e they are trivially retrieved) or appear mixed with side information which is available from other server. Thus each bit of the desired file can be retrieved and hence the algorithm is correct.

The key idea behind privacy is that regardless of θ , every realization of the query vector that fits the query structure is equally likely because of the uniformly random permutation Γ.

Finally, since the size of each query set is $N^{F-1} + \frac{1}{N-1}(N^{F-1} - 1)$, it easily follows that the algorithm achieves capacity.

Capacity achieving Protocol 2 (FS-PIR protocol 2)

The same authors Sun and Jafar soon came up with an improvement of Protocol 1 in terms of minimum file size required [18]. They proposed a new protocol which requires file size to be a multiple of N^{F-1} and still achieves capacity. Also, they showed that this is the minimum file size any capacity achieving protocol can offer and hence established the optimality of the protocol in terms of file size. The key observation which allows to reduce the file size is that enforcing symmetry across servers is not a necessary condition for PIR. The protocol exploits this fact and achieves capacity while preserving privacy with a smaller file size.

The protocol assumes file size to be $L = N^{F-1}$ for all files and is based on 2 key principles:

(i) Enforcing file Symmetry within the query to each server

(ii) Exploiting side information of undesired files to retrieve new desired information

Apart from the notation from protocol 1, the authors introduce two more terms to present the new protocol: $Count(Q, i_{[1:f]})$ which counts the number of f-sums of type $\{i_1, i_2, \dots i_f\}$ that are present in Q and $Max(Q, f)$ which denotes the maximum number of f-sums of same type in Q with the maximization being across all types of f-sums.

As per this notation, file symmetry translates to the condition that for every type of f-sum $\{i_1, i_2, \dots i_f\}$

$$Count(Q, i_{[1:f]}) = Count(Q, i'_{[1:f]}) \quad \text{for all } i_{[1:f]}, i'_{[1:f]} \tag{2.3}$$

To simplify the presentation of the protocol, the authors first present two subroutines *M-Sym* and *Exploit-SI*.

M-Sym subroutine takes as input a set Q comprising of various f-sums and outputs a set Q^* comprised of additional terms needed to be included in Q to make it file

symmetric. It does so by checking if there are $Max(Q, f)$ number of f-sums of type $\{i_1, i_2, \ldots i_f\}$, and if not then it generates the required number of instances to bring the count to $Max(Q, f)$. This is done for all types of f-sums and for all $f \in [F]$.

Algorithm 2 M-Sym Algorithm
Input: Q
Output: Q^*

Initialize: $Q^* \leftarrow \phi$
for $f \in [F]$ **do**
  for each $i_{1:f}$ **do**
    if $Count(Q, i_{[1:f]}) < Max(Q, f)$ **then**
      for $i \in [Max(Q, f) - Count(Q, i_{[1:f]})]$ **do**

$$Q^* \leftarrow Q^* \cup \{new(U_{i_1}) + new(U_{i_2}) + \cdots + new(U_{i_f})\}$$

      end for
    end if
  end for
end for

The *Exploit-SI* subroutine simply formalizes the side-information exploitation step which is the same as in Protocol 1. This subroutine take N query sets $Q_1, Q_2, \ldots, Q_N$ as input which are comprised of side information terms and have not been exploited. It then produces N new sets $Q_1^{'}, Q_2^{'}, \ldots, Q_N^{'}$ which are constructed by combining a new variable U_θ with side information terms of other query sets.

Algorithm 3 Exploit-SI Algorithm
Input: $Q_1, Q_2, \ldots, Q_N$
Output: $Q_1^{'}, Q_2^{'}, \ldots, Q_N^{'}$

Initialize: All output are initialized as null sets.
for $n \in [N]$ **do**
  for $n^{'} \in [N], n^{'} \neq n$ **do**
    for each $q \in Q_{n'}$ **do**

$$Q_{n'} \leftarrow Q_{n'} \cup \{new(U_\theta) + q\}$$

    end for
  end for
end for

Then, using these 2 subroutines the protocol is presented as:

Algorithm 4 FS-Capacity achieving Protocol 2
Input: θ
Output: Query sets $Q(n,\theta), n \in [N]$

1: *Initialize:* All query sets as null sets. Also initialize Block $\leftarrow 1$

2:

$$Q(1,\theta,Block,\mathcal{M}) \leftarrow new(U_\theta)$$

$$Q(1,\theta,Block,\mathcal{I}) \leftarrow \textit{M-Sym}(Q(1,\theta,Block,\mathcal{M}))$$

$$\forall n \in [2:N], \quad Q(n,\theta,Block,\mathcal{M}) \leftarrow \phi, \;\; Q(n,\theta,Block,\mathcal{I}) \leftarrow \phi$$

3: **for** $Block \in \{2,3,\ldots,f\}$ **do**

4:

$$(Q(1,\theta,Block,\mathcal{M}),\ldots,Q(N,\theta,Block,\mathcal{M})$$
$$\leftarrow \textit{Exploit-SI}(Q(1,\theta,Block-1,\mathcal{I}),\ldots,Q(N,\theta,Block-1,\mathcal{I}))$$

5: **for** $n \in [N]$ **do**

$$Q(n,\theta,Block,\mathcal{I}) \leftarrow \textit{M-Sym}(Q(n,\theta,Block,\mathcal{M}))$$

end for

6: **end for**

7: **for** $n \in [N]$ **do**

8:

$$Q(n,\theta) \leftarrow \bigcup_{Block\in[f]} \big(Q(n,\theta,Block,\mathcal{I}) \cup Q(n,\theta,Block,\mathcal{M})\big)$$

9: **end for**

Finally, as in FS-PIR Protocol 1 (2.1.3), a random permutation is applied to the file bits to obtain the actual query sets $Q_n^{[\theta]},\ n \in [N]$.

Example 3 *Consider 2 files a and b each of 2 bits both stored on 2 servers S_1 and S_2. To make the notation simpler lets use $W_1 = a,\ W_2 = b$, $U_1 = [a_1, a_2]$ and $U_2 = [b_1, b_2]$.*

The output of Algorithm 4 for $\theta = 1$ i.e retrieving file a is as follows:

S_1	S_2
a_1	$a_2 + b_1$
b_1	

The algorithm starts by querying S_1 for bit a_1, then to obfuscate the server, it also queries it for b_1. At this point it has 1 desired bit and 1 bit of side information. So, it mixes this side information with another desired bit i.e $b_1 + a_2$ and sends this is query to S_2. Clearly, the required file is retrieved.

Similarly the output of the algorithm for $\theta = 2$ is

S_1	S_2
a_1	$a_1 + b_2$
b_1	

Note that in these queries, all the desired bits are either present individually or are mixed with side information which is available from some other server. Thus the desired file can be correctly retrieved using these queries. Secondly, independent of the whether the user wants file a or b, the query structure at the individual servers is same. Thus a random permutation of bits provides needed privacy.

Also, note that this construction retrieves 2 desired file bits by downloading a total of 3 bits, so its rate is 2/3, which matches the capacity for this case ($C = (1+1/2)^{-1} = 2/3$ by eq. (2.1)).

Example 4 *Consider 3 files a, b and c each of 9 bits stored on 3 servers S_1, S_2 and S_3. To make the notation simpler lets use $W_1 = a$, $W_2 = b$ $W_3 = c$, $U_1 = [a_1, a_2, \ldots, a_9]$, $U_2 = [b_1, b_2, \ldots, b_9]$ and $U_3 = [c_1, c_2, \ldots, c_9]$.*

The output of Algorithm 4 for $\theta = 1$ i.e retrieving file a is as follows:

S_1	S_2	S_3
a_1	$a_2 + b_1$	$a_4 + b_1$
b_1	$a_3 + c_1$	$a_5 + c_1$
c_1	$b_2 + c_2$	$b_3 + c_3$
$a_6 + b_2 + c_2$	$a_8 + b_3 + c_3$	$a_9 + b_2 + c_2$
$a_7 + b_3 + c_3$		

The output of Algorithm 4 for $\theta = 2$ i.e retrieving file b is as follows:

S_1	S_2	S_3
a_1	$a_1 + b_2$	$a_1 + b_4$
b_1	$b_3 + c_1$	$b_5 + c_1$
c_1	$a_2 + c_2$	$a_3 + c_3$
$a_2 + b_6 + c_2$	$a_3 + b_8 + c_3$	$a_2 + b_9 + c_2$
$a_3 + b_7 + c_3$		

The output of Algorithm 4 for $\theta = 3$ i.e retrieving file c is as follows:

S_1	S_2	S_3
a_1	$a_1 + c_2$	$a_1 + c_4$
b_1	$b_1 + c_3$	$b_1 + c_5$
c_1	$a_2 + b_2$	$a_3 + b_3$
$a_2 + b_2 + c_6$	$a_3 + b_3 + c_8$	$a_2 + b_2 + c_9$
$a_3 + b_3 + c_7$		

Note that this construction retrieves 9 desired file bits by downloading a total of 13 bits, so its rate is 9/13, which matches the capacity for this case ($C = (1 + 1/3 + 1/9)^{-1} = 9/13$ by eq. (2.1)).

Some key points about the protocol:

1. $Q(n, \theta),\ n \in [N],\ \theta \in [F]$ is a union of F disjoint blocks. Block f contains only f-sums. For any type $i_{[1:f]}$ block f of $Q(n, \theta)$ contains $v(n, f)$ instances of type $i_{[1:f]}$, where $v(1, 1) = 1, v(n, 1) = 0,\ \forall n \in [2 : N]$ and $v(n, f) = \sum_{n' \neq n} v(n', f - 1), \forall f \in [2 : F]$.
2. No $u_i(j), j \in [N^{F-1}]$ variable appears more than once in query set to any given server.
3. The protocol is not symmetric wrt the servers. Also, the load distribution across servers is not uniform as different servers may receive different number of queries.
4. It provably offers the minimum file size amongst all capacity achieving PIR protocols.

Capacity achieving Protocol 3 (FS-PIR protocol 3)

With a slight modification in the definition of retrieval rate, the minimum file size required for achieving capacity can be greatly reduced. In the scenario when there is

more than one way (chosen randomly by the user) to retrieve a file each having a different download cost (in bits), a intuitive way of quantifying retrieval rate will be to take the ratio of file length to the expected download cost i.e. $R = L/\mathbb{E}(D)$. With this as the definition for retrieval rate Tian, Sun and Chen in 2019 [19] proposed a capacity achieving protocol with file size $L = N-1$. Furthermore they showed that amongst the class of all capacity achieving linear codes the above file size is optimal (or the lowest possible). For ease in presentation of the protocol, the authors consider the indexing of the file from 0 to $F-1$, instead of the usual 1 to F. Same is done for indexing of servers as well. The idea behind their scheme is quite simple and as follows:

1. The user first generates a $F-1$ lengthed random key (uniformly random) $\mathcal{F} \in \{0, 1, \ldots, N-1\}^{F-1}$. Say the key is $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_{F-2})$.
2. Say the user wants to retrieve the θ-th file, then he generates queries Q_n by keeping $f-1$ (all other except the θ-th) entries fixed and varying the θ-th entry over all elements of $\{0, 1, 2, \ldots, N-1\}$, i.e $Q_n = (\mathcal{F}_0, \ldots, \mathcal{F}_{\theta-1}, e_n, \mathcal{F}_\theta, \ldots, \mathcal{F}_{F-2})$, where $e_n \in \{0, 1, \ldots, N-1\}\ \forall\ n \in \{0, 1, \ldots, N-1\}$, and $e_i = e_j$ if and only if $i = j$.
3. The idea behind this construction is that each server would return a xor of F bits, each coming from a different file. Now since $F-1$ entries of each query are the same, the responses of servers would be of the form $\Delta \oplus$some bit of W_θ. By varying the θ-th entry over all elements of $\{0, 1, 2, \ldots, N-1\}$, we get responses as $\Delta, \Delta \oplus W_{\theta,1}, \Delta \oplus W_{\theta,2}, \ldots, \Delta \oplus W_{\theta,N-1}$ and hence we would be able to retrieve all bits of desired file.
4. The authors do this one-one mapping from set of e_n's to $\{0, 1, 2, \ldots, N-1\}$ by choosing e_n such that for the n-th server sum of all entries of the query modulo N equals n i.e $\left(\sum_{i=0}^{F-2} \mathcal{F}_i + e_n\right)_N = n, \forall\ n \in \{0, 1, \ldots, N-1\}$.

Finally, as in FS-PIR Protocol 1 and 2, a random permutation is applied to the bits of each file individually. ($\gamma_i : \{0, 1, 2, \ldots N-1\} \to \{0, 1, 2, \ldots N-1\}\ , i\ \in [F]$)

Example 5 *Consider 2 files a and b of 1 bit each stored on 2 servers S_0 and S_1. Suppose the user wants to retrieve file a. First he generates random key $\mathcal{F} \in \{0, 1\}$. If user chooses $\mathcal{F} = 0$, he constructs the queries as $Q_1 = (0, 0)$ and $Q_2 = (1, 0)$ which means no bit is downloaded from S_0 and a is downloaded from S_1, and hence the desired file is retrieved by downloading just 1 bit. If user chooses $\mathcal{F} = 1$, the user constructs the queries as $Q_1 = (1, 1)$ and $Q_2 = (1, 0)$, thus downloading $a + b$ from S_0 and b from S_1 and hence retrieving a by downloading total 2 bits. Since $\mathcal{F}$ is chosen uniformly randomly, $Prob(\mathcal{F} = 0) = Prob(\mathcal{F} = 1) = 0.5$. Thus, $\mathbb{E}(D) = 1.5$. The table below summarizes the queries for retrieving a and b using different keys (e_n bits to the servers*

are highlighted in red and each key $\mathcal{F}$ corresponds to one specific way of retrieving the desired file):

Key	*Retrieving* a *(*$\theta = 0$*)*				*Retrieving* b *(*$\theta = 1$*)*			
	S_0		S_1		S_0		S_1	
	Query	*Resp.*	*Query*	*Resp.*	*Query*	*Resp.*	*Query*	*Resp.*
$\mathcal{F} = 0$	(0,0)	–	(1,0)	a	(0,0)	–	(0,1)	b
$\mathcal{F} = 1$	(1,1)	$a + b$	(0,1)	b	(1,1)	$a + b$	(1,0)	a

Note that to download either of the files, on an average 3/2 bits are downloaded and both files are of size 1 bit. Thus $R = 2/3$ which matches capacity for the system. Another interesting thing to note is that for some key no bits are downloaded from 1 of the servers.

Example 6 *Consider 3 files a, b, c each of 2 bits stored on the three servers S_1, S_2 and S_3. Representation: $a = \{a_1, a_2\}$, $b = \{b_1, b_2\}$ and $c = \{c_1, c_2\}$.The tables below summarize the queries for retrieving a, b and c using different randomly generated keys (e_n bits to the servers are highlighted in red):*

Key	*Retrieving* a *(*$\theta = 0$*)*					
	S_0		S_1		S_2	
	Query	*Resp.*	*Query*	*Resp.*	*Query*	*Resp.*
$\mathcal{F} = (0,0)$	(0,0,0)	–	(1,0,0)	a_1	(2,0,0)	a_2
$\mathcal{F} = (0,1)$	(2,0,1)	$a_2 + c_1$	(0,0,1)	c_1	(1,0,1)	$a_1 + c_1$
$\mathcal{F} = (0,2)$	(1,0,2)	$a_1 + c_2$	(0,0,2)	c_2	(2,0,2)	$a_2 + c_2$
$\mathcal{F} = (1,0)$	(2,1,0)	$a_2 + b_1$	(0,1,0)	b_1	(1,1,0)	$a_1 + b_1$
$\mathcal{F} = (1,1)$	(1,1,1)	$a_1 + b_1 + c_1$	(2,1,1)	$a_2 + b_1 + c_1$	(0,1,1)	$b_1 + c_1$
$\mathcal{F} = (1,2)$	(0,1,2)	$b_1 + c_2$	(1,1,2)	$a_1 + b_1 + c_2$	(2,1,2)	$a_2 + b_1 + c_2$
$\mathcal{F} = (2,0)$	(1,2,0)	$a_1 + b_2$	(2,2,0)	$a_2 + b_2$	(0,2,0)	b_2
$\mathcal{F} = (2,1)$	(0,2,1)	$b_2 + c_1$	(1,2,1)	$a_1 + b_2 + c_1$	(2,2,1)	$a_2 + b_2 + c_1$
$\mathcal{F} = (2,2)$	(2,2,2)	$a_2 + b_2 + c_2$	(0,2,2)	$b_2 + c_2$	(1,2,2)	$a_1 + b_2 + c_2$

Key	*Retrieving b ($\theta = 1$)*					
	S_0		S_1		S_2	
	Query	*Resp.*	*Query*	*Resp.*	*Query*	*Resp.*
$\mathcal{F} = (0,0)$	*(0,0,0)*	–	*(0,1,0)*	b_1	*(0,2,0)*	b_2
$\mathcal{F} = (0,1)$	*(0,2,1)*	$b_2 + c_1$	*(0,0,1)*	c_1	*(0,1,1)*	$b_1 + c_1$
$\mathcal{F} = (0,2)$	*(0,1,2)*	$b_1 + c_2$	*(0,2,2)*	$b_2 + c_2$	*(0,0,2)*	c_2
$\mathcal{F} = (1,0)$	*(1,2,0)*	$a_1 + b_2$	*(2,2,0)*	$a_2 + b_2$	*(0,2,0)*	b_2
$\mathcal{F} = (1,1)$	*(1,1,1)*	$a_1 + b_1 + c_1$	*(1,2,1)*	$a_1 + b_2 + c_1$	*(1,0,1)*	$a_1 + c_1$
$\mathcal{F} = (1,2)$	*(1,0,2)*	$a_1 + c_2$	*(1,1,2)*	$a_1 + b_1 + c_2$	*(1,2,2)*	$a_1 + b_2 + c_2$
$\mathcal{F} = (2,0)$	*(2,1,0)*	$a_2 + b_1$	*(2,2,0)*	$a_2 + b_2$	*(2,0,0)*	a_2
$\mathcal{F} = (2,1)$	*(2,0,1)*	$a_2 + c_1$	*(2,1,1)*	$a_2 + b_1 + c_1$	*(2,2,1)*	$a_2 + b_2 + c_1$
$\mathcal{F} = (2,2)$	*(2,2,2)*	$a_2 + b_2 + c_2$	*(2,0,2)*	$a_2 + c_2$	*(2,1,2)*	$a_2 + b_1 + c_2$

Key	*Retrieving c ($\theta = 2$)*					
	S_0		S_1		S_2	
	Query	*Resp.*	*Query*	*Resp.*	*Query*	*Resp.*
$\mathcal{F} = (0,0)$	*(0,0,0)*	–	*(0,0,1)*	c_1	*(0,0,2)*	c_2
$\mathcal{F} = (0,1)$	*(0,1,2)*	$b_1 + c_2$	*(0,1,0)*	$b_1 + c_2$	*(0,1,1)*	$b_1 + c_2$
$\mathcal{F} = (0,2)$	*(0,2,1)*	$b_2 + c_1$	*(0,2,2)*	$b_2 + c_2$	*(0,2,0)*	b_2
$\mathcal{F} = (1,0)$	*(1,0,2)*	$a_1 + c_2$	*(1,0,0)*	a_1	*(1,0,1)*	$a_1 + c_1$
$\mathcal{F} = (1,1)$	*(1,1,1)*	$a_1 + b_1 + c_1$	*(1,1,2)*	$a_1 + b_1 + c_2$	*(1,1,0)*	$a_1 + b_1$
$\mathcal{F} = (1,2)$	*(1,2,0)*	$a_1 + b_2$	*(1,2,1)*	$a_1 + b_2 + c_1$	*(1,2,2)*	$a_1 + b_2 + c_2$
$\mathcal{F} = (2,0)$	*(2,0,1)*	$a_2 + c_1$	*(2,0,0)*	a_2	*(2,0,2)*	$a_2 + c_2$
$\mathcal{F} = (2,1)$	*(2,1,0)*	$a_2 + b_1$	*(2,1,1)*	$a_2 + b_1 + c_1$	*(2,1,2)*	$a_2 + b_1 + c_2$
$\mathcal{F} = (2,2)$	*(2,2,2)*	$a_2 + b_2 + c_2$	*(2,2,0)*	$a_2 + b_2$	*(2,2,1)*	$a_2 + b_2 + c_1$

In the above tables, each key $\mathcal{F}$ corresponds to one specific way of retrieving the desired file. Note that for retrieving any file, on average $\frac{1}{9} \times 2 + \frac{8}{9} \times 3 = \frac{26}{9}$ are downloaded. Hence $R = \frac{2}{26/9} = \frac{18}{26} = \frac{9}{13}$, which matches the capacity for the system.

2.2 Storage constrained Private Information Retrieval (SC-PIR)

Consider a system of N non-colluding servers storing f independent files. The servers are denoted as S_n where $n \in [N]$. Each file is assumed to be of size L bits. The files are denoted as W_i, where $i \in [F]$.

Assume that the servers have identical storage capacity and every server stores a fixed fraction $\mu \in [\frac{1}{N}, 1]$ of each file. This fraction μ is called the normalized storage capacity of the server. Therefore each server stores μL bits of each file and a total of μLF bits. For simplicity often the case where μ is an integral multiple of $1/N$ is considered, and results can then be extended to the case where t is not an integral multiple of $1/N$. The content of server S_n is denoted by Z_n, for any $n \in [N]$.

The user privately generates index $\theta \in [F]$ and wants to retrieve file W_θ without leaking any information about θ to any of the individual servers. In order to do so, it generates queries $Q_1^{[\theta]}, Q_2^{[\theta]}, \ldots Q_N^{[\theta]}$, where $Q_n^{[\theta]}$ is sent to server S_n.

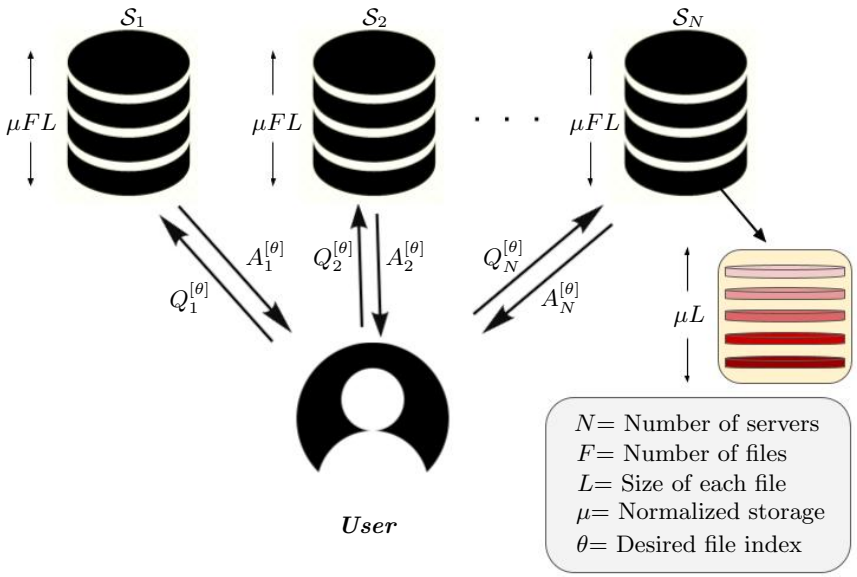


Figure 2.2: SC-PIR setup

Again, as queries are generated without any information about the file contents

$$I(W_1, W_2, \ldots, W_F; Q_1^{[\theta]}, Q_2^{[\theta]}, \ldots, Q_N^{[\theta]}) = 0 \quad \text{for all } \theta \in [F]. \tag{2.4}$$

and server S_n on receiving query $Q_n^{[\theta]}$ responds with answer $A_n^{[\theta]}$ which is a deterministic function of $Q_n^{[\theta]}$ and the contents stored on the server, i.e.,

$$H(A_n^{[\theta]}|Q_n^{[\theta]}, Z_n) = 0, \text{ for all } n \in [N] \text{ and } \theta \in [F]. \tag{2.5}$$

The desired file W_θ is then retrieved from all the answers i.e

$$H(W_\theta|A_1^{[\theta]}, A_2^{[\theta]}, \ldots, A_N^{[\theta]}) = 0 \text{ for all } \theta \in N \tag{2.6}$$

To ensure that the user request is private the following privacy constraint must be satisfied

$$I(\theta; Q_n^{[\theta]}, A_n^{[\theta]}, Z_n)) = 0 \text{ for all } \theta \in [F], n \in [N] \tag{2.7}$$

2.2.1 Capacity

For SC-PIR system the maximum achievable rate or capacity was characterized by Tandon *et al.* in 2018 [15].

Theorem 2 *For a system of F files and N servers, each with normalized storage μ, the optimal download cost per bit $D^*(\mu)$ (or the inverse of PIR capacity) is given by the lower convex hull of the following $(\mu, D^*(\mu))$ pairs, for $t = 1, 2, \ldots, N$:*

$$\left(\mu = \frac{t}{N}, \; D^*(\mu) = \sum_{f=0}^{F-1} \frac{1}{t^F} \right). \tag{2.8}$$

2.2.2 Capacity achieving Protocol 1

Now, we review the SC-PIR capacity achieving protocol proposed by Tandon *et al.* in [15]. Assume that $t \in \mathbb{Z}$. Taking inspiration from coded caching, the authors in [15] divide each file into $\binom{N}{t}$ equally sized sub-files and label them with a unique subset $\mathcal{S} \subseteq [1 : N]$ of size t. Then for each file, each server stores all the sub-files which contain its index. By doing so, two things are achieved

(i) Each server stores $\binom{N-1}{t-1}$ sub-files for each file, thus $\mu = \binom{N-1}{t-1}/\binom{N}{t} = t/N$.

(ii) Each subfile for every file is stored on exactly t servers.

Then to retrieve file W_θ, the user has to retrieve all the subfiles of W_θ. To do so, the capacity achieving FS-PIR protocol 1 (section 2.1.3) is used to retrieve individual subfiles from the set of t servers storing each subfile. By restricting focus to only a particular subfile of all files, this set of t servers can be considered as a FS-PIR system with F files of size $L/\binom{N}{t}$ each.

Since each subfile retrieval happens with a download cost per bit of $\left(1 + 1/N + \cdots + 1/N^{F-1}\right)$, the overall download cost per bit is

$$\begin{aligned} D(\mu) &= \frac{\binom{N}{t} \times \text{Number of bits downloaded to retrieve 1 subfile}}{\binom{N}{t} \times \text{Size of each subfile}} \\ &= \text{Download cost per bit for retrieving 1 subfile} \\ &= 1 + \frac{1}{t} + \frac{1}{t^2} + \cdots + \frac{1}{t^{F-1}} \end{aligned} \tag{2.9}$$

which equals the optimal download cost per bit for the SC-PIR system (eq 2.8).

For the case when $t \neq \mathbb{Z}$, capacity achieving PIR protocol can be constructed by using a technique called memory sharing using two SC-PIR systems with $t \in \mathbb{Z}$.

2.3 Review of Combinatorial Designs

Combinatorial designs appear to be a useful object in designing storage schemes. In this section we review some fundamental combinatorial block designs. We refer the reader to [20] for more details.

Definition 1 *(Design (X, A)). A design is a pair (X, A) such that the following are satisfied:*

D1) X is a set of elements called points.

D2) A is a collection of non-empty subsets of X called blocks.

A design can also be described by a matrix called the incidence matrix. We define this next.

Definition 2 *(Incidence Matrix). Let* (X, A) *be a design where* $X = \{x_1, x_2, \ldots, x_v\}$ *and* $A = \{A_1, A_2, \ldots, A_b\}$. *The incidence matrix of* (X, A) *is the* $v \times b$ *binary matrix* $M_{v\times b} = (M_{i,j})$ *defined by the rule*

$$M_{ij} = \begin{cases} 1 & \text{if } x_i \in A_j \\ 0 & \text{if } x_i \notin A_j \end{cases} \tag{2.10}$$

Next, we define t-designs.

Definition 3 *(t-designs). Let* v, k, λ *and* t *be positive integers such that* $v > k \geq t$. *A* t-(v, k, λ)*-design is a design* (X, A) *such that the following are satisfied:*

1. $|X| = v$
2. *every block contains exactly* k *points, and*
3. *every set of* t *distinct points is contained in exactly* λ *blocks.*

Now, we state some important properties of t-designs (refer [20]).

(i) The number of blocks that contain any i-element set of points is given by

$$\lambda_i = \lambda \binom{v-i}{t-i} \Big/ \binom{k-i}{t-i} \quad , i \in [t] \tag{2.11}$$

The total number of blocks b in a t-design are given by

$$b = \lambda \binom{v}{t} \Big/ \binom{k}{t} \tag{2.12}$$

(ii) Any t-(v, k, λ)-design is also a s-(v, k, λ_s)-design for any s with $1 \leq s \leq t$. (Note that the value of λ_s depends on s.)

The simplest types of t-designs are known as tactical configurations or sometimes referred to as just configurations.

Definition 4 *((v, k, b, r)-configuration). A* (v, k, b, r)*-configuration, also known as tactical configuration is a 1-design with* v *points and* b *blocks, each containing* k *points. Also, each point is contained in the same number of blocks* r*, called the repetition number.*

The necessary and sufficient condition for existence of a (v, k, b, r)-configuration is

$$bk = vr \tag{2.13}$$

Note that a (v, k, b, r)-configuration is a 1-(v, k, r)-design with $b = \frac{vr}{k}$.

The incidence matrix $M_{v\times b}$ of a (v, k, b, r)-configuration has some useful structure:

(i) Since each point is contained in r blocks, each row of M has exactly r ones.

(ii) Since each block contains k points, each column of M has exactly k ones.

Now, we propose a simple way of generating incidence matrices for tactical configurations.

Lemma 3 *For a (v, k, b, r)-configuration, one incidence matrix $M_{v\times b}$ can be generated as:*

1. *Fill the first row with 1's in the first r places, and 0 in the remaining positions.*
2. *For all rows $\in [2 : v]$, construct the next row by circularly shifting the previous row by r places.*

Proof: Clearly, each row of $M_{v\times b}$ has r ones. Now to prove that the above generated incidence matrix corresponds to a tactical configuration, we need to show that each column of the generated matrix has k ones. We prove the same through a construction. Note that, originally, each row of $M_{v\times b}$ is a length b vector.

Construct a $v \times vr$ matrix M' by filling ones so that row i has ones in columns $[(i-1)r + 1 : ir]$ and zeros elsewhere. So the first row has ones only in column $[1 : r]$. Second row has ones in columns $[r + 1 : 2r]$ and so on.

Next, construct matrices $M'_1, M'_2, \ldots, M'_k$ by considering the groups of b columns orderwise from M'. Now, given any $i \in [v]$ and $j \in [b]$, there is at maximum only one of these matrices can have a 1 in the (i, j)-th entry, i.e for any $i \in [v], j \in [b]$ if $M'_l(i, j) = 1$, then $M'_{l'}(i, j) = 0$ for all $l' \in [k]/l$, where $M'_l(i, j)$ represents the j-th entry in the i-th row of M_l'. This happens because each row of M' has exactly r ones and $r <= b$.

Finally, note that original matrix $M_{v\times b}$ can be obtained by taking a entry-wise union of the k constructed matrices ${M'}_i, i \in [k]$ i.e.

$$M_{v\times b} = \bigcup_{i\in[k]} {M'}_i$$

Since each column of ${M'}_i$'s has exactly one 1 and no two ${M'}_i$'s have 1 in the same location, each column of $M_{v\times b}$ has k ones in each column. Thus the incidence matrix $M_{v\times b}$ generated as per *Lemma 3* corresponds to a tactical configuration. □

A simple example is provided below to understand the proposed construction and construction involved in the proof easily.

Example 7 *Lets say we need to construct a* $(3, 2, 3, 2)$ *configuration. Incidence matrix for a configuration with these parameters can be obtained by filling 1's in the first two entries in the first row and circularly shifting the first row by 2 places to obtain the second row and eventually circularly shifting second by 2 entries row to obtain the third row.*

$$M_{3\times3} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Another way of obtaining the same matrix is by generating the matrix M' *as proposed in the proof of Lemma 3. So we generate a* 3×6 *matrix* M' *by filling first two entries as 1 in the first row, third and fourth entry in second row as 1's and fifth and sixth entry in third row as 1's and all other entries as zero.*

$$M' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Now this matrix has to be divided into $k = 2$ *smaller matrices with* $b = 3$ *columns each. So matrix* M'_1 *consists of the first* 3 *columns and matrix* M'_2 *consists of the next* 3 *columns.*

$$M_1' = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad M_2' = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Finally, the matrix $M_{3\times 3}$ can be obtained by taking entry-wise union of the matrices $M_1's$ and $M_2's$.

Next, we look at another useful class of block designs known as Balanced Incomplete Block Designs or BIBDs.

Definition 5 *((v, k, λ, b, r)-BIBD.) A 2-design with v points, b blocks, each containing k points, such that each point is in exactly r blocks and any given pair of points is in exactly λ blocks is called a (v, k, λ, b, r)-Balanced Incomplete Block Design (BIBD).*

Note: For a (v, k, λ, b, r)-BIBD,

$$bk = vr \quad \& \quad r(k-1) = \lambda(v-1) \tag{2.14}$$

Next, we look at some special classes of BIBD's.

Definition 6 *(Symmetric BIBD.) A BIBD with $b = v$ is called a symmetric BIBD.*

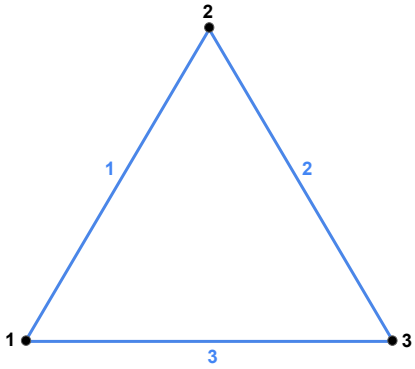
A useful property of symmetric BIBD's is that any 2 blocks contain exactly λ common points.

Definition 7 *(Projective Plane.) An $(N^2 + N + 1, N + 1, 1, N^2 + N + 1, N + 1)$-BIBD with $N \geq 2$ is called a projective plane of order N.*

Definition 8 *(Affine Plane.) An $(N^2, N, 1, N^2 + N, N + 1)$-BIBD with $N \geq 2$ is called a affine plane of order N.*

2.3.1 Examples

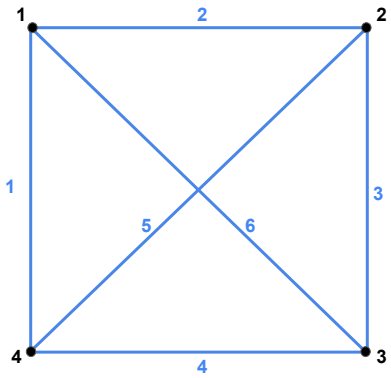
(a) (3,2,3,2)-configuration.



Considering the sides of a triangle as blocks and vertices as points, we get a (3,2,3,2)-configuration with incidence matrix

$$M_{3\times 3} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

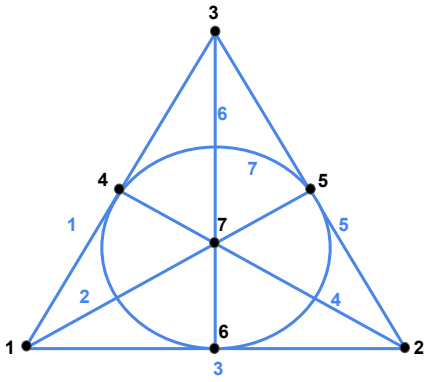
(b) (4,2,1,6,3)-BIBD.



Considering vertices as points and lines as blocks in the figure above, we get a (4,2,1,6,3)-BIBD with incidence matrix

$$M_{4\times 6} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(c) (Fano Plane)



The fano plane is a (7,3,1,7,3)-symmetric BIBD with incidence matrix

$$M_{7\times 7} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

CHAPTER 3

HOMOGENEOUS STORAGE CONSTRAINED PIR PROTOCOL

In this Chapter we propose a novel PIR protocol for uncoded systems storing a fraction of each file. While the storage capacity is identical for the servers, they need not store the same content. The proposed PIR protocol has two main components: a storage scheme and a retrieval scheme. The storage scheme is based on combinatorial designs, while the retrieval scheme works by projecting the given system to multiple instances of smaller systems with replicated servers having full storage capacity. It uses the retrieval scheme of FS-PIR systems for these smaller instances. Our scheme is flexible and can easily accommodate a range of system parameters.

3.1 Setup for PIR

We consider a system of N non-colluding servers storing F independent files. The servers are denoted as S_n where $n \in [N]$. Each file is assumed to be of size L bits. The files are denoted as W_i, where $i \in [F]$.

As the files are independent we have $H(W_1, W_2, \ldots, W_F) = \sum_{i=1}^{n} H(W_i) = FL$, where $H(\cdot)$ denotes the entropy function.

We assume that the servers have identical storage capacity and every server stores a fixed fraction $\mu \in [\frac{1}{N}, 1]$ of every file. This fraction μ is called the normalized storage capacity of the server. Therefore each server stores μL bits of each file and a total of μLF bits. For simplicity we consider the case where μ is an integral multiple of $1/N$. PIR protocols can be extended to other values of μ by memory sharing.

We denote the content of server S_n by Z_n, for any $n \in [N]$. A system of N non-colluding servers, storing F files and having a normalized storage of μ is referred to as a μ-(F, N) system, see Fig. 3.1 for an illustration.

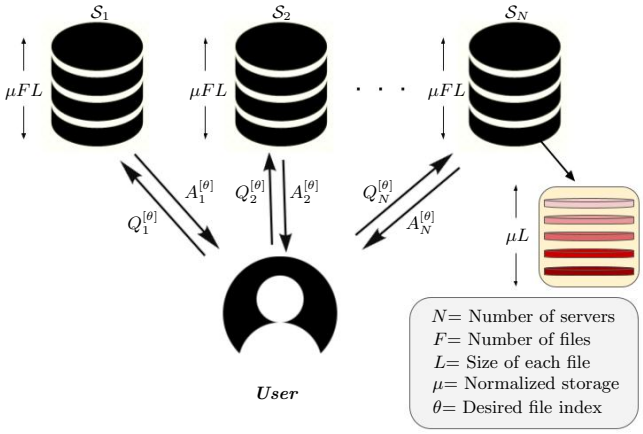


Figure 3.1: A μ-(F, N) system with N servers storing F independent files. The user generates an index $\theta \in [F]$ and a query $Q_n^{[\theta]}$ to each of the servers. The server responds with an answer denoted as $A_n^{[\theta]}$. The user recovers the requested file from all the answers.

The PIR problem is described as follows. A user privately generates an index $\theta \in [F]$ and wishes to retrieve file W_θ, without revealing any information about θ to any of the individual servers. This index θ is generated independent of the file content or the server contents. To retrieve W_θ, the user generates N queries $Q_1^{[\theta]}, Q_2^{[\theta]}, \ldots, Q_N^{[\theta]}$, where $Q_n^{[\theta]}$ is sent to server S_n, $n \in [N]$. Upon receiving these queries, each server S_n responds with an answer $A_n^{[\theta]}$, which is a deterministic function of the query and the contents stored on the server, i.e.,

$$H(A_n^{[\theta]}|Q_n^{[\theta]}, Z_n) = 0, \text{ for all } n \in [N] \text{ and } \theta \in [F]. \tag{3.1}$$

In this work, we assume that the user requests a linear combination of bits stored on the server as the query. The server responds by returning the value of the requested linear combination.

A PIR protocol must satisfy the following constraints:

$$H(W_\theta|A_1^{[\theta]}, \ldots, A_N^{[\theta]}, Q_1^{[\theta]}, \ldots, Q_N^{[\theta]}) = 0, \text{ for all } \theta \in [F]. \tag{3.2a}$$

$$I(\theta; Q_n^{[\theta]}, A_n^{[\theta]}, Z_n) = 0, \quad \forall n \in [N], \quad \text{for all } \theta \in [F] \tag{3.2b}$$

The first condition eq (3.2a) requires that from the answers obtained form all the servers, the user must be able to correctly retrieve the desired file W_θ.

The second condition eq (3.2b) ensures privacy of request i.e. the user does not reveal any information about the desired file index to any of the individual servers.

For any PIR protocol, the total size of all queries sent to the servers constitutes the upload cost and the amount of downloaded data constitutes the download cost. The total cost of PIR is the sum of upload cost and download cost, with the later usually being the dominant contributor.

3.2 Storage Scheme

Our storage scheme is based on combinatorial designs. Given a system with N servers and $\mu = t/N$, where $t \in \mathbb{Z}$, we first find a (v, k, b, r) configuration with $b = N$ and $r = t$. Next, we divide each file into v subfiles of equal size $\ell = L/v$ and store them in accordance to the incidence matrix of the chosen configuration. More formally, the storage scheme is given in the algorithm below.

Algorithm 5 Storage scheme for proposed PIR protocol
Input: N, t
Output: Z_n, for all $n \in [N]$

1: Construct a (v, k, b, r)-configuration with $b = N$ and $r = t$. Denote the $v \times b$ incidence matrix of this configuration by M whose (i, j)th entry is given by M_{ij}.
2: Divide each file W_i, for $i \in [F]$, into v subfiles. Denote them as W_{ij} where $j \in [v]$. Each subfile W_{ij} is of size L/v bits.
3: For all $i \in [F]$ and $j \in [v]$ store the subfile W_{ij} on server S_n if $M_{jn} = 1$.

Note that each server stores k out of the v subfiles for each file (since each column of M has exactly k ones). Thus, the normalized storage of each server is $\mu = k/v = r/b = t/N$.

3.3 Retrieval Scheme

Let $S(W_{ij})$ be the subset of servers that store the subfile W_{ij} for a fixed i, j where $i \in [F], j \in [v]$. Each subfile W_{ij} is present on exactly t servers (since each row of M has exactly $r = t$ ones). In other words, $|S(W_{ij})| = t$.

For any fixed $j \in [v]$, we denote the part of the Z_n containing the j^{th} subfiles W_{ij} as $Z_n^{[j]}$.

Algorithm 6 Retrieval scheme for proposed PIR protocol
Input: θ
Output: W_θ

1: For all $j \in [v]$, consider the system $S(W_{\theta j})$, which is a 1-(F, t) system storing W_{ij}, $i \in [F]$.
2: Using the FS-PIR Protocol 1 (or 2,3), generate queries $Q_n^{[\theta j]}$, for all $S_n \in S(W_{\theta j})$.
3: Using the answers $A_n^{[\theta j]}$ retrieve all bits of desired file $W_{\theta j}$

3.4 Correctness and Privacy

Suppose the user wants to retrieve the file W_θ. To retrieve file W_θ, the user needs to retrieve all the subfiles $W_{\theta j}$ for $j \in [v]$. We can break it down into v instances of retrieval of the subfiles of W_θ. The retrieval of each subfile $W_{\theta j}$ can be treated as a separate instance of a PIR retrieval problem.

Theorem 4 (Designs to PIR protocols) *Consider a μ-(F, b) system generated from a (v, k, b, r)-tactical configuration. Then with respect to the subfiles W_{ij}, where $i \in [F]$, the servers $S(W_{\theta j})$ form a 1-(F, r) system. For file size $L = v \times r^F$, the proposed protocol consisting of Algorithms 5 and using FS-PIR protocol 1 in Algorithm 6 is correct, private and achieves capacity.*

Proof: In the μ-(F, b) system the subfile $W_{\theta j}$ is present on r servers in the set $S(W_{\theta j})$. Furthermore, each of the servers in $S(W_{\theta j})$ contains all the subfiles W_{ij} for all $i \in [F]$. Therefore, with respect to the subfiles W_{ij}, $i \in [F]$, the servers $S(W_{\theta j})$ form a 1-(F, r) system.

For recovering $W_{\theta j}$, we only restrict our attention to the 1-(F, r) system obtained by restricting to the servers $S(W_{\theta j})$. Subfile $W_{\theta j}$ can be retrieved privately from the above system by using the **FS-PIR protocol 1**. Since $W_{\theta j}$ can be recovered for all $j \in [v]$ from the 1-(F, r) systems formed by $S(W_{\theta j})$, we are able to recover the file W_θ. Therefore, eq (3.2a) is satisfied.

We note two properties of the proposed protocol that we need to prove the privacy constraint.

P1) If we interchange all bits of file W_1 with W_i where $i \in [F]$ in the queries to all the servers, we would retrieve file W_i instead of W_1. This is because the storage is symmetric with respect to all the files.

P2) Any permutation of bits of $W_{ij},\ i \in [F], j \in [v]$ applied on all the queries to all the servers does not affect the retrieval process, since permutation is an invertible operation. This is equivalent to choosing a different permutation in the FS-PIR protocol 1.

Next to show privacy of recovery process, we introduce the following notation. Suppose to recover the subfile $W_{\theta j}$, the protocol for 1-(F, r) system storing $W_{\theta j}$ generates queries $Q_n^{[\theta j]}$, to send to server S_n, wherein $S_n \in S(W_{\theta j})$. Then denote the responses of S_n as $A_n^{[\theta j]}$, $S_n \in S(W_{\theta j})$. For S_n such that $S_n \notin S(W_{\theta j})$, we define $Q_n^{[\theta j]} = \emptyset$. We can combine all the queries sent by the user to recover W_θ as

$$Q_n^{[\theta]} = \bigcup_{j \in [v]} Q_n^{[\theta j]} \tag{3.3}$$

Likewise, we can combine all the responses from server S_n as

$$A_n^{[\theta]} = \bigcup_{j \in [v]} A_n^{[\theta j]} \tag{3.4}$$

To show that the protocol is private, we need to show eq (3.2b). Since the subfile $W_{\theta j}$ can be recovered privately, none of the servers can infer anything about θ given the queries, answers and stored content. Therefore we have

$$I(\theta; Q_n^{[\theta j]}, A_n^{[\theta j]}, Z_n^{[j]}) = 0 \text{ for all } S_n \in S(W_{\theta j}) \tag{3.5}$$

From this it follows that $I(\theta; Q_n^{[\theta j]}) = 0$ for all $S_n \in S(W_{\theta j})$. If the subfile $W_{\theta j}$ is not present on S_n, then $Q_n^{[\theta j]}$ is a null query and once again we have $I(\theta; Q_n^{[\theta j]}) = 0$.

$$I(\theta; Q_n^{[\theta j]}) = 0 \text{ for all } n \in [b] \tag{3.6}$$

Now, for any server S_n, $n \in [b]$

$$I(\theta; Q_n^{[\theta]}, A_n^{[\theta]}, Z_n) = I(\theta; Q_n^{[\theta]}, Z_n) \tag{3.7}$$

since $A_n^{[\theta]}$ is a deterministic function of $Q_n^{[\theta]}$ and Z_n. Therefore, using eq (3.3) we can write

$$\begin{aligned} I(\theta; Q_n^{[\theta]}, A_n^{[\theta]}, Z_n) &= I(\theta; \underset{j\in[v]}{\cup} Q_n^{[\theta j]}) + I(\theta; Z_n | Q_n^{[\theta]}) \\ &\overset{(a)}{=} I(\theta; \underset{j\in[v]}{\cup} Q_n^{[\theta j]}) + H(Z_n|Q_n^{[\theta]}) - H(Z_n|\theta, Q_n^{[\theta]}) \\ &\overset{(b)}{=} I(\theta; \underset{j\in[v]}{\cup} Q_n^{[\theta j]}) + H(Z_n) - H(Z_n) = I(\theta; \underset{j\in[v]}{\cup} Q_n^{[\theta j]}) \end{aligned}$$

The last equality (b) follows from the fact that θ and $Q_n^{[\theta]}$ are user generated quantities (generated without any communication to the server) and hence cannot contain any information about the server contents Z_n.

It remains to show that $I(\theta; \underset{j\in[v]}{\cup} Q_n^{[\theta j]}) = 0$.

Without loss of generality, assume that $\theta = 1$ and $n = 1$ and set $q := Q_1^{[1]}$.

We will show that, fixing q to server S_1 the user can retrieve not only file W_1, but it can actually retrieve any file $W_i,\ i \in [F]$ by appropriately altering the queries to other servers. Also, we show that the probability of recovering any file $W_i,\ i \in [F]$, given S_1 receives query q, is the same. Showing these will establish that $I(\theta; Q_n^{[\theta]}) = 0,\ n \in [N]$. We need the following properties of the FS-PIR protocol 1, see section 2.1.3

FSP1) There are exactly r^{F-1} bits for each subfile W_{ij}, involved in each query block. In other words, r^{F-1} bits of W_{ij}, $\forall\ i \in [F]$ are involved in the j^{th} query block to $S_n \in S(W_{ij})$.

FSP2) Any bit appears atmost once in the queries sent to a particular server, i.e. a bit involved in any query to S_n doesnot appear in any other query to S_n.

FSP3) The query structure is symmetric with respect to any file. In other words, the linear combinations are similar and only differ in the actual bits forming the queries.

To alter the queries to other servers so that we can recover a different file we proceed as follows.

i) Interchange all bits of file W_1 with W_i in the queries to all the servers. This gives a (new) set of queries to be sent to the servers using which we can retrieve W_i.

ii) Let q_{new} be the query to S_1 in the new set of queries. Next, we try to make all queries in the query sets q and q_{new} same. Recall that by FSP1) the same number of bits of W_{ij} occur in any query block to a server and by FSP3) all query blocks have the same structure. Therefore, by permuting the bits of W_{ij} we can map q_{new} to q uniquely with respect to the variables of q and q_{new}. This does not affect the recovery of the file W_i.

This establishes that with respect to the queries of S_1, the user can modify the queries to other servers so that any file $i \in [F]$ can be recovered. Furthermore, the variables which are not part of q and q_{new} can also be permuted without affecting the recovery. Since the number of variables not involved in these queries are same for all files, from the point of view of the server there is equal uncertainty as to which file was requested by the user. These permutations exhaust all the possibilities of queries consistent with FS-PIR Protocol 1. Therefore, eq (3.2b) is also satisfied and the proposed protocol is private.

Now the size of the subfile is $L/v = r^F$, and the rate of the PIR protocol for the 1-(F, r) system is $R = (\sum_{f=0}^{F-1} r^{-f})^{-1}$. Then to recover each of the subfiles $W_{\theta j}$ for a given index θ we require to download L/vR bits. For recovery of the entire file we need to download L/R bits. Thus the rate of the proposed PIR protocol for the μ-(F, b) system is $L/(L/R) = R = (\sum_{f=0}^{F-1} r^{-f})^{-1}$ which coincides with the capacity of the μ-(F, b) system when $L = v \times r^f$ (as per eq (2.8)).

Remark 1 *If for a* 1-(F, r) *system any FS-PIR protocol achieves a rate* R*, then the rate achieved for the SC-PIR protocol consisting of Algorithm 5 and using that given FS-PIR protocol in Algorithm 6 is also* R*. This can be seen as follows: Suppose, we represent rate and download cost of subfile from* 1-(F, r) *by* R_{fs} *and* D_{fs} *respectively. Then* $R_{fs} = \frac{L/v}{D_{fs}}$*. For the SC system* $D_{total} = vD_{fs}$*, so* $R_{fs} = \frac{L}{D} = \frac{L}{vD_{fs}} = R_{fs}$*. Also note that since the capacity of a* 1-(F, r) *system coincided with the capacity of a* $\mu = \frac{r}{N}$-(F, N) *system, if the FS-PIR protocol is capacity achieving for the* 1-(F, r) *system, the SC-PIR protocol will be capacity achieving for the* $\mu = \frac{r}{N}$-(F, N) *system.*

□

Corollary 5 (Designs to PIR protocols) *Consider a μ-(F, b) system generated from a (v, k, b, r)-tactical configuration. Then with respect to the subfiles W_{ij}, where $i \in [F]$, the servers $S(W_{\theta j})$ form a 1-(F, r) system. For file size $L = v \times r^{F-1}$, the proposed protocol consisting of Algorithms 5 and using FS-PIR protocol 2 in Algorithm 6 is correct, private and achieves capacity.*

Proof: Using **FS-PIR protocol 2** in Algorithm 6 allows us to achieve capacity for file sizes in multiples of $L = v \times r^{F-1}$. The proof for correctness and privacy follow from the same arguments as in the proof of Theorem 4. This is because, the properties of underlying full storage protocol FSP1,FSP2 and FSP3 used, also hold for **FS-PIR protocol 2**, with a minute difference that instead of r^{F-1}, there are now r^{F-2} bits in FSP1. As long as the number of bits are same for all subfile, the exact number doesn't matter, it just has to be the same for all $W_{ij},\ i \in [F]$.

Regarding, the capacity, it can be noted in the proof of Remark 1 that if the FS-PIR protocol for retrieving individual subfiles achieves FS-PIR capacity, then the proposed protocol in Algorithms 5 and 6 achieves SC-PIR capacity. □

Corollary 6 (Designs to PIR protocols) *Consider a μ-(F, b) system generated from a (v, k, b, r)-tactical configuration. Then with respect to the subfiles W_{ij}, where $i \in [F]$, the servers $S(W_{\theta j})$ form a 1-(F, r) system. For file size $L = v \times (r-1)$, the proposed protocol consisting of Algorithms 5 and using FS-PIR protocol 3 in Algorithm 6 is correct, private and achieves capacity.*

Proof: This result is established by using **FS-PIR Protocol 3** for retrieval of individual subfiles in 6.

Again, proof for correctness and privacy follow from the same arguments as in the proof of Theorem 4. This is because, the properties of underlying full storage protocol FSP1,FSP2 and FSP3 used, also hold for **FS-PIR protocol 3**, with a modification that instead of r^{F-1}, there is now 1 bit (out of the r bits of a subfile, where the 0-th bit is a dummy bit and actual subfile size is $r-1$ bits) of each subfile W_{ij} in FSP1. But again, since the number of bits are same for any subfile, the exact number doesn't matter, it just has to be the same for all $W_{ij},\ i \in [F]$.

Regarding, the capacity, it can be again noted in the proof of Remark 1 that if the FS-PIR protocol for retrieving individual subfiles achieves FS-PIR capacity, then the proposed protocol in Algorithms 5 and 6 achieves SC-PIR capacity.

□

3.5 Examples

Example 8 *Consider a* $\frac{2}{3}$-$(2,3)$ *system i.e., a system with 3 servers,* $\mu = 2/3$ *and 2 files* a *and* b*, each 12 bits long. We shall use the* $(3,2,3,2)$*-configuration with the incidence matrix* M *given below for the storage scheme. So, we divide each file into 3 subfiles (*$a \to a_1, a_2, a_3$*,* $b \to b_1, b_2, b_3$*), each 4 bits long. Let* a_{ij} *and* b_{ij}*,* $i \in [3], j \in [4]$ *represent the* j^{th} *bit of subfile* a_i *and* b_i *respectively. Store the subfiles on the servers as follows:*

$$M = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \qquad \begin{array}{ccc} S_1 & S_2 & S_3 \\ \hline a_1 & a_1 & a_2 \\ b_1 & b_1 & b_2 \\ a_3 & a_2 & a_3 \\ b_3 & b_2 & b_3 \end{array} \tag{3.8}$$

This system can be projected onto the following reduced 1-$(2,2)$ *systems.*

S_1	S_2
a_1	a_1
b_1	b_1

$S(W_{\theta 1})$

S_2	S_3
a_2	a_2
b_2	b_2

$S(W_{\theta 2})$

S_1	S_3
a_3	a_3
b_3	b_3

$S(W_{\theta 3})$

Suppose the user wishes to retrieve file a*. Subfile* a_1 *is retrieved from system* $S(W_{\theta 1})$ *using the FS-PIR protocol 1 as follows: We start by querying server* S_1 *for bit* a_{11}*. Now to obfuscate* S_1*, we also demand bit* b_{11} *from it. Similarly, we query server* S_2 *for bits* a_{12} *and* b_{12}*. At this point we have 2 desired bits and 2 undesired bits. Now, we query the servers for linear combination of a unknown desired bit and known undesired bit. So, we query* S_1 *for* $a_{13} + b_{12}$ *and* S_2 *for* $a_{14} + b_{11}$*. Clearly, we can retrieve all the desired bits of subfile* a_1 *using these queries. Privacy can be maintained by using a random*

permutation of bits of a_1 and b_1 instead of using them in the original order. Subfiles a_2 and a_3 can be retrieved in a similar manner. The queries to be sent to servers S_1, S_2 and S_3 for retrieval of file a are summarised as follows:

S_1	S_2	S_3
a_{11}	a_{12}	a_{21}
b_{11}	b_{12}	b_{21}
$a_{13} + b_{12}$	$a_{14} + b_{11}$	$a_{23} + b_{22}$
a_{31}	a_{22}	a_{32}
b_{31}	b_{22}	b_{32}
$a_{33} + b_{32}$	$a_{24} + b_{21}$	$a_{34} + b_{31}$

Suppose the queries for S_2 and S_3 were {a_{13}, b_{13}, $a_{11} + b_{14}$, a_{23}, b_{23}, $a_{21} + b_{24}$ }, { a_{21}, b_{21}, $a_{23} + b_{22}$, a_{33}, b_{33}, $a_{31} + b_{34}$ } then file b would be retrieved for the same set of queries for S_1. We can design similar queries for recovering file c. Therefore the queries to the servers do not leak information about the file requested.

Similarly, file b can be retrieved by using the following queries:

S_1	S_2	S_3
a_{11}	a_{12}	a_{21}
b_{11}	b_{12}	b_{21}
$a_{12} + b_{13}$	$a_{11} + b_{14}$	$a_{22} + b_{23}$
a_{31}	a_{22}	a_{32}
b_{31}	b_{22}	b_{32}
$a_{32} + b_{33}$	$a_{21} + b_{24}$	$a_{31} + b_{34}$

Finally, note that the capacity for the given $\frac{2}{3}$-(2,3) system is $R^*(\mu) = \left(1+1/t\right)^{-1} = \left(1 + 1/2\right)^{-1} = 2/3$. The rate of the above PIR scheme is $R(\mu) = 12/18 = 2/3$, which matches the capacity of the given system.

Example 9 *Consider a $\frac{1}{2}$-$(2, 6)$ system i.e., a system with 6 servers, $\mu = 1/2$ and 2 files a and b, each 36 bits long.*

We shall use the (4,2,1,6,3) BIBD with the incidence matrix M given below. Denote the files as 'a' and 'b'. Divide both files into $v = 4$ subfiles, each 9 bit long.

$$a \rightarrow a_1, a_2, a_3, a_4 \quad b \rightarrow b_1, b_2, b_3, b_4$$

Let a_{ij} and b_{ij}, $i \in [3], j \in [4]$ represent the j^{th} bit of subfile a_i and b_i respectively.

We store these on servers $S_1, S_2, \ldots S_6$ according to the storage scheme as:

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{array}{|cccccc|} \hline S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \\ \hline a_1 & a_2 & a_3 & a_1 & a_1 & a_2 \\ a_2 & a_4 & a_4 & a_3 & a_4 & a_3 \\ b_1 & b_2 & b_3 & b_1 & b_1 & b_2 \\ b_2 & b_4 & b_4 & b_3 & b_4 & b_3 \\ \hline \end{array} \tag{3.9}$$

File a can be retrieved using FS-PIR protocol 1 by using the following query sets sent to different servers:

S_1	S_2	S_3	S_4	S_5	S_6
a_{11}	a_{22}	a_{31}	a_{12}	a_{13}	a_{23}
b_{11}	b_{22}	b_{31}	b_{12}	b_{13}	b_{23}
$a_{14} + b_{12}$	$a_{26} + b_{21}$	$a_{34} + b_{32}$	$a_{16} + b_{11}$	$a_{18} + b_{11}$	$a_{28} + b_{21}$
$a_{15} + b_{13}$	$a_{27} + b_{23}$	$a_{35} + b_{33}$	$a_{17} + b_{13}$	$a_{19} + b_{12}$	$a_{29} + b_{22}$
a_{21}	a_{41}	a_{42}	a_{32}	a_{43}	a_{33}
b_{21}	b_{41}	b_{42}	b_{32}	b_{43}	b_{33}
$a_{24} + b_{22}$	$a_{44} + b_{42}$	$a_{46} + b_{41}$	$a_{36} + b_{31}$	$a_{48} + b_{41}$	$a_{38} + b_{31}$
$a_{25} + b_{23}$	$a_{45} + b_{43}$	$a_{47} + b_{43}$	$a_{37} + b_{33}$	$a_{49} + b_{42}$	$a_{39} + b_{32}$

Similarly, b can be retrieved using the following queries:

S_1	S_2	S_3	S_4	S_5	S_6
a_{11}	a_{22}	a_{31}	a_{12}	a_{13}	a_{23}
b_{11}	b_{22}	b_{31}	b_{12}	b_{13}	b_{23}
$a_{12}+b_{14}$	$a_{21}+b_{26}$	$a_{32}+b_{34}$	$a_{11}+b_{16}$	$a_{11}+b_{18}$	$a_{21}+b_{28}$
$a_{13}+b_{15}$	$a_{23}+b_{27}$	$a_{33}+b_{35}$	$a_{13}+b_{17}$	$a_{12}+b_{19}$	$a_{22}+b_{29}$
a_{21}	a_{41}	a_{42}	a_{32}	a_{43}	a_{33}
b_{21}	b_{41}	b_{42}	b_{32}	b_{43}	b_{33}
$a_{22}+b_{24}$	$a_{42}+b_{44}$	$a_{41}+b_{46}$	$a_{31}+b_{36}$	$a_{41}+b_{48}$	$a_{31}+b_{38}$
$a_{23}+b_{25}$	$a_{43}+b_{45}$	$a_{43}+b_{47}$	$a_{33}+b_{37}$	$a_{42}+b_{49}$	$a_{32}+b_{39}$

Note that the retrieval rate here is $R =$ Size of file retrieved/total number of bits downloaded $= 36/48 = 3/4$ which matches the capacity $C = (1+1/3)^{-1} = 3/4$ for the system.

Example 10 *Consider a $\frac{3}{7}$-$(2,7)$ system i.e., a system with 7 servers, $\mu = 3/7$ and 2 files a and b, each 21 bits long.*

We shall use the well known Fano plane as the required tactical configuration. Denote the files as 'a' and 'b'. Divide both files into $v = 7$ subfiles, each 3 bit long.

$$a \to a_1, a_2, \ldots, a_7 \quad b \to b_1, b_2, \ldots, b_7$$

Let a_{ij} and b_{ij}, $i \in [7], j \in [3]$ represent the j^{th} bit of subfile a_i and b_i respectively.

We store these on servers $S_1, S_2, \ldots S_7$ according to the storage scheme as:

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{array}{|ccccccc|} \hline S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 \\ \hline a_1 & a_3 & a_1 & a_1 & a_3 & a_2 & a_2 \\ a_2 & a_4 & a_5 & a_4 & a_6 & a_5 & a_4 \\ a_3 & a_5 & a_6 & a_7 & a_7 & a_7 & a_6 \\ b_1 & b_3 & b_1 & b_1 & b_3 & b_2 & b_2 \\ b_2 & b_4 & b_5 & b_4 & b_6 & b_5 & b_4 \\ b_3 & b_5 & b_6 & b_7 & b_7 & b_7 & b_6 \\ \hline \end{array} \tag{3.10}$$

Now, suppose that the user wants to retrieve file a. It can do so, by generating the following set of queries in accordance to the FS-PIR protocol 2:

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_{11}	a_{31}	$a_{12}+b_{11}$	$a_{13}+b_{11}$	$a_{32}+b_{31}$	a_{21}	$a_{22}+b_{21}$
b_{11}	b_{31}	a_{51}	a_{41}	$a_{63}+b_{61}$	b_{21}	$a_{42}+b_{41}$
$a_{23}+b_{21}$	$a_{43}+b_{41}$	b_{51}	b_{41}	a_{71}	$a_{52}+b_{51}$	a_{61}
$a_{33}+b_{31}$	$a_{53}+b_{51}$	$a_{62}+b_{61}$	$a_{73}+b_{71}$	b_{71}	$a_{72}+b_{71}$	b_{61}

Again, note that here retrieval rate $R = 21/28 = 3/4$ *which matches the capacity* $C = (1 + 1/3)^{-1} = 3/4$ *for the given system.*

Example 11 *Consider a* $\frac{2}{5}$*-*$(2, 10)$ *system with files* a *and* b*, and* $t = \mu N = 4$*. Assume that we want to retrieve file* a*.*

We will use a $(5, 2, 1, 10, 4)$*-BIBD for the storage scheme. So we divide each file into 5 subfiles and store them on the servers as follows:*

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
a_1	a_3	a_3	a_4	a_1	a_2	a_2	a_1	a_1	a_2
a_5	a_5	a_4	a_5	a_4	a_5	a_4	a_3	a_2	a_3
b_1	b_3	b_3	b_4	b_1	b_2	b_2	b_1	b_1	b_2
b_5	b_5	b_4	b_5	b_4	b_5	b_4	b_3	b_2	b_3

We use either FS-PIR protocol 1 or 2 to retrieve the individual subfiles. With FS-PIR protocol 1, the retrieval process looks like:

Assume each subfile is of 16 bits, which means the file is of size 80 bits.

S_1	S_2	S_3	S_4	S_5
a_{11}	a_{31}	a_{32}	a_{42}	a_{12}
b_{11}	b_{31}	b_{32}	b_{42}	b_{12}
a_{51}	a_{52}	a_{41}	a_{53}	a_{43}
b_{51}	b_{52}	b_{41}	b_{53}	b_{43}
$a_{15} + b_{12}$	$a_{35} + b_{32}$	$a_{38} + b_{31}$	$a_{48} + b_{41}$	$a_{18} + b_{11}$
$a_{16} + b_{13}$	$a_{36} + b_{33}$	$a_{39} + b_{33}$	$a_{49} + b_{43}$	$a_{19} + b_{13}$
$a_{17} + b_{14}$	$a_{37} + b_{34}$	$a_{3\,10} + b_{34}$	$a_{4\,10} + b_{44}$	$a_{1\,10} + b_{14}$
$a_{55} + b_{52}$	$a_{58} + b_{51}$	$a_{45} + b_{42}$	$a_{5\,11} + b_{51}$	$a_{4\,11} + b_{41}$
$a_{56} + b_{53}$	$a_{59} + b_{53}$	$a_{46} + b_{43}$	$a_{5\,12} + b_{52}$	$a_{4\,12} + b_{42}$
$a_{57} + b_{54}$	$a_{5\,10} + b_{54}$	$a_{47} + b_{44}$	$a_{5\,13} + b_{54}$	$a_{4\,13} + b_{44}$

S_6	S_7	S_8	S_9	S_{10}
a_{21}	a_{22}	a_{13}	a_{14}	a_{24}
b_{21}	b_{22}	b_{13}	b_{14}	b_{24}
a_{54}	a_{44}	a_{33}	a_{22}	a_{34}
b_{54}	b_{44}	b_{33}	b_{22}	b_{34}
$a_{25} + b_{22}$	$a_{28} + b_{21}$	$a_{1\,11} + b_{11}$	$a_{1\,14} + b_{11}$	$a_{2\,14} + b_{21}$
$a_{26} + b_{23}$	$a_{29} + b_{23}$	$a_{1\,12} + b_{12}$	$a_{1\,15} + b_{12}$	$a_{2\,15} + b_{22}$
$a_{27} + b_{24}$	$a_{2\,10} + b_{24}$	$a_{1\,13} + b_{14}$	$a_{1\,16} + b_{13}$	$a_{2\,16} + b_{23}$
$a_{5\,14} + b_{51}$	$a_{4\,14} + b_{41}$	$a_{3\,11} + b_{31}$	$a_{2\,11} + b_{21}$	$a_{3\,14} + b_{31}$
$a_{5\,15} + b_{52}$	$a_{4\,15} + b_{42}$	$a_{3\,12} + b_{32}$	$a_{2\,12} + b_{22}$	$a_{3\,15} + b_{32}$
$a_{5\,16} + b_{53}$	$a_{4\,16} + b_{43}$	$a_{3\,13} + b_{34}$	$a_{2\,13} + b_{24}$	$a_{3\,16} + b_{33}$

With FS-PIR protocol 2, the retrieval process would look like: Assume each subfile is of 4 bits, which means the file is of size 20 bits.

S_1	S_2	S_3	S_4	S_5
a_{11}	a_{31}	a_{41}	a_{51}	$a_{12} + b_{11}$
b_{11}	b_{31}	b_{41}	b_{51}	$a_{43} + b_{41}$
$a_{52} + b_{51}$	$a_{53} + b_{51}$	$a_{32} + b_{31}$	$a_{42} + b_{41}$	

S_6	S_7	S_8	S_9	S_{10}
a_{21}	$a_{22}+b_{21}$	$a_{13}+b_{11}$	$a_{14}+b_{11}$	$a_{24}+b_{21}$
b_{21}	$a_{44}+b_{41}$	$a_{33}+b_{31}$	$a_{23}+b_{21}$	$a_{34}+b_{31}$
$a_{54}+b_{51}$				

Note that with either of the protocols retrieval rate $R = 4/5$. *Also, for this system* $C = (1 + 1/t)^{-1} = 4/5$.

3.6 Subpacketization

Subpacketization required by a PIR scheme is often defined as the number of small packets/symbols into which each file needs to be divided in order to apply the PIR protocol. For the problem of homogeneous PIR, first capacity achieving PIR protocol was proposed by Tandon *et al.* [15] which required a subpacketization of $\binom{N}{t}t^F$. The subpacketization was then improved by Zhang *et al.* [14] where the authors achieved capacity by using a t times lesser subpacketization. Woolsey *et al.* further reduced the subpacketization to Nt^{F-1} [16]. In [16], the authors propose two protocols- one for the case when t divides N, and other when t doesnot divide N. Also, unlike our protocol, their scheme is a rigid one and doesn't exploit combinatorial designs.

For our scheme, we first require a sub-division of each file into v subfiles and then the subfiles have to follow a subpacketization as per the FS-PIR protocol used. Thus, for the retrieval of individual subfiles if we use **FS-PIR protocol 1** we would require a subpacketization of $\mathbf{vt^F}$, using **FS-PIR protocol 2** would require a subpacketization of $\mathbf{vt^{F-1}}$ and using **FS-PIR protocol 3** would require a subpacketization of $\mathbf{v(t-1)}$.

A lower subpacketization for any PIR scheme is desirable as it makes the scheme applicable to a wider variety of files. Also, a lower subpacketization usually implies lower upload cost. In order to minimize subpacketization for our scheme, we need to minimize the number of subfiles v.

Theorem 7 (Capacity achieving PIR protocols) *For any given* μ-(F, N) *system with* $\mu = r/N$, $r \in [N]$ *and file size* $L = v \times r^F$, *we can design a capacity achieving PIR scheme by using a* $\left(\frac{N}{\gcd(N,r)}, \frac{r}{\gcd(N,r)}, N, r\right)$*-configuration in Theorem 4.*

Proof: Since the parameters $v = \frac{N}{\gcd(N,r)}$, $k = \frac{r}{\gcd(N,r)}$, $b = N$ and r satisfy eq (2.13), a configuration with the above parameters exists. One way of constructing such configuration is presented in Lemma 3. The PIR scheme designed by using this configuration for the storage scheme in Theorem 4, and FS-PIR protocol 1 for the retrieval of individual subfiles achieves capacity for the μ-(F, N) system with $L = v \times r^F$. This is true since the FS-PIR protocol 1 achieves capacity for the 1-(F, r) systems with files size r^F. □

Corollary 8 (Capacity achieving PIR protocols) *For any given μ-(F, N) system with $\mu = r/N$, $r \in [N]$ and file size $L = v \times r^{F-1}$, we can design a capacity achieving PIR scheme by using a $\left(\frac{N}{\gcd(N,r)}, \frac{r}{\gcd(N,r)}, N, r\right)$-configuration in Corollary 5.*

Corollary 9 (Capacity achieving PIR protocols) *For any given μ-(F, N) system with $\mu = r/N$, $r \in [N]$ and file size $L = v \times (r-1)$, we can design a capacity achieving PIR scheme by using a $\left(\frac{N}{\gcd(N,r)}, \frac{r}{\gcd(N,r)}, N, r\right)$-configuration in Corollary 6.*

Thus, from the above results we can infer that using the **FS-PIR protocol 1** we can achieve a minimum subpacketization of $\frac{N}{\gcd(N,r)} r^F$, by using **FS-PIR protocol 2** we can achieve a minimum subpacketization of $\frac{N}{\gcd(N,r)} r^{F-1}$ and by **FS-PIR protocol 3** we can achieve a minimum subpacketization of $\frac{N}{\gcd(N,r)}(r-1)$. Also, interestingly in a recent independent work in [17], it was shown that $\frac{N}{\gcd(N,r)}(r-1)$ is the minimum subpacketization possible for any capacity achieving homogeneous storage constrained PIR scheme with equally sized subfiles. Thus it is established that our storage scheme is optimal in terms of subpacketization.

3.7 Memory Sharing

For the μ-(F, N) system generated from a (v, k, b, r)-configuration, memory sharing can be used to achieve PIR capacity at any arbitrary storage point $\mu \in [\frac{1}{b}, 1]$. The concept of memory sharing to achieve capacity at non integral values of t has commonly been employed by SC-PIR protocols [15, 21] and we provide it here for the sake of completeness. Memory sharing can be done as follows:

(i) Find r such that $\frac{r}{b} \leq \mu \leq \frac{r+1}{b}$. Set $\mu_1 = \frac{r}{b}$ and $\mu_2 = \frac{r+1}{b}$.

(ii) Find α such that $\mu = \alpha\mu_1 + (1-\alpha)\mu_2$.

(iii) Divide each file $W_i,\ i \in [F]$, into two partitions $W_i^{(1)}$ and $W_i^{(2)}$, of sizes αL and $(1-\alpha)L$ respectively.

(iv) Now, store partition $W_i^{(1)}$ using a (v_1, k_1, b, r)-configuration and partition $W_i^{(2)}$ using a $(v_2, k_2, b, r+1)$-configuration.

(v) The above gives two partitions of storage of each server $Z_j,\ j \in [b]$ which are denoted as $Z_j^{(1)}$ and $Z_j^{(2)}$, where $Z_j^{(k)}$ has subfiles from $W_i^{(k)},\ i \in [F]$ only. The sizes of $Z_j^{(1)}$ and $Z_j^{(2)}$ are $\alpha\mu_1 FL$ and $(1-\alpha)\mu_2 FL$ respectively.

(vi) Let D_1 be the total number of bits downloaded by use of the proposed protocol when $\mu = \mu_1$, and D_2 be the total number of bits downloaded by use of the proposed protocol when $\mu = \mu_2$. Since the protocol is optimal at μ_1 and μ_2, $D^*(\mu_1) = \frac{D_1}{L}$ and $D^*(\mu_2) = \frac{D_2}{L}$. Now, for file partitions $W_j^{(1)}$ and servers partitions $Z_j^{(1)}$, we can use the proposed protocol to retrieve file partition $W_\theta^{(1)}$, by downloading a total of αD_1 bits. Similarly $W_\theta^{(2)}$, can be retrieved by downloading $(1-\alpha)D_2$ bits. Thus, for file W_θ, the download cost per bit,

$$\begin{aligned} D(\mu) &= \frac{\text{Total number of bits downloaded}}{\text{Size of file retrieved}} \\ &= \frac{\alpha D_1 + (1-\alpha)D_2}{L} \\ &= \alpha D^*(\mu_1) + (1-\alpha)D^*(\mu_2) \end{aligned} \tag{3.11}$$

Thus, we can achieve the lower convex hull of $\left(\mu = \frac{r}{b},\ D^*(\mu) = \sum_{f=0}^{F-1} \frac{1}{r^f}\right)$ pairs, for $r = 1, 2, \ldots, b$ and hence our protocol achieves capacity for arbitrary μ (refer eq (2.8)).

CHAPTER 4

FAULT TOLERANCE

So far, we were given a system of N servers and F files and we stored these files on the servers so as to optimize the retrieval rate and have minimum subpacketization. But what would happen if after the storage phase, some server goes down. We cannot go and change the storage at other servers, we have to go on with what we have and try to get the best out of it.

There can be a variety of events that could lead to a server failure, such as loss of power, hardware malfunction, operating system crashes, cyberattack, natural disaster etc. The event of 1 server going down amongst a set of N servers is a very realistic scenario. Until the server is repaired and up again, the user has to access files using the remaining $N-1$ servers only. Failure of 2 servers is a comparatively rarer scenario, and failure of more servers is even rarer. If a system of 10 physically independent 99.9% reliable servers is considered, at any given time instance Pr(1 server down) = 0.991% , Pr(2 servers down) = 0.004% and Pr(3 servers down) = 0.00001% (refer [22]). In this chapter we analyze our proposed protocol under the scenarios of 1 and 2 servers going down. We observe that in the case of 1 server failure, our protocol achieves capacity for the new system and also is able to uniformly distribute the excess load due to 1 server failure on the remaining servers. We also compare the load distribution with the load distribution under homogeneous storage constrained PIR protocol of [16]. In the event of 2 server failure, we see that the rate our protocol achieves is slightly less than capacity, and is the same irrespective of which 2 servers go down. Also, as the number of servers increase, the gap to capacity goes down and we achieve capacity asymptotically as $N \to \infty$.

4.1 One server failure

When a server goes down, the rate of PIR protocol for the new system is bound to decrease to the fact that we now have lesser resources. The best we can do is achieve capacity for the new system. Another aspect to focus on is how the excess load due to a server failure gets distributed on the remaining servers. To quantify the load on the servers, we define a metric to quantify load on servers.

Definition 9 *(Server Load:) Server load (ζ_n) is equal to the number of requests the n^{th} server has to serve for retrieval of the desired file.*

Ideally, we would want ζ_n to be the same for all n, else the server which needs to serve more number of requests than others would act as a bottleneck in the retrieval process and we would not be optimizing resource usage. Also, if number of requests on a particular server are unevenly high, its request latency (which corresponds to the time taken to serve a request) may go up further delaying the whole retrieval process. Note that for the setting of FS-PIR without any server failure, PIR protocol of [4] achieves uniform load distribution whereas protocols of [18] and [23] do not uniformly distribute load on all the servers.

In case of storage constrained PIR with one server failure, our proposed protocol is capacity achieving for the one-reduced system and can uniformly distribute the load amongst all the remaining servers. We demonstrate the same through an example first.

Example 12 *In the $\frac{3}{7}$-$(2, 7)$ system of Example 10 with $t = \mu N = 3$ and $F = 2$ files a and b, assume that each file is now of length $L = 252$ bits and each subfile is $\ell = 36$ bits long. Each file is divided into $v = 7$ subfiles and these subfiles are stored on the servers using the Fano plane as shown in Example 10. Now suppose that due to some issue server S_1 fails. Due to this failure the 6 subfiles stored on S_1 are inaccessible. Assume that the user wants to retrieve file a.*

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_3	a_1	a_1	a_3	a_2	a_2
a_2	a_4	a_5	a_4	a_6	a_5	a_4
a_3	a_5	a_6	a_7	a_7	a_7	a_6
b_1	b_3	b_1	b_1	b_3	b_2	b_2
b_2	b_4	b_5	b_4	b_6	b_5	b_4
b_3	b_5	b_6	b_7	b_7	b_7	b_6

(4.1)

As server S_1 fails, the recovery of subfiles a_1, a_2, a_3 is affected as they are now available from 2 servers only, while subfiles a_4, a_5, a_6 and a_7 are available from 3 servers each. From the storage scheme, it can observed that all the remaining active servers $S_2 - S_7$ each contain 1 subfile whose recovery is affected and 2 subfiles whose recovery is unaffected. The subfiles whose recovery is affected will now have to be retrieved from 2 servers whereas other subfiles will be recovered from 3 servers.

In the example under consideration, subfiles a_1, a_2 and a_3 are retrieved from 2 servers each using the capacity achieving protocol of [4] . Subfile a_1 is retrieved from the 1-$(2, 2)$ system with servers S_3, S_4 and files a_1, b_1. For this system, $C = (1 + 1/2)^{-1} = \ell/D$ (from eq (2.1)) and $\ell = 36$. So total $D = 54$ requests have to be served in order to retrieve subfile a_1. Since the protocol of [4] is symmetric wrt servers, each of the participating servers would receive equal number of queries. Thus, both servers S_3 and S_4 would serve 27 requests for retrieval subfile a_1. Similarly for subfiles a_2 and a_3 each participating server would serve 27 requests for retrieval of these subfiles.

The retrieval of subfiles a_4, a_5, a_6 and a_7 is not affected and they are retrieved from 3 servers each. Subfile a_4 is retrieved from the 1-$(2, 3)$ system with servers S_2, S_4, S_7 and files a_4, b_4. For this system, $C = (1+1/3)^{-1} = \ell/D$ (from eq (2.1)) and $\ell = 36$. So total $D = 48$ requests have to be served in order to retrieve subfile a_4. Since the subfile is retrieved from 3 servers , each of the servers S_2, S_4 and S_7 would serve 16 requests each. Similarly for subfiles a_5, a_6 and a_7 each of the three participating server would serve 16 requests for retrieval of these subfiles.

Table 4.1 captures the number of requests each server serves for the retrieval of

different subfiles. Also, it can be seen from the table that each server receives a total of 59 requests and hence server load $\zeta_n = 59$ *for all* $n \in [2, 7]$. *Also* $\zeta_{\max} = 59$ *and* $\zeta_{total} = 354$.

Firstly, we note that to retrieve 252 *bits of the desired file we download* 354 *bits, hence* $R = \frac{252}{354} = \frac{42}{59}$. *For the new system,* $\mu' = \mu = \frac{3}{7}$, $N' = 6$ *and* $t' = \mu * N' = \frac{18}{7} = 3 \times (\frac{2}{7}) + 4 \times (\frac{3}{7})$. *So for this system optimal download cost* $D^*(\mu) = (\frac{3}{7}) \times (1 + \frac{1}{2}) + (\frac{4}{7}) * (1 + \frac{1}{3}) = \frac{59}{42}$ *(using eq (2.8)) and* $C = \frac{42}{59}$. *Thus, rate of the retrieval process matches capacity for the new system.*

Next, we compare load distribution under our protocol with load distribution of the protocol proposed by Woolsey et al. in [16]. As per their protocol, the storage after failure of S_1 *would look like:*

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_2	a_3	a_4	a_5	a_6	a_7	a_1
a_3	a_4	a_5	a_6	a_7	a_1	a_2
b_1	b_2	b_3	b_4	b_5	b_6	b_7
b_2	b_3	b_4	b_5	b_6	b_7	b_1
b_3	b_4	b_5	b_6	b_7	b_1	b_2

Again, subfiles a_1, a_2, a_3 *are available from 2 servers each and subfiles* a_4, a_5, a_6, a_7 *are available from 3 servers each. As subfiles* a_1, a_2 *and* a_3 *are retrieved from 2 servers, each participating server would serve* 27 *requests each. For subfiles* a_4, a_5, a_6 *and* a_7 *each of the three participating server would serve* 18 *requests each. Table 4.2 captures the number of requests each server serves for the retrieval of different subfiles. Also, it can be seen from the table that* $\zeta_{\max} = 70$ *and* $\zeta_{total} = 354$.

Subfile	Number of requests served					
	S_2	S_3	S_4	S_5	S_6	S_7
a_1		27	27			
a_2					27	27
a_3	27			27		
a_4	16		16			16
a_5	16	16			16	
a_6		16		16		16
a_7			16	16	16	
Server load (ζ_n)	59	59	59	59	59	59

Table 4.1: Server load for Proposed PIR protocol

Subfile	Number of requests served					
	S_2	S_3	S_4	S_5	S_6	S_7
a_1					27	27
a_2	27					27
a_3	27	27				
a_4	16	16	16			
a_5		16	16	16		
a_6			16	16	16	
a_7				16	16	16
Server load (ζ_n)	70	59	48	48	59	70

Table 4.2: Server load for protocol of Woolsey *et al.* [16]

Clearly, for both our protocol and protocol of Woolsey et al. the total load is same, but our protocol is able to distribute the load uniformly on all the servers. Figure 4.1 shows the distribution of load due on the servers in the one-reduced system for the two protocols.

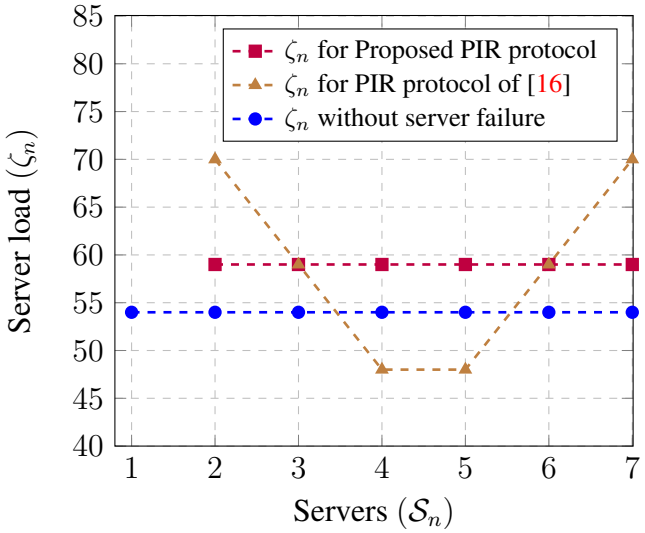


Figure 4.1: Server load on i-th server for a one-reduced $\frac{3}{7}$-$(2, 7)$ system

Example 13 *Consider a $\frac{4}{7}$-$(2, 7)$ system with files a and b each $L = 1008$ bit long. As per our storage scheme we divide each file into $v = 7$ subfiles each of $\ell = 144$ bits. The i-th subfiles of a and b are denoted by a_i and b_i, $i \in [7]$ respectively. We use the $(7, 4, 2, 7, 4)$-BIBD for the storage scheme and store the subfiles as shown below.*

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{4.2}$$

Assume that we want to retrieve file a, but server S_1 fails due to some issue. Due to this failure the 8 subfiles stored on S_1 are inaccessible. Assume that the user wants to retrieve file a.

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_4	a_1	a_1	a_1	a_2	a_2
a_3	a_5	a_3	a_2	a_2	a_3	a_3
a_4	a_6	a_6	a_5	a_4	a_5	a_4
a_5	a_7	a_7	a_6	a_7	a_7	a_6
b_1	b_4	b_1	b_1	b_1	b_2	b_2
b_3	b_5	b_3	b_2	b_2	b_3	b_3
b_4	b_6	b_6	b_5	b_4	b_5	b_4
b_5	b_7	b_7	b_6	b_7	b_7	b_6

(4.3)

As server S_1 fails, the recovery of subfiles a_1, a_3, a_4, a_5 is affected as they are now available from 3 servers only, while subfiles a_2, a_6 and a_7 are available from 4 servers each. From the storage scheme, it can observed that all the remaining active servers $S_2 - S_7$ each contain 2 subfiles whose recovery is affected and 2 subfiles whose recovery is unaffected. The subfiles whose recovery is affected will now have to be retrieved from 3 servers whereas other subfiles will be recovered from 4 servers.

Subfiles a_1, a_3, a_4 and a_5 are retrieved from 3 servers each using the capacity achieving protocol of [4]. Subfile a_1 is retrieved from the 1-$(2, 3)$ system with servers S_3, S_4, S_5 and files a_1, b_1. For this system, $C = (1 + 1/3)^{-1} = \ell/D$ (from eq (2.1)) and $\ell = 144$. So total $D = 192$ requests have to be served in order to retrieve subfile a_1. Since the protocol of [4] is symmetric wrt servers, each of the participating servers would receive equal number of queries. Thus, all 3 servers S_3, S_4 and S_5 would serve 64 requests for retrieval subfile a_1. Similarly for subfiles a_3, a_4 and a_5 each participating server would serve 64 requests for retrieval of these subfiles.

The retrieval of subfiles a_2, a_6 and a_7 is not affected and they are retrieved from 4 servers each. Subfile a_2 is retrieved from the 1-$(2, 4)$ system with servers S_4, S_5, S_6, S_7 and files a_2, b_2. For this system, $C = (1 + 1/4)^{-1} = \ell/D$ (from eq (2.1)) and $\ell = 144$. So total $D = 180$ requests have to be served in order to retrieve subfile a_2. Since the subfile is retrieved from 4 servers , each of the servers S_4, S_5, S_6, S_7 would serve 45 requests each. Similarly for subfiles a_6 and a_7 each of the four participating server would serve 16 requests for retrieval of these subfiles.

Table 4.3 captures the number of requests each server serves for the retrieval of different subfiles. Also, it can be seen from the table that each server receives a total of 59 requests and hence server load $\zeta_n = 218$ *for all* $n \in [2, 7]$. *Also* $\zeta_{\max} = 218$ *and* $\zeta_{total} = 1308$.

Note that to retrieve 1008 *bits of the desired file we download* 1308 *bits, hence* $R = \frac{1008}{1308} = \frac{84}{109}$. *For the new system,* $\mu' = \mu = \frac{4}{7}$, $N' = 6$ *and* $t' = \mu * N' = \frac{24}{7} = 4 \times (\frac{3}{7}) + 3 \times (\frac{4}{7})$. *So for this system optimal download cost* $D^*(\mu) = (\frac{4}{7}) \times (1 + \frac{1}{3}) + (\frac{3}{7}) * (1 + \frac{1}{4}) = \frac{109}{84}$ *(using eq (2.8)) and* $C = \frac{84}{109}$. *Thus, rate of the retrieval process matches capacity for the new system.*

Next, we compare load distribution under our protocol with load distribution of the protocol proposed by Woolsey et al. in [16]. As per their protocol, the storage after failure of S_1 *would look like:*

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_2	a_3	a_4	a_5	a_6	a_7	a_1
a_3	a_4	a_5	a_6	a_7	a_1	a_2
a_4	a_5	a_6	a_7	a_1	a_2	a_3
b_1	b_2	b_3	b_4	b_5	b_6	b_7
b_2	b_3	b_4	b_5	b_6	b_7	b_1
b_3	b_4	b_5	b_6	b_7	b_1	b_2
b_4	b_5	b_6	b_7	b_1	b_2	b_3

Again, subfiles a_1, a_2, a_3 *and* a_4 *are available from 3 servers each whereas subfiles* a_5, a_6, a_7 *are available from 4 servers each. As subfiles* a_1, a_2, a_3 *and* a_4 *are retrieved from 3 servers, each participating server would serve* 64 *requests each. For subfiles* a_5, a_6 *and* a_7 *each of the four participating servers would serve* 45 *requests each. Table 4.4 captures the number of requests each server serves for the retrieval of different subfiles. Also, it can be seen from the table that* $\zeta_{\max} = 237$ *and* $\zeta_{total} = 1308$.

Subfile	Number of requests served					
	S_2	S_3	S_4	S_5	S_6	S_7
a_1		64	64	64		
a_2			45	45	45	45
a_3		64			64	64
a_4	64			64		64
a_5	64		64		64	
a_6	45	45	45			45
a_7	45	45		45	45	
Server load (ζ_n)	218	218	218	218	218	218

Table 4.3: Server load for Proposed PIR protocol

Subfile	Number of requests served					
	S_2	S_3	S_4	S_5	S_6	S_7
a_1				64	64	64
a_2	64				64	64
a_3	64	64				64
a_4	64	64	64			
a_5	45	45	45	45		
a_6		45	45	45	45	
a_7			45	45	45	45
Server load (ζ_n)	237	218	199	199	218	237

Table 4.4: Server load for protocol of Woolsey *et al.* [16]

Again, for both our protocol and protocol of Woolsey et al. the total load is same, but our protocol is able to distribute the load uniformly on all the servers.

4.1.1 Results on Fault Tolerance

Now, we formally state the 2 results discussed through the examples.

Theorem 10 *Consider a μ-(F, N) system generated from a $(v, k, b = N, r = t)$ configuration. Assume $r \in \mathbb{Z}^+ \setminus \{1\}$. In this system if a server fails, the proposed protocol*

is capacity achieving for the one-reduced system, assuming that the subfile size is large enough to support capacity achieving PIR protocol for both 1-(F, r) and 1-$(F, r-1)$ systems.

Proof: If any server fails, then the k subfiles of each file stored on it are unavailable. For the desired file, these k subfiles now have to be retrieved from remaining $r-1$ servers on which they are stored. The other $v-k$ subfiles are unaffected and hence would be retrieved from r servers. The k subfiles available on $r-1$ servers, are retrieved by using the capacity achieving PIR protocol of [4] for 1-$(F, r-1)$ system. Thus for retrieving these subfiles $R = \frac{\ell}{D_{r-1}} = (1+\frac{1}{r-1}+\cdots+\frac{1}{(r-1)^{F-1}})^{-1}$, where $\ell = \frac{L}{v}$ is the subfile size. So, for retrieving each of these k subfiles $D_{r-1} = \frac{L}{v}(1+\frac{1}{r-1}+\cdots+\frac{1}{(r-1)^{F-1}})$ bits are downloaded.

Similarly, the other $v-k$ subfiles are available on r servers each, and are retrieved by using the capacity achieving PIR protocol of [4] for 1-(F, r) system. Thus for retrieving these subfiles $R = \frac{\ell}{D_r} = (1+\frac{1}{r}+\cdots+\frac{1}{r^{F-1}})^{-1}$. So, for retrieving each of these $v-k$ subfiles $D_r = \frac{L}{v}(1+\frac{1}{r}+\cdots+\frac{1}{r^{F-1}})$ bits are downloaded.

Thus, total download cost

$$D_{\text{total}} = \frac{kL}{v}\left(1+\frac{1}{r-1}+\cdots+\frac{1}{(r-1)^{F-1}}\right)+\frac{(v-k)L}{v}\left(1+\frac{1}{r}+\cdots+\frac{1}{r^{F-1}}\right) \tag{4.4}$$

and download cost per bit

$$D(\mu) = \frac{k}{v}\left(1+\frac{1}{r-1}+\cdots+\frac{1}{(r-1)^{F-1}}\right)+\left(1-\frac{k}{v}\right)\left(1+\frac{1}{r}+\cdots+\frac{1}{r^{F-1}}\right) \tag{4.5}$$

Next, we calculate the optimal download cost per bit for the new system after a server has failed. Note that the one-reduced system has $N' = b-1$ servers with normalized storage $\mu' = \mu = \frac{r}{b}$ and total normalized storage for the system $t' = \mu(b-1) = r\frac{b-1}{b}$.

Note that $t' = r\frac{b-1}{b} = \frac{r}{b}(r-1)+\left(1-\frac{r}{b}\right)r$ where $r, r-1 \in \mathbb{Z}^+$, Using eq (2.8), the optimal download cost per bit for the given one-reduced system,

$$D^*(\mu) = \frac{r}{b}\left(1 + \frac{1}{r-1} + \cdots + \frac{1}{(r-1)^{F-1}}\right) + \left(1 - \frac{r}{b}\right)\left(1 + \frac{1}{r} + \cdots + \frac{1}{r^{F-1}}\right) \tag{4.6}$$

But for any (v, k, b, r) configuration from eq. (2.13) $bk = vr \implies \frac{r}{b} = \frac{k}{v}$. Thus $D(\mu) = D^*(\mu)$. □

Remark 2 *Theorem 10 holds true under all three full storage PIR protocols of [4],[18] and [23]. If protocol of [4] is used for subfile retrieval, minimum file size required $L = v \times lcm(t^F, t^{F-1})$. With protocols of [18] and [23] minimum file sizes required are $L = v \times lcm(t^{F-1}, t^{F-2})$ and $L = v \times lcm(t-1, t-2)$ respectively.*

Theorem 11 *Consider a μ-(F, N) system generated from a $(v, k, \lambda, b = v, r = k)$ symmetric BIBD and suppose that server S_m fails after the storage phase. Assume $r \in \mathbb{Z}^+ \setminus \{1\}$ and PIR protocol of [4] is used for retrieving individual subfiles. Then for the one-reduced system, server load is distributed uniformly on all the available servers during retrieval of the desired file i.e $\zeta_i = \zeta_j$ for all $i, j \in [N] \setminus m$.*

Proof: In a symmetric $(v, k, \lambda, b = v, r = k)$ BIBD , any two blocks have exactly λ elements in common. Therefore, if any server fails, recovery of exactly λ subfiles on each server is affected. Thus each of the available server has λ subfiles which are effected and are available on only $r - 1$ servers each and $k - \lambda$ subfiles which are unaffected and available on r servers each. Server load for each server is a sum of loads for the k files present on the server. Since the protocol of [4] is symmetric wrt servers, for each subfile, all the participating servers will share the load equally. Assume that the total download cost for a subfile to be retrieved from r servers is D_r and for a subfile to be retrieved from $r - 1$ servers is D_{r-1}. Then each server would serve $\frac{D_{r-1}}{r-1}$ requests each for the λ subfiles available from $r - 1$ servers and $\frac{D_r}{r}$ requests each for the $k - \lambda$ subfiles available from r servers. Thus $\zeta_n = \frac{\lambda}{r-1}D_{r-1} + \frac{k-\lambda}{r}D_r$ for all $n \in [N] \setminus m$. □

Remark 3 *Theorem 11 does not hold true if protocols of [18] or [23] are used for subfile retrieval. This is because these protocols are not server symmetric and may divide the load of retrieving a subfile unevenly on the involved servers, hence skewing the total load distribution as well.*

4.2 Two server failure

Next, we analyse the scenario of 2 out of N servers going down. We assume that the system under consideration has $t \in \mathbb{Z}^+ \setminus \{1, 2\}$. We observe that if BIBD's are used for the storage scheme of the proposed protocol, the rate of PIR protocol for the new system does not depend on which 2 servers go down. This is unlike the protocol of Woolsey *et al.* wherein the rate of PIR protocol for the new system is dependent on which 2 servers go down. We demonstrate the same through an example below:

Example 14 *Consider the* $\frac{3}{7} - (2, 7)$ *system of Example 12 with files* a *and* b *of sizes* $L = 252$ *bits. The files are divided and stored on the servers as per the Fano plane as shown in Example 12. Again, say the user wants to retrieve file* a*. Assume that servers* S_1 *and* S_2 *are both down and unable to serve user requests.*

We wish to characterize the rate of proposed PIR protocol for the new system. After failure of servers S_1 *and* S_2*, the storage as per our proposed protocol looks like:*

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_3	a_1	a_1	a_3	a_2	a_2
a_2	a_4	a_5	a_4	a_6	a_5	a_4
a_3	a_5	a_6	a_7	a_7	a_7	a_6
b_1	b_3	b_1	b_1	b_3	b_2	b_2
b_2	b_4	b_5	b_4	b_6	b_5	b_4
b_3	b_5	b_6	b_7	b_7	b_7	b_6

(4.7)

Due the the server failures, subfiles a_3 *is now available from only 1 server whereas subfiles* a_1, a_2, a_4 *and* a_5 *are available only from 2 servers. The subfiles* a_6 *and* a_7 *are unaffected and are still available from 3 servers each. As per the analysis done in Example 12, a subfile of size* $l = 36$ *bits, if retrieved from 2 servers, needs each server to serve* 27 *requests. If the same subfile is retrieved from 3 servers, it need each participating server to serve* 16 *requests each. Similarly, if the subfile has to be retrieved from only 1 server, it would need the server to serve* 72 *requests(the whole projected database which has 36 bits of both subfiles has to be downloaded). Table 4.9*

shows the number of requests each server has to serve for retrieval of different subfiles under proposed protocol.

The proposed protocol has a total download cost of $D = 384$ *bits, which corresponds to rate* $R = \frac{L}{D} = \frac{252}{384} = 0.65625$.

Next, we compare this with the retrieval rate of protocol proposed by Woolsey et al. under the same scenario. As per the protocol of Woolsey et al., after failure of servers S_1 *and* S_2*, the storage at the servers will look like:*

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_2	a_3	a_4	a_5	a_6	a_7	a_1
a_3	a_4	a_5	a_6	a_7	a_1	a_2
b_1	b_2	b_3	b_4	b_5	b_6	b_7
b_2	b_3	b_4	b_5	b_6	b_7	b_1
b_3	b_4	b_5	b_6	b_7	b_1	b_2

It can be noted that in the new system, subfile a_2 *and* a_3 *are available only on 1 server each, subfiles* a_1 *and* a_4 *are available on 2 servers each, whereas the other subfiles are unaffected and available from 3 servers each. Table 4.10 shows the number of requests each server has to serve for retrieval of different subfiles under protocol of Woolsey et al.*

Subfile	No: of requests served				
	S_3	S_4	S_5	S_6	S_7
a_1	27	27			
a_2				27	27
a_3			72		
a_4		27			27
a_5	27			27	
a_6	16		16		16
a_7		16	16	16	
Total Download cost = 384					

Table 4.5: Download cost for Proposed PIR protocol

Subfile	No: of requests served				
	S_3	S_4	S_5	S_6	S_7
a_1				27	27
a_2					72
a_3	72				
a_4	27	27			
a_5	16	16	16		
a_6		16	16	16	
a_7			16	16	16
Total Download cost = 396					

Table 4.6: Download cost for Woolsey *et al.*

Next, we assume that instead of S_1 and S_2, servers S_1 and S_3 go down and compare the performance of the two protocols. The tables below show the storage and download costs for proposed protocol and protocol of Woolsey et al. [16].

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_3	a_1	a_1	a_3	a_2	a_2
a_2	a_4	a_5	a_4	a_6	a_5	a_4
a_3	a_5	a_6	a_7	a_7	a_7	a_6
b_1	b_3	b_1	b_1	b_3	b_2	b_2
b_2	b_4	b_5	b_4	b_6	b_5	b_4
b_3	b_5	b_6	b_7	b_7	b_7	b_6

Table 4.7: two-reduced system storage for proposed protocol

S_1	S_2	S_3	S_4	S_5	S_6	S_7
a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_2	a_3	a_4	a_5	a_6	a_7	a_1
a_3	a_4	a_5	a_6	a_7	a_1	a_2
b_1	b_2	b_3	b_4	b_5	b_6	b_7
b_2	b_3	b_4	b_5	b_6	b_7	b_1
b_3	b_4	b_5	b_6	b_7	b_1	b_2

Table 4.8: two-reduced system storage for Woolsey *et al.*

Subfile	No: of requests served				
	S_2	S_4	S_5	S_6	S_7
a_1	72				
a_2				27	27
a_3	27		27		
a_4	16	16			16
a_5	27			27	
a_6			27		27
a_7		16	16	16	
Total Download cost = 384					

Table 4.9: Download cost for Proposed PIR protocol

Subfile	No: of requests served				
	S_2	S_4	S_5	S_6	S_7
a_1				27	27
a_2	27				27
a_3	72				
a_4	27	27			
a_5		27	27		
a_6		16	16	16	
a_7			16	16	16
Total Download cost = 384					

Table 4.10: Download cost for Woolsey *et al.*

It should be noted that the total download cost for proposed protocol remains the same in both the cases, but for the protocol of Woolsey et al. the total download cost depends on which two servers fail.

Theorem 12 *Suppose that we use a $(v, k, \lambda, b = N, r = t)$ BIBD for the storage scheme of proposed protocol. Now, if any two servers fail, the retrieval rate for the two-reduced system does not depend on which 2 particular servers fail. The gap to optimal download cost per bit for our scheme is given by*

$$D(\mu) - D^*(\mu) = \frac{\lambda}{v}\left\{ \sum_{i=1}^{f-1} \left(\frac{1}{r^i} + \frac{1}{(r-2)^i} - \frac{2}{(r-1)^i}\right) \right\} \tag{4.8}$$

Proof: Given a (v, k, λ, b, r)-BIBD, if any 2 servers fail

(i) λ subfiles are left on $(r-2)$ servers.

(ii) $2(k-\lambda)$ subfiles are left on $(r-1)$ servers.

(iii) Remaining $(v-(2k-\lambda))$ subfiles are on r servers.

Let us represent the total download cost of a subfile from $r, r-1$ and $r-2$ servers as D_r, D_{r-1} and D_{r-2} respectively.

So, the total download cost for overall retrieval

$$D_{\text{total}} = \frac{L}{v}(v-(2k-\lambda))D_r + \frac{L}{v}(2(k-\lambda))D_{r-1} + \frac{L}{v}(\lambda)D_{r-2} \tag{4.9}$$

and download cost per bit

$$D(\mu) = \frac{v-(2k-\lambda)}{v}D_r + \frac{2(k-\lambda)}{v}D_{r-1} + \frac{\lambda}{v}D_{r-2} \tag{4.10}$$

Next, we calculate the optimal download cost per bit for the new system. Note that for the two-reduced system $t' = \mu(N-2) = \mu(r-2) = r\frac{b-2}{b} = \frac{2r}{b}(r-1) + (1-\frac{2r}{b})r$.

Thus using eq (2.8) the optimal download cost per bit for the two-reduced system

$$D^*(\mu) = \frac{2r}{b}D_{r-1} + (1-\frac{2r}{b})D_r \tag{4.11}$$

Finally,

$$\begin{aligned} D(\mu) - D^*(\mu) &= D_r\left(1 - \frac{2k-\lambda}{v} - 1 + \frac{2r}{b}\right) \\ &\quad + D_{r-1}\left(\frac{2(k-\lambda)}{v} - \frac{2r}{b}\right) + D_{r-2}\left(\frac{\lambda}{v}\right) \\ &= D_r\left(\frac{\lambda}{v}\right) - D_{r-1}\left(\frac{2\lambda}{v}\right) + D_{r-2}\left(\frac{\lambda}{v}\right) \end{aligned}$$

Plugging in the values of D_r, D_{r-1}, D_{r-2}, we get

$$D(\mu) - D^*(\mu) = \frac{\lambda}{v}\Big\{ \big(\frac{1}{r} + \frac{1}{r-2} - \frac{2}{r-1}\big) + \big(\frac{1}{r^2} + \frac{1}{(r-2)^2} - \frac{2}{(r-1)^2}\big) + \dots + \big(\frac{1}{r^{f-1}} + \frac{1}{(r-2)^{f-1}} - \frac{2}{(r-1)^{f-1}}\big)\Big\}$$

□

Since the performance of our protocol is the same irrespective of which 2 servers go down , it can be expected that our protocol is not severely affected in the worst case. Also, it can be noted that the gap to capacity goes down a the number of servers increases. Hence, in the asymptotic case $N \to \infty$, $D(\mu) = D^*(\mu)$ and capacity is achieved for the two-reduced system.

CHAPTER 5

SUMMARY AND FUTURE DIRECTIONS

Coming towards the end of this thesis, in this Chapter we summarise our work and discuss possible directions to expand our work.

In this work we proposed a novel capacity achieving PIR scheme with low subpacketization for uncoded homogeneous storage constrained PIR systems. The proposed PIR scheme is also capable of achieving optimal subpacketization and is flexible and can accommodate a number of system parameters. The proposed PIR protocol has two main components: a storage scheme and a retrieval scheme. The storage scheme is based on combinatorial designs, more specifically tactical configurations. The retrieval scheme works by projecting the given system to multiple instances of smaller systems with replicated servers having full storage capacity. The benefits of our proposed PIR protocol are summarised below. The proposed PIR protocol:

- Exists for any given N, F and μ.
- Achieves capacity
- Achieves optimal subpacketization for any homogeneous storage constraint PIR protocol with equally sized subfiles
- Achieves capacity for the one residual system, without any modification in the contents of available servers
- Uniformly distributes excess load due to 1 server failure on the set of available servers if a symmetric BIBD is used for storage scheme.
- For the two residual system, maintains a fixed retrieval rate independent of which 2 servers fail if storage scheme is based on a BIBD.

As a extension of this work, it would be interesting to reduce subpacketization requirements when $t \notin \mathbb{Z}$. Currently existing SC-PIR protocols use memory sharing to extend PIR protocols for the case of non-integral t, which requires a high subpacketization. Exploiting the fault tolerance of SC-PIR protocol may serve as one possible way to reduce subpacketization for systems with non-integral t. Another possible direction for future work is to apply the theory of combinatorial designs to construct a PIR

protocol for Storage constrained Heterogeneous PIR systems where each server has a different normalized storage. Currently existing PIR protocols for Heterogeneous PIR either have high subpacketization requirements or have a iterative placement strategy for the storage scheme. The challenge is to come up with a way of using combinatorial designs to construct a one shot storage scheme for Heterogeneous PIR systems which has low subpacketization requirements. Another interesting direction is the setting of Weakly PIR which brings in a practical trade off between the privacy and performance of PIR protocols. A practical example of weakly PIR would be say we are retrieving a video file from a server with 100's of audio files, 100's of images and 100's of videos. We might not be much concerned if the server knows that we are retrieving some video file but doesn't know which one. By, doing this sacrifice on perfect privacy, we gain in terms of the rate of the protocol, which means that now we will have to download a lesser amount of data to recover our desired file. The amount of privacy loss is usually quantified by metrics defined on the basis of mutual information between the desired file index and queries received by a server. Again, as a extension of our work, designing efficient PIR schemes based on combinatorial designs for different allowed information leakage values is a interesting direction.

REFERENCES

[1] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pp. 41–50, IEEE, 1995.

[2] R. Henry, F. Olumofin, and I. Goldberg, "Practical pir for electronic commerce," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 677–690, 2011.

[3] N. B. Shah, K. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *2014 IEEE International Symposium on Information Theory*, pp. 856–860, IEEE, 2014.

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

[5] H. Sun and S. A. Jafar, "The capacity of robust private information retrieval with colluding databases," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2361–2370, 2017.

[6] K. Banawan and S. Ulukus, "The capacity of private information retrieval from coded databases," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1945–1956, 2018.

[7] S. Kumar, H.-Y. Lin, E. Rosnes, and A. G. i Amat, "Achieving maximum distance separable private information retrieval capacity with linear codes," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4243–4273, 2019.

[8] M. A. Attia, D. Kumar, and R. Tandon, "The capacity of private information retrieval from uncoded storage constrained databases," *IEEE Transactions on Information Theory*, vol. 66, no. 11, pp. 6617–6634, 2020.

[9] S. El Rouayheb, A. Sprintson, and C. Georghiades, "On the index coding problem

and its relation to network coding and matroid theory," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3187–3195, 2010.

[10] S. A. Jafar, "Topological interference management through index coding," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 529–568, 2013.

[11] H. Sun and S. A. Jafar, "Index coding capacity: How far can one go with only shannon inequalities?," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3041–3055, 2015.

[12] A. Beimel, Y. Ishai, E. Kushilevitz, and I. Orlov, "Share conversion and private information retrieval," in *2012 IEEE 27th Conference on Computational Complexity*, pp. 258–268, IEEE, 2012.

[13] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[14] W. Zhang, Z. Zhou, U. Parampalli, and V. Sidorenko, "Capacity-achieving private information retrieval scheme with a smaller sub-packetization," *Advances in Mathematics of Communications*, p. 0, 2019.

[15] M. A. Attia, D. Kumar, and R. Tandon, "The capacity of uncoded storage constrained pir," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1959–1963, IEEE, 2018.

[16] N. Woolsey, R.-R. Chen, and M. Ji, "A new design of private information retrieval for storage constrained databases," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1052–1056, IEEE, 2019.

[17] J. Zhu, Q. Yan, X. Tang, and Y. Miao, "Capacity-achieving private information retrieval schemes from uncoded storage constrained servers with low sub-packetization," *arXiv preprint arXiv:2102.08058*, 2021.

[18] H. Sun and S. A. Jafar, "Optimal download cost of private information retrieval for arbitrary message length," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 2920–2932, 2017.

[19] C. Tian, H. Sun, and J. Chen, "Capacity-achieving private information retrieval codes with optimal message size and upload cost," *IEEE Transactions on Information Theory*, vol. 65, no. 11, pp. 7613–7627, 2019.

[20] D. Stinson, *Combinatorial designs: constructions and analysis*. Springer Science & Business Media, 2007.

[21] R. Tandon, "The capacity of cache aided private information retrieval," 2017.

[22] "ITIC 2015 - 2016 Global Server Hardware, Server OS Reliability Report." http://hosteddocs.ittoolbox.com/iticserverhardwareosreliabilityreport.pdf.

[23] H. Sun and S. A. Jafar, "Optimal download cost of private information retrieval for arbitrary message length," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 2920–2932, 2017.

[24] S. Kumar, E. Rosnes, and A. G. i Amat, "Private information retrieval in distributed storage systems using an arbitrary linear code," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1421–1425, IEEE, 2017.

[25] B. Cherowitzo, "t-designs." http://math.ucdenver.edu/wcherowi/courses/m6406/tdesigns.pd.

[26] R. Tajeddine, O. W. Gnilke, and S. El Rouayheb, "Private information retrieval from mds coded data in distributed storage systems," *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 7081–7093, 2018.

[27] M. Abdul-Wahid, F. Almoualem, D. Kumar, and R. Tandon, "Private information retrieval from storage constrained databases–coded caching meets pir," *arXiv preprint arXiv:1711.05244*, 2017.

[28] S. Agrawal, K. S. Sree, and P. Krishnan, "Coded caching based on combinatorial designs," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1227–1231, IEEE, 2019.

[29] N. Woolsey, R.-R. Chen, and M. Ji, "An optimal iterative placement algorithm for pir from heterogeneous storage-constrained databases," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[30] H. Sun, *Fundamental Limits of Private Information Retrieval*. PhD thesis, UC Irvine, 2017.

[31] A. Jäschke, B. Grohmann, F. Armknecht, and A. Schaad, "Industrial feasibility of private information retrieval.," *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 813, 2017.

[32] S. Kumar, A. G. i Amat, E. Rosnes, and L. Senigagliesi, "Private information retrieval from a cellular network with caching at the edge," *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 4900–4912, 2019.

[33] H.-Y. Lin, S. Kumar, E. Rosnes, A. G. i Amat, and E. Yaakobi, "Weakly-private information retrieval," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1257–1261, IEEE, 2019.

[34] K. Banawan, B. Arasli, Y.-P. Wei, and S. Ulukus, "The capacity of private information retrieval from heterogeneous uncoded caching databases," *IEEE Transactions on Information Theory*, vol. 66, no. 6, pp. 3407–3416, 2020.

[35] K. Banawan, B. Arasli, Y.-P. Wei, and S. Ulukus, "The capacity of private information retrieval from heterogeneous uncoded caching databases," 2019.

[36] Z. Jia, *On the capacity of weakly-private information retrieval*. PhD thesis, UC Irvine, 2019.

LIST OF PAPERS BASED ON THESIS

1. M. Shrivastava and P. Sarvepalli, "Capacity Achieving Uncoded PIR Protocol based on Combinatorial Designs." *arXiv preprint arXiv:2103.09804 (2021).*