A Novel Scheme for Cache-aided Multiuser Private Information Retrieval with User-to-user Privacy

Xiang Zhang*, Kai Wan†, Hua Sun‡, Mingyue Ji*, and Giuseppe Caire§
Department of Electrical and Computer Engineering, University of Utah*
School of Electronic Information and Communications, Huazhong University of Science and Technology†
Department of Electrical Engineering, University of North Texas‡
Department of Electrical Engineering and Computer Science, Technische Universität Berlin§
Email: *{xiang.zhang, mingyue.ji}@utah.edu, †kai_wan@hust.edu.cn, ‡hua.sun@unt.edu, §caire@tu-berlin.de

2023 57th Asilomar Conference on Signals, Systems, and Computers | 979-8-3503-2574-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/IEEECONF59524.2023.10476849

***Abstract*—Cache-aided Multiuser Private Information Retrieval (MuPIR) is an approach to achieve efficient file retrieval while ensuring multiuser demand privacy against curious servers in multiuser cache-aided PIR systems. More specifically, the demands of the users should be protected from any individual server (i.e., server privacy) during the two-phase retrieval procedure which consists of a cache placement phase and a private delivery phase. One limitation of the MuPIR model is that the users' demands are exposed to each other which is highly undesirable in modern-day distributed retrieval systems where user-to-user privacy has also become an important consideration. Motivated by this, we propose *cache-aided MuPIR with user privacy* (MuPIR-U), a new problem formulation that simultaneously enforces server privacy and user-to-user privacy. Besides server privacy, it is also required that each user should be prevented from learning other users' demands. We present an optimal scheme for the case of two files and an arbitrary number of users and servers which leverages private cache to achieve demand privacy among users. Interestingly, it is shown that the inclusion of user-to-user privacy incurs no extra download cost in the large memory regime when comparing to MuPIR.**

I. Introduction

Private Information Retrieval (PIR) [1] seeks efficient ways for a user to privately retrieve a file from a set of distributed and noncolluding servers while hiding the identity of the requested file from any individual server (referred to as *server privavy*). The information-theoretic capacity of PIR was shown to be $(\sum_{i=1}^{K} 1/N^{i-1})^{-1}$, with N and K being the number of servers and files respectively, by Sun and Jafar [2]. Later, many variants of PIR have been proposed and studied. This includes PIR with colluding servers [3] where subsets of servers can communicate, coded and/or storage-constrained PIR [4]–[6] where the servers store MDS-encoded files or the server is storage-limited and can only store a fraction of the file library, symmetric PIR [7] where the user gains no knowledge beyond the requested file, and multi-message PIR [8] to explore joint retrieval benefits, etc. In addition, the role of user-side cache (sometimes referred to as side information) has been investigated in cache-aided PIR systems. For example, single-user cache-aided PIR has been studied under various types of restrictions on the form of the cache content (coded/uncoded/linear combinations of files, known/unknown by servers etc.) [9]–[17]. Moreover, cache-aided multiuser PIR (MuPIR) [18] takes advantage of the (privacy-constrained) global caching gain identified in coded caching [19], [20] to improve retrieval efficiency due to the inclusion of multiple users.

In addition to server privacy in PIR, user-to-user privacy was considered in private coded caching [21]–[23] with multiple users and a single server where each user is prevented from learning the demands of other users. Private coded caching relies on private cache, i.e., the cache placement is carried out in a way such that each user does not know the cached contents of other users, to achieve user-to-user privacy. Although server privacy and user-to-user privacy has been studied separately in either the single-user multiserver setting (i.e., PIR) or the multiuser single-server setting (i.e., coded caching), it is unclear how to *optimally* incorporate user-to-user privacy into MuPIR in order to design efficient MuPIR-U algorithms. MuPIR-U is of particular interest in the design of large-scale information retrieval systems where requests of clients should be protected from both the database servers and other untrusted clients. Unfortunately, the original MuPIR scheme proposed in [18] does not naturally achieve user-to-user privacy due to the exposure of the queries/answers to the users and each user's awareness of the global cache placement. To illustrate, let us look at an example with two files $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$, two users and two servers that achieves the memory-load pair $(2/3, 1)$. User 1 and 2 store $\{a_1, b_1\}$ and $\{a_2, b_2\}$ respectively in the cache placement phase[1]. In the private delivery phase, each server $n \in \{1, 2\}$ generates an answer $A_n^{[(\theta_1,\theta_2)]}$ as follows:

$$\begin{aligned} A_1^{[(\theta_1,\theta_2)]} &= (u_1, u_2, 1)(a_1, a_2, a_3)^{\mathrm{T}} + (v_1, v_2, 1)(b_1, b_2, b_3)^{\mathrm{T}}, \\ A_2^{[(\theta_1,\theta_2)]} &= \left[(g_1, g_2, 1)(a_1, a_2, a_3)^{\mathrm{T}}, (h_1, h_2, 1)(b_1, b_2, b_3)^{\mathrm{T}}\right] \end{aligned} \tag{1}$$

where $u_i, v_i, g_i, h_i, \forall i$ are the (binary) query coefficients which are chosen by the users based on the their demands $(\theta_1, \theta_2) \in \{1, 2\}^2$. For example, the coefficients corresponding to $(\theta_1, \theta_2) = (1, 2)$ are chosen in a way that 1) $u_1 = g_1$ and $v_2 = h_2$ are chosen randomly and independently from

[1]It is assumed that each user knows which bits are stored by the other users although it may not know the exact values of these bits. For example, user 1 knows that the second bit of both files are stored by user 2, but it does not know the values of a_2 and b_2.

$\{0,1\}$, and 2) (u_2, g_2) and (v_1, h_1) are chosen as two random permutations of $(0,1)$. Because the queries and answers (1) are fully observable by both users, it leaks information about the other user's demand to each user. More specifically, from user 1's view, due to the specific alignment of the coefficients between the servers, it knows that b_1 and b_3 are decodable by user 2 but a_1 and a_3 are not. Hence, user 1 knows that user 2 wants file B, implying that the above MuPIR scheme fails to guarantee user-to-user privacy. More generally, each user's knowledge of other users' cached contents, combined with its full observation of the queries and answers, helps the user to infer the demands of other users. Therefore, the MuPIR scheme of [18] does not guarantee user-to-user privacy.

In this paper, we propose *cache-aided MuPIR with user privacy* (MuPIR-U), a novel problem formulation which takes into consideration both server and user-to-user privacy in multiuser PIR systems. In the proposed model (See Fig. 1), multiple cache-equipped users are connected to multiple servers with a replicated file library. The retrieval process consists of two phases, the cache placement phase where the users fulfill their cache memories without knowing their future demands of the files, and the private delivery phase where the user's demands are revealed and collected by a trusted server (through private upload links) which then generates a set of queries for the servers. Each server responds with an answer that is broadcast to all users. With the received answers and its cache, each user should be able to recover its desired file. Both server and user-to-user privacy should be enforced, that is, we aim to design delivery schemes with minimal download cost while ensuring that 1) each server should be prevented from knowing the joint demands of the users, and 2) each user should not be able to infer the demands of other users. For the case of two files and an arbitrary number of users and servers, we present a novel cache placement and private delivery scheme that achieves the optimal load when the memory size is above a certain threshold. To achieve user-to-user privacy, we leverage techniques used in private coded caching [21] to hide the stored contents of each user from other users so that each potential cache configuration corresponds to a different set of user demands, therefore hiding the the remaining demands from each user's perspective.

Notation: For $i \leq j$, denote $[i:j] \triangleq \{i, i+1, \cdots, j\}$, $A_{i:j} \triangleq \{A_i, \cdots, A_j\}$ and $A_{(i:j)} \triangleq (A_i, \cdots, A_j)$. Write $[1:j]$ as $[j]$ for brevity. Let $\mathbf{S}_n \triangleq (\mathcal{S}_i)_{i=1}^{2^n}$ denote a vector containing all the subsets of $[n]$ arranged in lexicographic order. Denote $\varphi_n(\mathcal{S}_i) \triangleq i$ as the index of $\mathcal{S}_i$ in $\mathbf{S}_n$. For example, if $n=2$, we have $\mathcal{S}_1 = \emptyset, \mathcal{S}_2 = \{1\}, \mathcal{S}_3 = \{1,2\}, \mathcal{S}_4 = \{2\}$ and thus $\mathbf{S}_2 = (\emptyset, \{1\}, \{1,2\}, \{2\})$, $\varphi_2(\emptyset) = 1, \varphi_2(\{1\}) = 2$ etc. An (n,k)-MDS generator matrix $\mathbf{M} \in \mathbb{R}^{n \times k}$ has the property that every (k,k) submatrix of $\mathbf{M}$ is invertible.

II. Problem Formulation

Consider a (K, K_u, N) cache-aided MuPIR system as shown in Fig. 1 which consists of N servers each holding a library of K independent and equally sized files $W_1, \cdots, W_K$

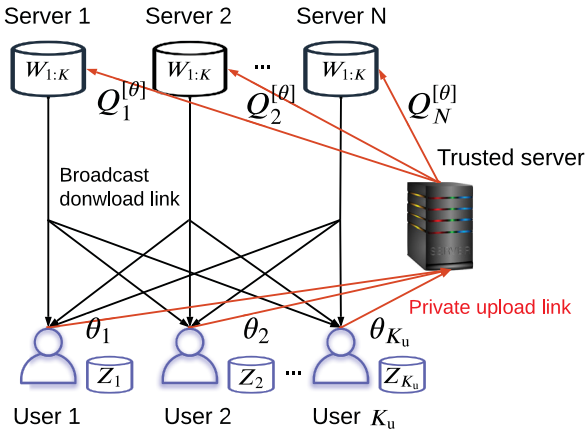

Fig. 1. System model of cache-aided MuPIR-U. In the private delivery phase, the trusted server collects the users' demands through private upload links and then generate a query for each server.

where $H(W_k) = L, \forall k$, $H(W_{1:K}) = \sum_{k=1}^{K} H(W_k)$, and K_u cache-equipped users each may request a random file. Each server is connected to the users through an error-free broadcast channel. Each user $k \in [K_\mathrm{u}]$ is equipped with a cache memory Z_k that can store up to ML bits of data. The system works in two phases, a cache placement phase followed by a private delivery phase. In the placement phase, the users fill up their cache memory as a function of the files $W_{1:K}$ without the knowledge of their future demands. It is assumed that the cached contents of the users are known by the servers but each user's cache is *kept secret* from other users[2]. In the private delivery phase, the users reveal their file requests represented by the demand vector $\boldsymbol{\theta} \triangleq (\theta_1, \cdots, \theta_{K_\mathrm{u}})$ where user k wants $W_{\theta_k}, \forall k$. In order to retrieve the desired files, a trusted server first collects each user's demand through a private upload link and then generates a set of queries $Q_1^{[\boldsymbol{\theta}]}, \cdots, Q_N^{[\boldsymbol{\theta}]}$ each to be sent to a different server.[3] Upon receiving $Q_n^{[\boldsymbol{\theta}]}$, server n responds with an answer $A_n^{[\boldsymbol{\theta}]}$ that is broadcast to all users. Using the answers downloaded from the servers and the cached content, each user k should be able to recover its desired file, which expressed in terms of conditional entropy, is

$$H\left(W_{\theta_k} | Q_{1:N}^{[\boldsymbol{\theta}]}, A_{1:N}^{[\boldsymbol{\theta}]}, Z_k\right) = 0, \ \forall k \in [K_\mathrm{u}]. \tag{2}$$

Meanwhile, the server and user-to-user privacy constraints should be satisfied. Server privacy requires the user demands to be protected from any individual server, i.e., $\boldsymbol{\theta}$ should be independent of the queries and answers seen by each server, i.e.,

$$I\left(\boldsymbol{\theta}; Q_n^{[\boldsymbol{\theta}]}, A_n^{[\boldsymbol{\theta}]} | W_{1:K}, Z_{1:K_\mathrm{u}}\right) = 0, \ \forall n \in [N] \tag{3}$$

[2]This can be achieved by employing a random cache placement for the users such that each user is unable to know the cache of other users.

[3]The query generation procedure is further explained in Remark 1.

The conditioning on $Z_1, \cdots, Z_{K_\mathrm{u}}$ reflects the assumption of the servers' awareness of the users' cache. User-to-user privacy requires that each user should be prevented from inferring other users' demands, i.e.,

$$I\Big(\boldsymbol{\theta}_{\setminus\{k\}}; Q_{1:N}^{[\boldsymbol{\theta}]}, A_{1:N}^{[\boldsymbol{\theta}]} \big| \theta_k, Z_k\Big) = 0, \ \forall k \in [K_\mathrm{u}] \tag{4}$$

where $\boldsymbol{\theta}_{\setminus\{k\}} \triangleq (\theta_i)_{i\neq k}$ denotes the demands of all users except user k. It is assumed that the queries from the trusted server and the answers from the servers are fully accessible by each user as implied by (4).

Let $D = \sum_{n=1}^{N} H(A_n^{[\boldsymbol{\theta}]})$ denote the total number of bits downloaded from the servers. The *load*, denoted by R, is defined as the normalized download cost, i.e., $R \triangleq D/L$. A memory-load pair (M, R) is said to be achievable if there exists a cache-aided MuPIR-U scheme that achieves the load R at memory size M. Let $R^\star$ denote the minimum achievable load. We aim to design the cache placement and private delivery phases to achieve a load as small as possible while ensuring both server and user privacy.

Remark 1 (Trusted server & query generation): The reason that we employ a trusted server to perform query generation is explained as follows. The first thing to notice is that the queries have to be generated based on the users' joint demands and their cached contents. Hence, as opposed to MuPIR [18] where the users cooperatively generate the queries, here the users are not suitable to perform query generation as they are not allowed to collect other users' demands. As a result, a trusted server is used to collect the users' individual demands via private upload links (See Fig. 1) so that $\boldsymbol{\theta}_{\setminus\{k\}}$ is kept secret from user $k, \forall k$. The trusted server also knows the users' cache which is necessary for query generation. One more thing to notice is that the set of queries $\{Q_n^{[\boldsymbol{\theta}]}\}_{n=1}^N$ and answers $\{A_n^{[\boldsymbol{\theta}]}\}_{n=1}^N$ are fully accessible to the users because each user cannot infer other users' demands by solely observing the queries/answers without knowing their cache.

III. Main Result

Theorem 1: For cache-aided MuPIR-U with two files, $K_\mathrm{u} \geq 2$ users and $N \geq 2$ servers, the memory-load pair

$$(M^+, R^+) = \left(\frac{2^{K_\mathrm{u}}(N-1)}{L}, \frac{N+1}{L}\right) \tag{5}$$

where $L \triangleq (2^{K_\mathrm{u}-1}+1)(N-1)+1$ is achievable. When $M \geq M^+$, the optimal load is given by

$$R^\star = (1-M/2)(1+1/N), \ \forall M \in [M^+, 2]. \tag{6}$$

Proof: The load of (6) can be achieved by memory sharing [19] between the achievable points (M^+, R^+) and $(2, 0)$ which is trivially achievable. Conversely, the optimal load $R^\star$ should be lower bounded by $R^\star \geq \widehat{R} \triangleq (1-M/2)(1+1/N)$ where $\widehat{R}$ denotes the optimal load for cache-aided PIR with a single user [9]. Because the multiuser privacy requirement (3) implies the demand privacy for each user, and the incorporation of user-to-user privacy (4) can only incur possibly higher load, any converse bound of [9] is also a valid converse for MuPIR-U. Since the achievable load (6) matches the lower bound $\widehat{R}$ when $M \geq M^+$, it is optimal. In Section IV, we present a novel scheme that achieves the memory-load pair (M^+, R^+). ■

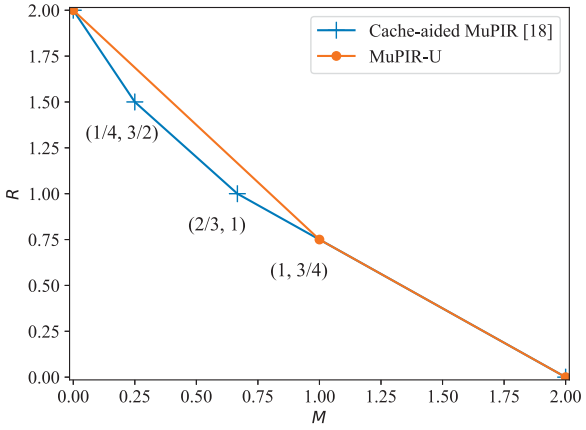

Fig. 2. Comparison with MuPIR for $(K, K_\mathrm{u}, N) = (2, 2, 2)$.

Remark 2: Theorem 1 revealed an interesting fact that there is no extra penalty on the optimal load when adding user-to-user privacy to MuPIR if the memory size is above a certain threshold. In addition, memory sharing between (M^+, R^+) and the trivially-achievable point $(0, 2)$ implies the achievability of $R = 2 - (1 + 1/2^{K_\mathrm{u}})M$, $M \in [0, M^+]$. A comparison of the achievable load with the optimal load of cache-aided MuPIR [18] for $(K, K_\mathrm{u}, N) = (2, 2, 2)$ is given in Fig. 2.

IV. Proposed Scheme

In this section, we present the proposed scheme which achieves the memory-load pair (5) in Theorem 1. An example with 2 users and 2 servers is provided to highlight the design idea before proceeding to the description of the general scheme for arbitrary K_u and N.

A. A Motivating Example

Example 1: Consider $(K, K_\mathrm{u}, N) = (2, 2, 2)$ for which we show that $(1, 3/4)$ is achievable. Let A and B denote the two files each consisting of $L = 4$ bits, i.e., $A = (A_1, A_2, A_3, A_4), B = (B_1, B_2, B_3, B_4)$. Let $\mathbf{M}$ be a (binary) $(4, 3)$-MDS generator matrix defined as

$$\mathbf{M} \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{7}$$

Encode the files using the above MDS matrix as follows:

$$\begin{bmatrix} \widetilde{A}_1 \\ \widetilde{A}_2 \\ \widetilde{A}_3 \\ \widetilde{A}_4 \end{bmatrix} = \mathbf{M} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}, \quad \begin{bmatrix} \widetilde{B}_1 \\ \widetilde{B}_2 \\ \widetilde{B}_3 \\ \widetilde{B}_4 \end{bmatrix} = \mathbf{M} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}. \tag{8}$$

Note that any three out of the four encoded bits can recover the three original uncoded bits due to the MDS property. Also note that the last bit of each file has not been encoded. The above encoding process (i.e., $\mathbf{M}$) is publicly known by all users and servers.

1) Cache placement phase: The servers first generate an independent and random permutation $\boldsymbol{p}_i \triangleq (p_{i,1}, p_{i,2}, p_{i,3}, p_{i,4})(i = 1, 2)$ of $(1, 2, 3, 4)$ for each file. These permutations are known by the servers but *kept secret from the users*. We define a subfile $A_{\mathcal{S}}$ as a small piece of A that is stored exclusively by a set of users $\mathcal{S} \subseteq [K_{\mathrm{u}}]$ in the placement phase. The MDS-encoded bits are then assigned to the subfiles as follows:

$$\begin{aligned} &A_{\emptyset} = \widetilde{A}_{p_{1,1}}, A_{\{1\}} = \widetilde{A}_{p_{1,2}}, A_{\{1,2\}} = \widetilde{A}_{p_{1,3}}, A_{\{2\}} = \widetilde{A}_{p_{1,4}}; \\ &B_{\emptyset} = \widetilde{B}_{p_{2,1}}, B_{\{1\}} = \widetilde{B}_{p_{2,2}}, B_{\{1,2\}} = \widetilde{B}_{p_{2,3}}, B_{\{2\}} = \widetilde{B}_{p_{2,4}}. \end{aligned} \tag{9}$$

Each user k stores all subfiles $A_{\mathcal{S}}, B_{\mathcal{S}}$ where $k \in \mathcal{S}$, i.e., the user cache (from the servers' view) is

$$\begin{aligned} \text{[Server view]} \quad Z_1 &= \left\{A_{\{1\}}, A_{\{1,2\}}, B_{\{1\}}, B_{\{1,2\}}\right\}, \\ Z_2 &= \left\{A_{\{1,2\}}, A_{\{2\}}, B_{\{1,2\}}, B_{\{2\}}\right\}. \end{aligned} \tag{10}$$

Alternatively, from each user's view, the cache becomes

$$\begin{aligned} \text{[User view]} \quad Z_1 &= \left\{\widetilde{A}_{p_{1,2}}, \widetilde{A}_{p_{1,3}}, \widetilde{B}_{p_{2,2}}, \widetilde{B}_{p_{2,3}}\right\}, \\ Z_2 &= \left\{\widetilde{A}_{p_{1,3}}, \widetilde{A}_{p_{1,4}}, \widetilde{B}_{p_{2,3}}, \widetilde{B}_{p_{2,4}}\right\}. \end{aligned} \tag{11}$$

Because the users *do not* know the permutations $\boldsymbol{p}_1, \boldsymbol{p}_2$, each user only sees two randomly chosen MDS-encoded bits of each file in its cache. For example, from user 1's view, the cache is $Z_1 = \{\widetilde{A}_{p_{1,2}}, \widetilde{A}_{p_{1,3}}, \widetilde{B}_{p_{2,2}}, \widetilde{B}_{p_{2,3}}\}$. User 1 knows it has stored $\widetilde{A}_{p_{1,2}}$ and $\widetilde{A}_{p_{1,3}}$, but cannot tell which of them is assigned to which of $A_{\{1\}}$ and $A_{\{1,2\}}$. In this case, we say that user 1 *cannot distinguish* $A_{\{1\}}$ *from* $A_{\{1,2\}}$. User 1 cannot distinguish $B_{\{1\}}$ from $B_{\{1,2\}}$ either. Similarly, user 2 cannot distinguish $A_{\{1,2\}}$ from $A_{\{2\}}$, and $B_{\{1,2\}}$ from $B_{\{2\}}$ respectively. A direct consequence of this inability to identify the stored subfiles is that each user does know which MDS-encoded bits have been stored by the other user, which is crucial to achieving user-to-user privacy. Since each MDS symbol has only one bit, we have $M = 1$.

2) Private delivery phase: The task of this phase is to design the query to each server in order for the users to correctly recover their requested files as well as ensuring both server and user-to-user privacy. In particular, the answers of the servers take the form of linear combinations of the (MDS-encoded and uncoded) bits of the files. The answer of server 1 is a linear combination $A_1^{[\boldsymbol{\theta}]} = \sum_{i=1}^{4} \left(u_i \widetilde{A}_i + v_i \widetilde{B}_i\right) + A_4 + B_4$, and the answer of server 2 consists of two linear combinations $A_2^{[\boldsymbol{\theta}]} = (A_{2,1}^{[\boldsymbol{\theta}]}, A_{2,2}^{[\boldsymbol{\theta}]})$ where $A_{2,1}^{[\boldsymbol{\theta}]} = \sum_{i=1}^{4} g_i \widetilde{A}_i + A_4$ and $A_{2,2}^{[\boldsymbol{\theta}]} = \sum_{i=1}^{4} h_i \widetilde{B}_i + B_4$. These answers can be written in a matrix form as

$$\begin{bmatrix} A_1^{[\boldsymbol{\theta}]} \\ A_{2,1}^{[\boldsymbol{\theta}]} \\ A_{2,2}^{[\boldsymbol{\theta}]} \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & 1 & v_1 & v_2 & v_3 & v_4 & 1 \\ g_1 & g_2 & g_3 & g_4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_1 & h_2 & h_3 & h_4 & 1 \end{bmatrix} \begin{bmatrix} \widetilde{A}_{(1:4)}^{\mathrm{T}} \\ A_4 \\ \widetilde{B}_{(1:4)}^{\mathrm{T}} \\ B_4 \end{bmatrix}. \tag{12}$$

The binary coefficients $u_i, v_i, g_i, h_i, \forall i$ need to be chosen depending on $\boldsymbol{\theta}$. Recall that $\widetilde{A}_{(1:4)} \triangleq (\widetilde{A}_1, \cdots, \widetilde{A}_4)$. The query structure of (12) is fixed regardless of $\boldsymbol{\theta}$, i.e., the coefficients u_i and v_i always correspond to the MDS-encoded bits $\widetilde{A}_i$ and $\widetilde{B}_i$ respectively, $\forall i \in [4]$. This fixed query structure is necessary to achieve privacy as will be shown later. Since three bits are downloaded, the achieved load is $R = 3/4$.

TABLE I
CHOICE OF COEFFICIENTS UNDER DIFFERENT $\boldsymbol{\theta}$

$\boldsymbol{\theta}$	$(1,1)$	$(1,2)$	$(2,1)$	$(2,2)$
$\widetilde{A}_1$	$u_1 \neq g_1$	$u_1 = g_1$	$u_1 = g_1$	$u_1 = g_1$
$\widetilde{A}_2$	$u_2 = g_2$	$u_2 = g_2$	$u_2 \neq g_2$	$u_2 = g_2$
$\widetilde{A}_3$	$u_3 = g_3$	$u_3 = g_3$	$u_3 = g_3$	$u_3 \neq g_3$
$\widetilde{A}_4$	$u_4 = g_4$	$u_4 \neq g_4$	$u_4 = g_4$	$u_4 = g_4$
$\widetilde{B}_1$	$v_1 = h_1$	$v_1 = h_1$	$v_1 = h_1$	$v_1 \neq h_1$
$\widetilde{B}_2$	$v_2 = h_2$	$v_2 \neq h_2$	$v_2 = h_2$	$v_2 = h_2$
$\widetilde{B}_3$	$v_3 \neq h_3$	$v_3 = h_3$	$v_3 = h_3$	$v_3 = h_3$
$\widetilde{B}_4$	$v_4 = h_4$	$v_4 = h_4$	$v_4 \neq h_4$	$v_4 = h_4$

Suppose $\boldsymbol{\theta} = (1, 2)$. To illustrate the query design, without loss of generality, let us assume $\boldsymbol{p}_1 = \boldsymbol{p}_2 = (1, 2, 3, 4)$. A specific mapping can then be established in (9) and the user cache becomes $Z_1 = \{\widetilde{A}_2, \widetilde{A}_3, \widetilde{B}_2, \widetilde{B}_3\}$, $Z_2 = \{\widetilde{A}_3, \widetilde{A}_4, \widetilde{B}_3, \widetilde{B}_4\}$. The query coefficients are chosen as follows: Let $u_1 = g_1, u_2 = g_2, u_3 = g_3, v_1 = h_1, v_3 = h_3, v_4 = h_4$ be chosen randomly and independently from $\{0, 1\}$. Also let (u_4, g_4) and (v_2, h_2) be chosen as two independent random permutations of $(0, 1)$. With such a choice, we show that the users can correctly recover their desired files while maintaining server and user-to-user privacy at the same time.

Decodability. Let us look at user 1. First, by subtracting $A_{2,1}^{[\boldsymbol{\theta}]}$ and $A_{2,2}^{[\boldsymbol{\theta}]}$ from $A_1^{[\boldsymbol{\theta}]}$, both users obtain $\widetilde{A}_4 + \widetilde{B}_2$ (i.e., $A_{\{2\}} + B_{\{1\}}$ from the server's view). Since $\widetilde{B}_2 \in Z_1$, user 1 can decode $\widetilde{A}_4$. Now user 1 has $\widetilde{A}_2, \widetilde{A}_3, \widetilde{A}_4$, from which A_1, A_2, A_3 can be recovered due to the MDS property. $\widetilde{A}_1$ can also be decoded by repeating the MDS encoding (8) one more time. Moreover, removing all $\widetilde{A}_i, i \in [4]$ from $A_{2,1}^{[\boldsymbol{\theta}]}$, user 1 decodes A_4. Hence, user 1 correctly decodes all the 4 bits of file A. Similarly, user 2 can decode $\widetilde{B}_2$ from $\widetilde{A}_4 + \widetilde{B}_2$ since $\widetilde{A}_4 \in Z_2$. With $\widetilde{B}_2, \widetilde{B}_3, \widetilde{B}_4$, it can recover B_1, B_2, B_3 and $\widetilde{B}_1$. By removing all $\widetilde{B}_i, i \in [4]$ from $A_{2,2}^{[\boldsymbol{\theta}]}$, user 2 obtains B_4. Therefore, both users can recover their desired files. For other user demands, the choice of coefficients are listed in Table I.[4] The general rule to choose the query coefficients under arbitrary permutations $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is that a specific summation of two MDS-encoded bits can be recovered by

[4]For the two coefficients corresponding to the same MDS-encoded bit in (12), if they are equal, the value is chosen randomly from $\{0, 1\}$; if they are not equal, they are chosen as a random permutation of $(0, 1)$.

subtracting $A_{2,1}^{[\boldsymbol{\theta}]}+A_{2,2}^{[\boldsymbol{\theta}]}$ from $A_1^{[\boldsymbol{\theta}]}$. In particular, the recovered summation should be $A_{\{2\}}+B_{\{1\}}$ for $\boldsymbol{\theta}=(1,2)$, $A_\emptyset+B_{\{1,2\}}$ for $\boldsymbol{\theta}=(1,1)$, $A_{\{1\}}+B_{\{2\}}$ for $\boldsymbol{\theta}=(2,1)$ and $A_{\{1,2\}}+B_\emptyset$ for $\boldsymbol{\theta}=(2,2)$.

Server privacy. Because each server does not know the query sent to the other server, it does not know which coefficients are aligned (i.e., being equal) among the two servers and which are not. For whichever alignment pattern, the coefficients $\{u_i, v_i, \forall i\}$ appear to be randomly i.i.d. over $\{0,1\}$ from the view of server 1, regardless of $\boldsymbol{\theta}$. Therefore, server 1 cannot determine $\boldsymbol{\theta}$. Similarly, server 2 cannot determine $\boldsymbol{\theta}$ either.

User-to-user privacy. We show that the above scheme is private from user 1's perspective, i.e., user 1 cannot determine θ_2 by observing the queries/answers. Due to the unawareness of the random permutations generated by the servers, user 1 cannot distinguish between the two stored subfiles $\widetilde{A}_{p_{1,2}}$ (i.e., $A_{\{1\}}$ from server view) and $\widetilde{A}_{p_{1,3}}$ ($A_{\{1,2\}}$), and between the two unstored subfiles $\widetilde{A}_{p_{1,1}}$ ($A_\emptyset$) and $\widetilde{A}_{p_{1,4}}$ ($A_{\{2\}}$). For the case of $\theta_1=1$, the query coefficients are chosen to recover $A_{\{2\}}+B_{\{1\}}$ if $\theta_2=2$, and $A_\emptyset+B_{\{1,2\}}$ if $\theta_2=1$ (See Table I). Intuitively, user 1 cannot distinguish $A_{\{2\}}+B_{\{1\}}$ from $A_\emptyset+B_{\{1,2\}}$ as it cannot distinguish $A_{\{2\}}$ from $A_\emptyset$, and $B_{\{1\}}$ from $B_{\{1,2\}}$. Therefore, user 1 cannot tell if the query is generated for $\boldsymbol{\theta}=(1,2)$ or $(1,1)$, implying its ignorance of θ_2. For $\theta_1=2$, the summation to be recovered is $A_{\{1\}}+B_{\{2\}}$ if $\theta_2=1$, and $A_{\{1,2\}}+B_\emptyset$ if $\theta_2=2$. Again, since user 1 cannot distinguish $A_{\{1\}}$ from $A_{\{1,2\}}$, and $B_{\{2\}}$ from $B_\emptyset$, it cannot distinguish between $A_{\{1\}}+B_{\{2\}}$ and $A_{\{1,2\}}+B_\emptyset$. This implies that user 1 cannot determine if the demand vector is $(2,1)$ or $(2,2)$. As a result, the above scheme is private from user 1's view. It can be verified that the scheme is also private from user 2's view. $\diamondsuit$

Remark 3: On a higher level, server privacy in Example 1 is achievable due to each server's unawareness of the alignment pattern of the query coefficients between the two servers, which is the key ingredient for PIR to achieve privacy. User-to-user privacy is achievable due to each user's unawareness of the random permutations $\boldsymbol{p}_i$'s applied to different files. In particular, the unawareness of the mapping (9) from the MDS-encoded bits to the subfiles means that each user does not know what the cached contents of remaining users are. From any user's view, each different cache realization of the remaining users can possibly correspond to a different sets of demands. Therefore, that user is unable to infer other users' demands. This is a technique used in private coded caching [21] to achieve inter-user privacy. Here, we provide an *optimal* way to combine the existing techniques to achieve server and user-to-user privacy simultaneously.

B. General Scheme

In this section, we present the scheme to achieve (M^+, R^+) in Theorem 1 for two files and an arbitrary number of users and servers. The proposed scheme leverages MDS encoding and is formally described as follows.

Let W_1, W_2 be the two files each consisting of $L=(2^{K_\mathrm{u}-1}+1)(N-1)+1$ symbols over some finite field $\mathbb{F}_q$. For ease of notation, let $U\triangleq 2^{K_\mathrm{u}-1}+1$. W_k is split into $U+1$ smaller pieces, i.e., $W_k=(\boldsymbol{W}_{k,1},\cdots,\boldsymbol{W}_{k,U},W_{k,U+1}), k\in\{1,2\}$. Each of the first U pieces contains $N-1$ symbols, i.e., $\boldsymbol{W}_{k,i}\triangleq(W_{k,i}(1),\cdots,W_{k,i}(N-1)), \forall i\in[U]$ for which $W_{k,i}(j)\in\mathbb{F}_q, \forall j$. The last piece $W_{k,U+1}$ contains only one symbol. We first prepare the files through MDS encoding.

File preparation: An MDS encoding is applied to the first U pieces of each file:

$$\big(\widetilde{\boldsymbol{W}}_{k,1},\cdots,\widetilde{\boldsymbol{W}}_{k,2^{K_\mathrm{u}}}\big)^{\mathrm{T}}=\mathbf{M}(\boldsymbol{W}_{k,1},\cdots,\boldsymbol{W}_{k,U})^{\mathrm{T}} \tag{13}$$

where $\mathbf{M}$ is a $(2^{K_\mathrm{u}},U)$-MDS generator matrix whose elements are chosen from some finite field[5]. Each MDS-encoded piece $\widetilde{\boldsymbol{W}}_{k,i}, i\in[2^{K_\mathrm{u}}]$ contains $N-1$ symbols from $\mathbb{F}_q$. Note that the last piece $W_{k,U+1}$ is not encoded. After the MDS encoding, the servers generate an independent random permutation $\boldsymbol{p}_k\triangleq(p_{k,1},\cdots,p_{k,2^{K_\mathrm{u}}})$ of $(1:2^{K_\mathrm{u}})$ for each file $W_k,\forall k$. These permutations are *kept secret from the users but known by all servers and the trusted server.* The MDS-encoded pieces are then assigned to the subfiles $\{W_{k,\mathcal{S}},\mathcal{S}\subseteq[K_\mathrm{u}]\}$ according to

$$W_{k,\mathcal{S}}=\widetilde{\boldsymbol{W}}_{k,p_{k,\varphi_{K_\mathrm{u}}(\mathcal{S})}},\ \forall\mathcal{S}\subseteq\big[2^{K_\mathrm{u}}\big]. \tag{14}$$

Recall that $\varphi_{K_\mathrm{u}}(\mathcal{S})$ denotes the index of the subset $\mathcal{S}$ in the power set of $[K_\mathrm{u}]$ arranged in a lexicographic order.

1) Cache placement phase: Each user u stores all subfiles $W_{k,\mathcal{S}}$ if $u\in\mathcal{S}$, i.e., $Z_u=\{W_{k,\mathcal{S}},\forall\mathcal{S}\text{ s.t. }u\in\mathcal{S},k\in\{1,2\}\}$. Hence, the memory size is $M=2\sum_{i=0}^{K_\mathrm{u}-1}\binom{K_\mathrm{u}-1}{i}(N-1)/L=2^{K_\mathrm{u}}(N-1)/(U(N-1)+1)$. Note that each user stores $2^{K_\mathrm{u}-1}$ MDS-encoded pieces of each file, but does not know which of them are assigned to which subfiles due to its unawareness of the server-generated random permutations.

2) Private delivery phase: The answers of the servers are linear combinations of the file pieces as given by (15). $\boldsymbol{u}_i^n,\boldsymbol{v}_i^n,\boldsymbol{g}_i,\boldsymbol{h}_i\in\{0,1\}^{1\times(N-1)}$ represent the binary coefficient vectors. Each server n's answer has one symbol $A_n^{[\boldsymbol{\theta}]}=\sum_{i=1}^{2^{K_\mathrm{u}}}\big(\boldsymbol{u}_i^n\widetilde{\boldsymbol{W}}_{1,i}^{\mathrm{T}}+\boldsymbol{v}_i^n\widetilde{\boldsymbol{W}}_{2,i}^{\mathrm{T}}\big)+W_{1,U+1}+W_{2,U+1},\forall n\in[N-1]$ and server N's answer has two symbols: $A_N^{[\boldsymbol{\theta}]}=(A_{N,1}^{[\boldsymbol{\theta}]},A_{N,2}^{[\boldsymbol{\theta}]})$ where $A_{N,1}^{[\boldsymbol{\theta}]}=\sum_{i=1}^{2^{K_\mathrm{u}}}\boldsymbol{g}_i\widetilde{\boldsymbol{W}}_{1,i}^{\mathrm{T}}+W_{1,U+1}$ and $A_{N,2}^{[\boldsymbol{\theta}]}=\sum_{i=1}^{2^{K_\mathrm{u}}}\boldsymbol{h}_i\widetilde{\boldsymbol{W}}_{2,i}^{\mathrm{T}}+W_{2,U+1}$. Since a total of $N+1$ linear combinations (each containing one symbol over $\mathbb{F}_q$) are downloaded, the achieved load is $R=(N+1)/(U(N-1)+1)$.

Suppose $\boldsymbol{\theta}=(\theta_1,\cdots,\theta_{K_\mathrm{u}})$. Denote $\mathcal{D}_k$ as the set of users which request file W_k, i.e., $\mathcal{D}_k\triangleq\{u\in[K_\mathrm{u}]:\theta_u=k\},k\in\{1,2\}$. Denote

$$\mathbf{Y}_n\triangleq\begin{bmatrix}\mathbf{I}_n\\ \mathbf{0}\end{bmatrix}\in\mathbb{F}_2^{(n+1)\times n} \tag{16}$$

as a binary matrix consisting of the $n\times n$ identify matrix $\mathbf{I}_n$ and an extra row of zeros. Let $\mathcal{Y}_n$ denote a set containing all the $n+1$ rows of $\mathbf{Y}_n$. Let $i_k\triangleq\varphi_{K_\mathrm{u}}([K_\mathrm{u}]\backslash\mathcal{D}_k), k=1,2$, i.e.,

[5]The existence of $\mathbf{M}$ is guaranteed if the field size is large enough.

$$\begin{bmatrix} A_1^{[\boldsymbol{\theta}]} \\ A_2^{[\boldsymbol{\theta}]} \\ \vdots \\ A_{N-1}^{[\boldsymbol{\theta}]} \\ A_{N,1}^{[\boldsymbol{\theta}]} \\ A_{N,2}^{[\boldsymbol{\theta}]} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{u}_1^1 & \boldsymbol{u}_2^1 & \cdots & \boldsymbol{u}_{2^{K_\mathrm{u}}}^1 & 1 & \boldsymbol{v}_1^1 & \boldsymbol{v}_2^1 & \cdots & \boldsymbol{v}_{2^{K_\mathrm{u}}}^1 & 1 \\ \boldsymbol{u}_1^2 & \boldsymbol{u}_2^2 & \cdots & \boldsymbol{u}_{2^{K_\mathrm{u}}}^2 & 1 & \boldsymbol{v}_1^2 & \boldsymbol{v}_2^2 & \cdots & \boldsymbol{v}_{2^{K_\mathrm{u}}}^2 & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \boldsymbol{u}_1^{N-1} & \boldsymbol{u}_2^{N-1} & \cdots & \boldsymbol{u}_{2^{K_\mathrm{u}}}^{N-1} & 1 & \boldsymbol{v}_1^{N-1} & \boldsymbol{v}_2^{N-1} & \cdots & \boldsymbol{v}_{2^{K_\mathrm{u}}}^{N-1} & 1 \\ \boldsymbol{g}_1 & \boldsymbol{g}_2 & \cdots & \boldsymbol{g}_{2^{K_\mathrm{u}}} & 1 & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & 0 & \boldsymbol{h}_1 & \boldsymbol{h}_2 & \cdots & \boldsymbol{h}_{2^{K_\mathrm{u}}} & 1 \end{bmatrix}}_{(N+1)\times L} \begin{bmatrix} \widetilde{W}_{1,1}^{\mathrm{T}} \\ \vdots \\ \widetilde{W}_{1,2^{K_\mathrm{u}}}^{\mathrm{T}} \\ W_{1,U+1} \\ \widetilde{W}_{2,1}^{\mathrm{T}} \\ \vdots \\ \widetilde{W}_{2,2^{K_\mathrm{u}}}^{\mathrm{T}} \\ W_{2,U+1} \end{bmatrix} \tag{15}$$

i_k denotes the lexicographic index of the subset $[K_\mathrm{u}]\backslash\mathcal{D}_k$ of $[K_\mathrm{u}]$. The query coefficients are chosen as follows: Let

$$\begin{bmatrix} \boldsymbol{u}_{i_1}^1 \\ \vdots \\ \boldsymbol{u}_{i_1}^{N-1} \\ \boldsymbol{g}_{i_1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{v}_{i_2}^1 \\ \vdots \\ \boldsymbol{v}_{i_2}^{N-1} \\ \boldsymbol{h}_{i_2} \end{bmatrix} \tag{17}$$

be chosen independently as two random row permutations of $\mathbf{Y}_{N-1}$. Also let $\boldsymbol{g}_i = \boldsymbol{u}_i^n, \forall n \in [N-1], \forall i \neq i_1$ and $\boldsymbol{h}_i = \boldsymbol{v}_i^n, \forall n \in [N-1], \forall i \neq i_2$ be chosen randomly and independently from $\mathcal{Y}_{N-1}$. The purpose of such a choice of the query coefficients is to recover a linear combination of $\widetilde{\boldsymbol{W}}_{1,i_1}$ and $\widetilde{\boldsymbol{W}}_{2,i_2}$ from which each user in $\mathcal{D}_1$ can decode $\widetilde{\boldsymbol{W}}_{1,i_1}$ while each user in $\mathcal{D}_2$ can decode $\widetilde{\boldsymbol{W}}_{2,i_2}$. Then with the aid of the cache, the users can recover their desired files which is proved as follows.

Decodability. First of all, each user can obtain $N-1$ linear equations by subtracting $A_{N,1}^{[\boldsymbol{\theta}]} + A_{N,2}^{[\boldsymbol{\theta}]}$ from each $A_n^{[\boldsymbol{\theta}]}, n \in [N-1]$, which are

$$\begin{aligned} &A_n^{[\boldsymbol{\theta}]} - \left(A_{N,1}^{[\boldsymbol{\theta}]} + A_{N,2}^{[\boldsymbol{\theta}]}\right) = \\ &\left(\boldsymbol{u}_{i_1}^n - \boldsymbol{g}_{i_1}\right)\widetilde{\boldsymbol{W}}_{1,i_1}^{\mathrm{T}} + \left(\boldsymbol{v}_{i_2}^n - \boldsymbol{h}_{i_2}\right)\widetilde{\boldsymbol{W}}_{2,i_2}^{\mathrm{T}}, \forall n \in [N-1]. \end{aligned} \tag{18}$$

Note that $W_{1,[K_\mathrm{u}]\backslash\mathcal{D}_1} = \widetilde{\boldsymbol{W}}_{1,i_1}$, $W_{2,[K_\mathrm{u}]\backslash\mathcal{D}_2} = \widetilde{\boldsymbol{W}}_{2,i_2}$. Now let us look at user 1. Suppose $\theta_1 = 1$. Because $\widetilde{\boldsymbol{W}}_{2,i_2}$ has been stored by user 1 as $1 \in [K_\mathrm{u}]\backslash\mathcal{D}_2$, user 1 can further obtain $N-1$ linear combinations of $\widetilde{\boldsymbol{W}}_{1,i_1}$ which are

$$\begin{bmatrix} y_1 - x_1 \\ \vdots \\ y_{N-1} - x_{N-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_{i_1}^1 - \boldsymbol{g}_{i_1} \\ \vdots \\ \boldsymbol{u}_{i_1}^{N-1} - \boldsymbol{g}_{i_1} \end{bmatrix} \widetilde{\boldsymbol{W}}_{1,i_1}^{\mathrm{T}} \tag{19}$$

where $x_n \triangleq (\boldsymbol{v}_{i_2}^n - \boldsymbol{h}_{i_2})\widetilde{\boldsymbol{W}}_{2,i_2}^{\mathrm{T}}$, $y_n \triangleq A_n^{[\boldsymbol{\theta}]} - (A_{N,1}^{[\boldsymbol{\theta}]} + A_{N,2}^{[\boldsymbol{\theta}]}), n \in [N-1]$. Due to the special choice of the coefficients in (17), it can be easily verified that the coefficient matrix on the RHS of (19) has full rank. Hence, user 1 can decode $\widetilde{\boldsymbol{W}}_{1,i_1}$ by matrix inversion. Together with the $2^{K_\mathrm{u}-1}$ stored subfiles, user 1 now has $U = 2^{K_\mathrm{u}-1}+1$ MDS-encoded pieces of W_1, from which the U original uncoded file pieces $\{\boldsymbol{W}_{1,i}, i \in [U]\}$ can be recovered due to the MDS property. Moreover, user 1 can obtain all the MDS-encoded pieces $\{\widetilde{\boldsymbol{W}}_{1,i}, i \in [2^{K_\mathrm{u}}]\}$ by repeating the encoding process of (13). Finally, the last symbol $W_{1,U+1}$ can be decoded by removing the interference of the MDS-encoded pieces from $A_{N,1}^{[\boldsymbol{\theta}]}$. As a result, user 1 correctly recovers W_1.

In fact, for any user k, it is either in $[K_\mathrm{u}]\backslash\mathcal{D}_1$ or $[K_\mathrm{u}]\backslash\mathcal{D}_2$. This means one of $\widetilde{\boldsymbol{W}}_{1,i_1}$ and $\widetilde{\boldsymbol{W}}_{2,i_2}$ is stored (but not requested) by user k while the other is not stored but requested by user k. Therefore, user k can obtain $N-1$ linearly independent equations of the desired MDS-encoded piece $\widetilde{\boldsymbol{W}}_{\theta_k,i_{\theta_k}}$ according to (18) from which $\widetilde{\boldsymbol{W}}_{\theta_k,i_{\theta_k}}$ (i.e., $W_{\theta_k,[K_\mathrm{u}]\backslash\mathcal{D}_{\theta_k}}$) can be decoded. Together with the $2^{K_\mathrm{u}-1}$ stored subfiles, user k can recover the first U uncoded pieces $\{\boldsymbol{W}_{\theta_k,i}, i \in [U]\}$ of W_{θ_k}. User k can then construct all the MDS-encoded pieces $\{\widetilde{\boldsymbol{W}}_{\theta_k,i}, i \in [2^{K_\mathrm{u}}]\}$ by repeating (13). Finally, the last piece $W_{\theta_k,U+1}$ can be decoded by removing the interference of the MDS-encoded pieces from $A_{N,\theta_k}^{[\boldsymbol{\theta}]}$. As a result, we have proved that all users can correctly recover their desired files.

Server privacy. Server privacy is rather straightforward. Since each server only sees a subset of the coefficient vectors corresponding to each MDS-encoded piece, it is not sure if the coefficients are aligned or not across the servers. Because different choices of the coefficients correspond to different user demands, each server is unable to determine what $\boldsymbol{\theta}$ is. Therefore, server privacy is satisfied.

User privacy. Let us focus on user 1. User privacy means that, given any $\theta_1 \in \{1,2\}$, user 1 cannot determine what $(\theta_2, \cdots, \theta_{K_\mathrm{u}})$ is. Suppose $\theta_1 = 1$. Note that the query coefficients are fully observable by the users, so user 1 knows if the coefficient vectors corresponding to each MDS-encoded file piece are aligned across the servers or not. In particular, there is always one MDS-encoded piece of each file ($\widetilde{\boldsymbol{W}}_{1,i_1}$ of file W_1 and $\widetilde{\boldsymbol{W}}_{2,i_2}$ of file W_2) whose coefficient vectors are not aligned according to (17). Since $1 \notin [K_\mathrm{u}]\backslash\mathcal{D}_1, 1 \in [K_\mathrm{u}]\backslash\mathcal{D}_2$, we know that $\widetilde{\boldsymbol{W}}_{2,i_2}$ (i.e., $W_{2,[K_\mathrm{u}]\backslash\mathcal{D}_2}$) is stored by user 1 while $\widetilde{\boldsymbol{W}}_{1,i_1}$ (i.e., $W_{1,[K_\mathrm{u}]\backslash\mathcal{D}_1}$) is not stored by user 1 for whatever $(\theta_2, \cdots, \theta_{K_\mathrm{u}}) \in \{1,2\}^{1\times(K_\mathrm{u}-1)}$. Note that $\mathcal{D}_1, \mathcal{D}_2$ and i_1, i_2 are different for different $(\theta_2, \cdots, \theta_{K_\mathrm{u}})$. Intuitively, if user 1 cannot distinguish among the pieces $\{\widetilde{\boldsymbol{W}}_{1,i_1}, \forall(\theta_2, \cdots, \theta_{K_\mathrm{u}})\}$ and among the pieces $\{\widetilde{\boldsymbol{W}}_{2,i_2}, \forall(\theta_2, \cdots, \theta_{K_\mathrm{u}})\}$, it will not be able to tell what $(\theta_2, \cdots, \theta_{K_\mathrm{u}})$ is. This is actually achieved by the secret random mapping of the MDS-encoded pieces to the subfiles in (14). More specifically, the random mapping ensures that each user cannot distinguish among the set of the stored MDS-encoded pieces and among the set of the

unstored MDS-encoded pieces of each file. As a result, user 1 cannot distinguish among $\{(\widetilde{\boldsymbol{W}}_{1,i_1}, \widetilde{\boldsymbol{W}}_{2,i_2}), \forall(\theta_2, \cdots, \theta_{K_\mathrm{u}})\}$. Therefore, user 1 cannot determine the demands of other users. For $\theta_1 = 2$, it can be proved that the delivery scheme is private from user 1's view in a similar way. Due to symmetry, the above scheme is also private from any other user's perspective.

V. Conclusion

In this work, we introduced cache-aided multiuser PIR with user privacy (MuPIR-U), a new problem formulation that incorporates user-to-user privacy into conventional cache-aided multiuser PIR systems. A novel scheme was proposed for the case of two files with an arbitrary number of users and servers. The proposed scheme was shown to achieve optimal load in the large memory regime. Finding the optimal memory-load trade-off of MuPIR-U for an arbitrary number of files, users and servers can be challenging, which is left as future work.

Acknowledgment

The work of X. Zhang and M. Ji was supported through NSF CAREER Award 2145835 and NSF grant 2312227. The work of K. Wan was partially funded by the National Natural Science Foundation of China (NSFC-12141107). The work of H. Sun was supported in part by NSF under Grant CCF-2007108, Grant CCF-2045656 and Grant CCF-2312228. The work of G. Caire was partially funded by the European Research Council under the ERC Advanced Grant N. 789190, CARENET.

References

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

[2] 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.

[3] 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, 2018.

[4] 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.

[5] M. A. Attia, D. Kumar, and R. Tandon, "The capacity of private information retrieval from uncoded storage constrained databases," *arXiv preprint arXiv:1805.04104*, 2018.

[6] N. Woolsey, R.-R. Chen, and M. Ji, "An optimal iterative placement algorithm for pir from heterogeneous storage-constrained databases," *arXiv preprint arXiv:1904.02131*, 2019.

[7] H. Sun and S. A. Jafar, "The capacity of symmetric private information retrieval," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 322–329, Jan 2019.

[8] K. Banawan and S. Ulukus, "Multi-message private information retrieval: Capacity results and near-optimal schemes," *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6842–6862, Oct 2018.

[9] R. Tandon, "The capacity of cache aided private information retrieval," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 1078–1082.

[10] Y.-P. Wei, K. Banawan, and S. Ulukus, "Fundamental limits of cache-aided private information retrieval with unknown and uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 65, no. 5, pp. 3215–3232, 2018.

[11] S. Kadhe, B. Garcia, A. Heidarzadeh, S. El Rouayheb, and A. Sprintson, "Private information retrieval with side information," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2032–2043, 2020.

[12] Y. Wei, K. Banawan, and S. Ulukus, "Private information retrieval with partially known private side information," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, March 2018, pp. 1–6.

[13] A. Heidarzadeh, S. Kadhe, S. El Rouayheb, and A. Sprintson, "Single-server multi-message individually-private information retrieval with side information," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1042–1046.

[14] A. Heidarzadeh, B. Garcia, S. Kadhe, S. E. Rouayheb, and A. Sprintson, "On the capacity of single-server multi-message private information retrieval with side information," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2018, pp. 180–187.

[15] A. Heidarzadeh, F. Kazemi, and A. Sprintson, "Capacity of single-server single-message private information retrieval with private coded side information," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1662–1666.

[16] F. Kazemi, E. Karimi, A. Heidarzadeh, and A. Sprintson, "Single-server single-message online private information retrieval with side information," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 350–354.

[17] A. Heidarzadeh, F. Kazemi, and A. Sprintson, "The role of coded side information in single-server private information retrieval," *IEEE Transactions on Information Theory*, vol. 67, no. 1, pp. 25–44, 2021.

[18] X. Zhang, K. Wan, H. Sun, M. Ji, and G. Caire, "On the fundamental limits of cache-aided multiuser private information retrieval," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 5828–5842, 2021.

[19] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *Information Theory, IEEE Transactions on*, vol. 60, no. 5, pp. 2856–2867, 2014.

[20] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *Networking, IEEE/ACM Transactions on*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

[21] K. Wan and G. Caire, "On coded caching with private demands," *IEEE Transactions on Information Theory*, vol. 67, no. 1, pp. 358–372, 2020.

[22] C. Gurjarpadhye, J. Ravi, S. Kamath, B. K. Dey, and N. Karamchandani, "Fundamental limits of demand-private coded caching," *IEEE Transactions on Information Theory*, 2022.

[23] V. R. Aravind, P. Sarvepalli, and A. Thangaraj, "Subpacketization in coded caching with demand privacy," *arXiv preprint arXiv:1909.10471*, 2019.