2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)
Allerton Park and Retreat Center
Monticello, IL, USA, October 2-5, 2018

Capacity of Private Linear Computation for Coded Databases

Sarah A. Obead[†], Hsuan-Yin Lin[‡], Eirik Rosnes[‡], and Jörg Kliewer[†]
[†]Helen and John C. Hartmann Department of Electrical and Computer Engineering
New Jersey Institute of Technology, Newark, New Jersey 07102, USA
[‡]Simula UiB, N–5020 Bergen, Norway

***Abstract*—We consider the problem of private linear computation (PLC) in a distributed storage system. In PLC, a user wishes to compute a linear combination of f messages stored in noncolluding databases while revealing no information about the coefficients of the desired linear combination to the databases. In extension of our previous work we employ linear codes to encode the information on the databases. We show that the PLC capacity, which is the ratio of the desired linear function size and the total amount of downloaded information, matches the maximum distance separable (MDS) coded capacity of private information retrieval for a large class of linear codes that includes MDS codes. In particular, the proposed converse is valid for any number of messages and linear combinations, and the capacity expression depends on the rank of the coefficient matrix obtained from all linear combinations.**

I. Introduction

Private information retrieval (PIR) from public databases has been the focus of attention for several decades in the computer science community (see, e.g., [1], [2]). Recently, the aspect of minimizing the communication cost, e.g., the required rate or bandwidth of privately querying the databases with the desired requests and downloading the corresponding information from the databases has in particular been of interest also in the information theory and coding communities (see, e.g., [3]–[10]). Those settings primarily focus on distributed storage systems (DSSs), where data is encoded by a linear code and then stored across several databases.

A recently proposed generalization of the PIR problem [11]–[15] addresses the *private computation (PC)* of functions of these messages. In PC a user has access to a given number of databases and intends to compute a function of messages stored in these databases. This function is kept private from the databases, as they may be under the control of an adversary. In [11], [12], the authors consider that the data is replicated at each database, which are assumed to be noncolluding. Both works characterize the fundamental information-theoretic communication overhead needed to privately compute a given *linear* function, called private linear computation (PLC), and specify the corresponding capacity and achievable rates as a function of the number of messages and the number of databases, respectively. Interestingly, the obtained PLC capacity is equal to the PIR capacity of [6]. The extension to the coded case is addressed in [13], [14]. In [13], a PC scheme for polynomial functions of the data over t colluding databases is constructed by generalizing the star-product PIR scheme of [9]. Specifically, for systematically coded storage, this scheme considers functions that are polynomials of fixed degree g, and achieves a rate matching the best asymptotic rate (when the number of messages goes to infinity) of coded PIR $\mathsf{R} = 1 - \gamma_{\mathsf{c}}$ for $g = t = 1$ (the case of linear function retrieval and noncolluding databases), where γ_{c} denotes the code rate. In our previous work [14], we consider maximum distance separable (MDS) coded storage and the computation of linear message functions. The presented scheme is able to achieve the MDS-coded PIR capacity in [8], referred to as the MDS-PIR capacity in the sequel, and strictly generalizes the achievable schemes in [11], [12].

On the other hand, for the case of noncolluding databases, a PIR protocol for a DSS where data is stored using a non-MDS linear code, is proposed in [10]. The protocol is shown to achieve the asymptotic MDS-PIR capacity for several example codes. In [16], [17], this approach is extended to colluding databases and it is proved that both the asymptotic and the nonasymptotic MDS-PIR capacity can be achieved for a large class of linear codes in the noncolluding case. For noncolluding databases, the first family of non-MDS codes for which the PIR capacity is known is found in [18], [19]. Further, PIR on linearly-coded databases for the case of colluding databases is also addressed in [5], [9], [20]. For the PLC case however, capacity results for arbitrary linear codes have not been addressed so far in the open literature to the best of our knowledge.

In this work, we intend to fill this void and propose an extension of our previous works in [14] and [17] to PLC for arbitrary linear codes with noncolluding databases. Surprisingly, we show that the PLC capacity for a large class of linear codes matches the MDS-PIR capacity when a user wishes to compute an arbitrary linear combination of f independent equal-sized messages over some finite field $\mathbb{F}_q$, distributed over n noncolluding linearly-coded databases. In particular, by extending our previous work [14] where only achievable rates are stated, we also adapt the converse proof

This work is supported by US NSF grant CNS-1526547 and the Research Council of Norway (grant 240985/F20).

of [19, Thm. 4] to the PLC problem in this paper. In contrast to [12], our converse is valid for any number of messages f and any number of desired linear combinations μ, and our capacity result depends on the rank of the coefficient matrix obtained from all μ linear combinations.

II. Definitions and Problem Statement

A. Notation and Definitions

We denote by $\mathbb{N}$ the set of all positive integers and for some $a, b \in \mathbb{N}$, $a \leq b$, $[a : b] \triangleq \{a, a+1, \ldots, b\}$. Random and deterministic quantities are carefully distinguished as follows. A random variable is denoted by a capital Roman letter, e.g., X, while its realization is denoted by the corresponding small Roman letter, e.g., x; vectors are boldfaced, e.g., $\boldsymbol{X}$ denotes a random vector and $\boldsymbol{x}$ denotes a deterministic vector; random matrices are represented by bold sans serif letters, e.g., $\mathsf{\mathbf{X}}$, and X represents its realization. In addition, sets are denoted by calligraphic upper case letters, e.g., $\mathcal{X}$, and $\mathcal{X}^{\mathrm{c}}$ denotes the complement of a set $\mathcal{X}$ in a universe set. For a given index set $\mathcal{S}$, we also write $\mathsf{\mathbf{X}}^{\mathcal{S}}$ and $Y_{\mathcal{S}}$ to represent $\{\mathsf{\mathbf{X}}^{(v)} : v \in \mathcal{S}\}$ and $\{Y_j : j \in \mathcal{S}\}$, respectively. Furthermore, some constants are also depicted by Greek letters or a special font, e.g., X. We denote a submatrix of X that is restricted in columns by the set $\mathcal{I}$ by $\mathsf{X}|_{\mathcal{I}}$. The function $\mathrm{H}(\cdot)$ represents the entropy of its argument and $\mathrm{I}(X\,; Y)$ denotes the mutual information of the random variables X, Y. $(\cdot)^{\mathsf{T}}$ denotes the transpose of its argument, while $\mathrm{rank}(\mathsf{V})$ denotes the rank of a matrix V. We use the customary code parameters $[n, k]$ to denote a code $\mathscr{C}$ over the finite field $\mathbb{F}_q$ of blocklength n and dimension k. A generator matrix of $\mathscr{C}$ is denoted by $\mathsf{G}^{\mathscr{C}}$. A set of coordinates of $\mathscr{C}$, $\mathcal{I} \subseteq [1 : n]$, of size k is said to be an *information set* if and only if $\mathsf{G}^{\mathscr{C}}|_{\mathcal{I}}$ is invertible. The function $\chi(\boldsymbol{x})$ denotes the support of a vector $\boldsymbol{x}$.

B. System Model

The PLC problem for coded DSSs is described as follows. We consider a DSS that stores f messages $\mathsf{\mathbf{W}}^{(1)}, \ldots, \mathsf{\mathbf{W}}^{(f)}$, where each message $\mathsf{\mathbf{W}}^{(m)} = \bigl(W_{i,j}^{(m)}\bigr)$, $m \in [1 : f]$, can be seen as a random matrix of size $\beta \times k$ over $\mathbb{F}_q$ with $\beta, k \in \mathbb{N}$. Each message is encoded using an $[n, k]$ code as follows. Let $\boldsymbol{W}_i^{(m)} = \bigl(W_{i,1}^{(m)}, \ldots, W_{i,k}^{(m)}\bigr)$, $i \in [1 : \beta]$, be a message vector corresponding to the i-th row of $\mathsf{\mathbf{W}}^{(m)}$. Each $\boldsymbol{W}_i^{(m)}$ is encoded by an $[n, k]$ code $\mathscr{C}$ over $\mathbb{F}_q$ into a length-n codeword $\boldsymbol{C}_i^{(m)} = \bigl(C_{i,1}^{(m)}, \ldots, C_{i,n}^{(m)}\bigr)$. The βf generated codewords $\boldsymbol{C}_i^{(m)}$ are then arranged in the array $\mathsf{\mathbf{C}} = \bigl((\mathsf{\mathbf{C}}^{(1)})^{\mathsf{T}} | \ldots | (\mathsf{\mathbf{C}}^{(f)})^{\mathsf{T}}\bigr)^{\mathsf{T}}$ of dimensions $\beta f \times n$, where $\mathsf{\mathbf{C}}^{(m)} = \bigl((\boldsymbol{C}_1^{(m)})^{\mathsf{T}} | \ldots | (\boldsymbol{C}_\beta^{(m)})^{\mathsf{T}}\bigr)^{\mathsf{T}}$. The code symbols $C_{1,j}^{(m)}, \ldots, C_{\beta,j}^{(m)}$, $m \in [1 : f]$, for all f messages are stored on the j-th database, $j \in [1 : n]$.

Let $\mathsf{L} \triangleq \beta \cdot k$. Then, each message $\mathsf{\mathbf{W}}^{(m)}$, $m \in [1 : f]$, can also be seen as a random vector variable $\mathsf{\mathbf{W}}^{(m)} = (W_1^{(m)}, \ldots, W_{\mathsf{L}}^{(m)})$ of L symbols that are chosen independently and uniformly at random from $\mathbb{F}_q$. Thus,

$$\mathrm{H}\Bigl(\mathsf{\mathbf{W}}^{(m)}\Bigr) = \mathsf{L}, \forall\, m \in [1 : f],$$

$$\mathrm{H}\Bigl(\mathsf{\mathbf{W}}^{(1)}, \ldots, \mathsf{\mathbf{W}}^{(f)}\Bigr) = f\mathsf{L} \quad \text{(in q-ary units)}.$$

C. Private Linear Computation for Coded DSSs

We consider the case where no set of databases can collude. A user wishes to privately compute exactly one linear function from the μ *candidate* linear functions $\mathsf{\mathbf{X}}^{(1)}, \ldots, \mathsf{\mathbf{X}}^{(\mu)}$ from the coded DSS, where $\mathsf{\mathbf{X}}^{(v)} = \bigl(X_1^{(v)}, \ldots, X_{\mathsf{L}}^{(v)}\bigr)$. The μ-tuple $\bigl(X_l^{(1)}, \ldots, X_l^{(\mu)}\bigr)^{\mathsf{T}}$, $\forall\, l \in [1 : \mathsf{L}]$, is mapped by a deterministic matrix V of size $\mu \times f$ over $\mathbb{F}_q$ by

$$\begin{pmatrix} X_l^{(1)} \\ \vdots \\ X_l^{(\mu)} \end{pmatrix} = \mathsf{V}_{\mu \times f} \begin{pmatrix} W_l^{(1)} \\ \vdots \\ W_l^{(f)} \end{pmatrix}. \tag{1}$$

Hence, the user privately generates an index $v \in [1 : \mu]$ and wishes to compute the v-th linear function while keeping the index v private from each database. Here, we also assume that the rank of V is equal to $\mathrm{rank}(\mathsf{V}) = r \leq \min\{\mu, f\}$ and the indices corresponding to a basis for the row space of V are denoted by the set $\mathcal{L} \triangleq \{\ell_1, \ldots, \ell_r\} \subseteq [1 : \mu]$.

In order to retrieve the linear function $\mathsf{\mathbf{X}}^{(v)}$ from the coded DSS, $v \in [1 : \mu]$, the user sends a query $Q_j^{(v)}$ over $\mathbb{F}_q$ to the j-th database for all $j \in [1 : n]$. Since the queries are generated by the user without any prior knowledge of the realizations of the candidate linear functions, the queries are independent of the linear functions. In other words, we have

$$\mathrm{I}\Bigl(\mathsf{\mathbf{X}}^{(1)}, \ldots, \mathsf{\mathbf{X}}^{(\mu)}; Q_1^{(v)}, \ldots, Q_n^{(v)}\Bigr) = 0, \forall\, v \in [1 : \mu].$$

In response to the received query, database j sends the answer $A_j^{(v)}$ back to the user. $A_j^{(v)}$ is a deterministic function of $Q_j^{(v)}$ and the data stored in the database. Thus, $\forall\, v \in [1 : \mu]$,

$$\mathrm{H}\Bigl(A_j^{(v)} \Bigm| Q_j^{(v)}, \mathsf{\mathbf{C}}^{[1:f]}\Bigr) = 0, \forall\, j \in [1 : n].$$

Next, we define a PLC protocol for coded DSSs as follows.

Definition 1: Consider a DSS with n noncolluding databases storing f messages using an $[n, k]$ code. The user wishes to retrieve the v-th linear function $\mathsf{\mathbf{X}}^{(v)}$, $v \in [1 : \mu]$, from the available information $Q_j^{(v)}$ and $A_j^{(v)}$, $j \in [1 : n]$. For a PLC protocol, the following conditions must be satisfied $\forall\, v \in [1 : \mu]$,

[Privacy]

$$\mathrm{I}\bigl(v\,; Q_j^{(v)}, A_j^{(v)}, \mathsf{\mathbf{X}}^{(1)}, \ldots, \mathsf{\mathbf{X}}^{(\mu)}\bigr) = 0, \forall\, j \in [1 : n],$$

[Recovery]

$$\mathrm{H}\bigl(\mathsf{\mathbf{X}}^{(v)} \bigm| A_1^{(v)}, \ldots, A_n^{(v)}, Q_1^{(v)}, \ldots, Q_n^{(v)}\bigr) = 0.$$

To measure the efficiency of a PLC protocol, we consider the required number of downloaded symbols for retrieving the L symbols of the candidate linear function.

Definition 2 (PLC Rate and Capacity for Coded DSSs): The rate of a PLC protocol, denoted by R, is the ratio of the desired linear function size L to the total required download cost D, i.e.,

$$\mathsf{R} \triangleq \frac{\mathsf{L}}{\mathsf{D}}.$$

The PLC *capacity* $\mathsf{C}_{\mathrm{PLC}}$ is the maximum possible PLC rate over all possible PLC protocols for a given $[n, k]$ storage code.

D. MDS-PIR Capacity-Achieving Codes

In [16], [17], a PIR protocol for any linearly-coded DSS that uses an $[n, k]$ code to store f messages, named Protocol 1, is proposed. The corresponding PIR rate depends on the following property of the underlying storage code $\mathscr{C}$.

Definition 3: Let $\mathscr{C}$ be an arbitrary $[n, k]$ code. A $\nu \times n$ binary matrix $\mathsf{\Lambda}_{\kappa,\nu}(\mathscr{C})$ is said to be a *PIR achievable rate matrix* for $\mathscr{C}$ if the following conditions are satisfied.

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

In other words, each coordinate j of $\mathscr{C}$, $j \in [1 : n]$, appears exactly κ times in $\{\chi(\boldsymbol{\lambda}_i)\}_{i\in[1:\nu]}$, and every set $\chi(\boldsymbol{\lambda}_i)$ contains an information set.

In [16], [17], it is shown that the MDS-PIR capacity (i.e., the PIR capacity for a DSS where data is encoded and stored using an MDS code) can be achieved using Protocol 1 for a special class of $[n, k]$ codes. In particular, to achieve the MDS-PIR capacity using Protocol 1, the $[n, k]$ storage code should possess a specific underlying structure as given by the following theorem.

Theorem 1 ([16], [17]): Consider a DSS that uses an $[n, k]$ code $\mathscr{C}$ to store f messages. If a PIR achievable rate matrix $\mathsf{\Lambda}_{\kappa,\nu}(\mathscr{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ exists, then the MDS-PIR capacity

$$\mathsf{C}_{\text{MDS-PIR}} \triangleq \left(1 - \frac{k}{n}\right)\left[1 - \left(\frac{k}{n}\right)^{f}\right]^{-1}$$

is achievable.

This gives rise to the following definition.

Definition 4: A PIR achievable rate matrix $\mathsf{\Lambda}_{\kappa,\nu}(\mathscr{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ for an $[n, k]$ code $\mathscr{C}$ is called an *MDS-PIR capacity-achieving* matrix, and $\mathscr{C}$ is referred to as an *MDS-PIR capacity-achieving* code.

In Section IV, we will present a PLC protocol and a general achievable rate for any $\operatorname{rank}(\mathsf{V}) = r$ by using the PIR achievable rate matrix $\mathsf{\Lambda}_{\kappa,\nu}$ of an $[n, k]$ code.

III. PLC Capacity for MDS-PIR Capacity-Achieving Codes

The main result of this paper is the derivation of the PLC capacity for a coded DSS where data is encoded and stored using a linear code from the class of MDS-PIR capacity-achieving codes introduced in [16], [17], which is stated in the following theorem.

Theorem 2: Consider a DSS that uses an $[n, k]$ MDS-PIR capacity-achieving code $\mathscr{C}$ to store f messages. Then, the maximum achievable PLC rate over all possible PLC protocols, i.e., the PLC capacity, is

$$\mathsf{C}_{\text{PLC}} \triangleq \left(1 - \frac{k}{n}\right)\left[1 - \left(\frac{k}{n}\right)^{r}\right]^{-1} \tag{2}$$

for any rank r of the linear mapping from (1).

The achievable scheme is provided in Section IV, while the converse proof is given in Section V. Note that the class of MDS-PIR capacity-achieving codes includes MDS codes, cyclic codes, Reed-Muller codes, and certain classes of distance-optimal local reconstruction codes [16], [17].

We remark here that it is known that if $\operatorname{rank}(\mathsf{V}) = f$, then the PLC capacity for an MDS-coded DSS is equal to the MDS-PIR capacity [14], i.e., if $\operatorname{rank}(\mathsf{V}) = f$, then

$$\mathsf{C}_{\text{PLC}} = \left(1 - \frac{k}{n}\right)\left[1 - \left(\frac{k}{n}\right)^{f}\right]^{-1} = \mathsf{C}_{\text{MDS-PIR}}.$$

IV. Achievable Scheme

We start with a coded PIR query scheme for a message of μ *dependent* virtual messages. Given that the messages are stored using an $[n, k]$ MDS-PIR capacity-achieving code $\mathscr{C}$, we find the associated $\nu \times n$ MDS-PIR capacity-achieving matrix $\mathsf{\Lambda}_{\kappa,\nu}$ and obtain the PIR interference matrices $\mathsf{A}_{\kappa\times n}$ and $\mathsf{B}_{(\nu-\kappa)\times n}$ as given by the following definition.

Definition 5 ([16], [17]): For a given $\nu\times n$ PIR achievable rate matrix $\mathsf{\Lambda}_{\kappa,\nu}(\mathscr{C}) = (\lambda_{u,j})$, we define the PIR interference matrices $\mathsf{A}_{\kappa\times n} = (a_{i,j})$ and $\mathsf{B}_{(\nu-\kappa)\times n} = (b_{i,j})$ for the code $\mathscr{C}$ with

$$a_{i,j} \triangleq u \text{ if } \lambda_{u,j} = 1, \forall j \in [1 : n], i \in [1 : \kappa], u \in [1 : \nu],$$
$$b_{i,j} \triangleq u \text{ if } \lambda_{u,j} = 0, \forall j \in [1 : n], i \in [1 : \nu - \kappa], u \in [1 : \nu].$$

Note that in Definition 5, for each $j \in [1 : n]$, distinct values of $u \in [1 : \nu]$ should be assigned for all i. Thus, the assignment is not unique in the sense that the order of the entries of each column of A and B can be permuted. Next, for the sake of illustrating our query set generation algorithm, we make use of the following definition.

Definition 6: By $\mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ we denote the set of column coordinates of matrix $\mathsf{A}_{\kappa\times n} = (a_{i,j})$ in which at least one of its entries is equal to u, i.e.,

$$\mathcal{S}(u|\mathsf{A}_{\kappa\times n}) \triangleq \{j \in [1 : n]\colon \exists\, a_{i,j} = u, i \in [1 : \kappa]\}.$$

As a result, we require the size of the message to be $\mathsf{L} = \nu^{\mu} \cdot k$ (i.e., $\beta = \nu^{\mu}$). The scheme is completed in μ rounds.

A. Query Generation

Before running the main algorithm to generate query sets, the following index preparation for the coded symbols stored in each database is performed.

***1) Index Preparation*:** The goal is to make the coded symbols queried from each database to appear to be chosen randomly and independently from the desired linear function index. Note that the linear function is computed separately for the t-th row of all messages, $t \in [1 : \beta]$. Therefore, similar to the PLC scheme in [12] and the MDS-coded PLC scheme in [14], we apply a permutation that is fixed across all coded symbols for the t-th row to maintain the dependency across the associated message elements. Let $\pi(\cdot)$ be a random permutation function over $[1 : \beta]$, and let

$$U^{(v)}_{t,j} \triangleq \boldsymbol{v}_v \boldsymbol{C}_{\pi(t),j}, \quad t \in [1 : \beta], \quad j \in [1 : n],$$

denote the t-th permuted code symbol associated with the v-th virtual message $\mathbf{X}^{(v)}$ stored in the j-th database. Here, $\boldsymbol{v}_v$ represents the v-th row vector of the matrix $\mathsf{V}_{\mu\times f} = (v_{i,j})$ and $\boldsymbol{C}_{t,j} \triangleq \left(C^{(1)}_{t,j}, \ldots, C^{(f)}_{t,j}\right)^{\mathsf{T}}$. The permutation $\pi(\cdot)$ is randomly selected privately and uniformly by the user.

***2) Preliminaries*:** For $\tau \in [1:\mu]$, a sum $U^{(v_1)}_{i_1,j} + U^{(v_2)}_{i_2,j} + \cdots + U^{(v_\tau)}_{i_\tau,j}$, $j \in [1:n]$, of τ distinct symbols is a τ-sum for any subset $\{i_1, i_2, \dots, i_\tau\} \subseteq [1:\beta]$, and $\{v_1, v_2, \dots, v_\tau\} \subseteq [1:\mu]$ determines the *type* of the τ-sum. Since we have $\binom{\mu}{\tau}$ different selections of τ distinct elements out of μ elements, a τ-sum can have $\binom{\mu}{\tau}$ different types. The query generation procedure is subdivided into μ rounds. For a requested linear function indexed by $v \in [1:\mu]$, a query set $Q^{(v)}_j$, $j \in [1:n]$, is composed of μ disjoint subsets, one generated by each round $\tau \in [1:\mu]$. For each round the query subset is further subdivided into two subsets. The first subset $Q^{(v)}_j(\mathcal{D};\tau)$ consists of τ-sums with a single symbol from the *desired* linear function and $\tau-1$ symbols from *undesired* linear functions, while the second subset $Q^{(v)}_j(\mathcal{U};\tau)$ contains τ-sums with symbols only from undesired linear functions. To this end, in the scheme we introduce κn auxiliary query sets $Q^{(v)}_j(a_{i,j},\mathcal{D};\tau)$, $i \in [1:\kappa]$, where each query consists of a distinct symbol from the desired linear function and $\tau-1$ symbols from undesired linear functions, and $(\nu-\kappa)n$ auxiliary query sets $Q^{(v)}_j(b_{i,j},\mathcal{U};\tau)$, $i \in [1:\nu-\kappa]$, to represent the side information query sets for each database $j \in [1:n]$. We utilize these sets to generate the query sets of each round according to the PIR interference matrices $\mathsf{A}_{\kappa\times n}$ and $\mathsf{B}_{(\nu-\kappa)\times n}$. The query sets for all databases are generated by Algorithm 1 through the following procedures.[1]

***3) Initialization (Round $\tau = 1$)*:** In the initialization step, the algorithm generates the auxiliary queries for the first round. This round is described by Steps 5 to 11 of Algorithm 1, where we have $\tau = 1$ for the τ-sum. At this point, Algorithm 1 invokes the subroutine `Initial-Round` given in Algorithm 2 to generate $Q^{(v)}_j(a_{i,j},\mathcal{D};1)$, $i \in [1:\kappa]$, such that each of these query sets contains $\alpha_1 = \kappa^{\mu-1}$ distinct code symbols. Furthermore, to maintain function symmetry, the algorithm asks each database for the same number of distinct symbols of all other linear functions in $Q^{(v)}_j(a_{i,j},\mathcal{U};1)$, $i \in [1:\kappa]$, resulting in a total number of $\binom{\mu-1}{1}\kappa^{\mu-1}$ symbols. As a result, the queried code symbols in the auxiliary query sets for each database are symmetric with respect to all linear function vectors indexed by $v' \in [1:\mu]$. We associate the symbols of undesired functions in κ groups, each placed in the undesired query sets $Q^{(v)}_j(a_{i,j},\mathcal{U};1)$, $i \in [1:\kappa]$, to be exploited as distinct side information for the second round of the scheme. Since this procedure produces κ undesired query sets for each database, database symmetry is maintained.

***4) Desired Function Symbols for Rounds $\tau > 1$*:** For the following rounds a similar process is repeated in terms of generating auxiliary query sets containing distinct code symbols from the desired linear function $\mathbf{U}^{(v)} = (U^{(v)}_{t,j})$. This is accomplished in Steps 16 to 18 by calling the subroutine `Desired-Q`, given in Algorithm 3, to generate

[1]Note that a query $Q^{(v)}_j$ sent to the j-th database usually indicates the row indices of the stored code symbols that the user requests, while the answer $A^{(v)}_j$ to the query $Q^{(v)}_j$ refers to the particular code symbols requested through the query. In Algorithm 1, with some abuse of notation, the generated queries are sets containing their answers.

Algorithm 1: `Q-Gen`

```
   Input  : $v$, $\mu$, $\kappa$, $\nu$, $n$, $\mathsf{A}_{\kappa\times n}$, and $\mathsf{B}_{(\nu-\kappa)\times n}$
   Output: $Q^{(v)}_1, \dots, Q^{(v)}_n$
1  for $\tau \in [1:\mu]$ do
2      $Q^{(v)}_j(\mathcal{D};\tau) \leftarrow \emptyset$, $Q^{(v)}_j(\mathcal{U};\tau) \leftarrow \emptyset$, $j \in [1:n]$
3      $\alpha_\tau \leftarrow \kappa^{\mu-1} + \sum_{h=1}^{\tau-1}\binom{\mu-1}{h}\kappa^{\mu-(h+1)}(\nu-\kappa)^h$
4      ▷ Generate query sets for the initial
         round
5      if $\tau = 1$ then
6          for $u \in [1:\nu]$ do
7              for $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ do
8                  $Q^{(v)}_j(u,\mathcal{D};\tau), Q^{(v)}_j(u,\mathcal{U};\tau) \leftarrow$
                     Initial-Round$(u,\alpha_\tau,j,v,\tau)$
9              end
10         end
11     end
12     ▷ Generate query sets for the following
         rounds $\tau > 1$
13     else
14         for $u \in [1:\nu]$ do
15             ▷ Generate desired symbols for the
                 following rounds $\tau > 1$
16             for $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ do
17                 $Q^{(v)}_j(u,\mathcal{D};\tau) \leftarrow$ Desired-Q$(u,\alpha_\tau,j,v,\tau)$
18             end
19             ▷ Generate side information for the
                 following rounds $\tau > 1$
20             for $j \in \mathcal{S}(u|\mathsf{B}_{(\nu-\kappa)\times n})$ do
21                 $Q^{(v)}_j(u,\mathcal{U};\tau-1) \leftarrow$ Exploit-SI$(u, Q^{(v)}_1(u,\mathcal{U},\tau-$
                   $1),\dots,Q^{(v)}_n(u,\mathcal{U},\tau-1),j,v,\tau)$
22             end
23         end
24         ▷ Generate the final desired query sets
             for the following rounds $\tau > 1$
25         for $j \in [1:n]$ do
26             $\tilde{Q}^{(v)}_j(\mathcal{U};\tau-1) \leftarrow \bigcup_{i\in[1:\nu-\kappa]} Q^{(v)}_j(b_{i,j},\mathcal{U};\tau-1)$
27             $\tilde{Q}^{(v)}_j(1,\mathcal{U};\tau-1),\dots,\tilde{Q}^{(v)}_j(\kappa,\mathcal{U};\tau-1) \leftarrow$
                 Partition$\big(\tilde{Q}^{(v)}_j(\mathcal{U};\tau-1)\big)$
28             for $i \in [1:\kappa]$ do
29                 $Q^{(v)}_j(a_{i,j},\mathcal{D};\tau) \leftarrow$
                     SetAddition$\big(Q^{(v)}_j(a_{i,j},\mathcal{D};\tau),\tilde{Q}^{(v)}_j(i,\mathcal{U};\tau-1)\big)$
30             end
31         end
32         ▷ Generate the query sets of undesired
             symbols by forcing message symmetry
             for the following rounds $\tau > 1$
33         for $u \in [1:\nu]$ do
34             for $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ do
35                 $Q^{(v)}_j(u,\mathcal{U};\tau) \leftarrow$ M-Sym$\big(Q^{(v)}_j(u,\mathcal{D};\tau),j,v,\tau\big)$
36             end
37         end
38     end
39     for $u \in [1:\nu]$ do
40         for $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ do
41             $Q^{(v)}_j(\mathcal{D};\tau) \leftarrow Q^{(v)}_j(\mathcal{D};\tau) \cup Q^{(v)}_j(u,\mathcal{D};\tau)$
42             $Q^{(v)}_j(\mathcal{U};\tau) \leftarrow Q^{(v)}_j(\mathcal{U};\tau) \cup Q^{(v)}_j(u,\mathcal{U};\tau)$
43         end
44     end
45 end
46 for $j \in [1:n]$ do
47     $Q^{(v)}_j \leftarrow \bigcup_{\tau\in[1:\mu]}\big(Q^{(v)}_j(\mathcal{D};\tau) \cup Q^{(v)}_j(\mathcal{U};\tau)\big)$
48 end
```

Algorithm 2: `Initial-Round`

```
Input  : $u$, $\alpha_\tau$, $j$, $v$, and $\tau$
Output: $\phi^{(v)}(u,\mathcal{D};\tau), \phi^{(v)}(u,\mathcal{U};\tau)$
1 $\phi^{(v)}(u,\mathcal{D};\tau) \leftarrow \emptyset$, $\phi^{(v)}(u,\mathcal{U};\tau) \leftarrow \emptyset$
2 for $l \in [1:\alpha_\tau]$ do
3     $\phi^{(v)}(u,\mathcal{D};\tau) \leftarrow \phi^{(v)}(u,\mathcal{D};\tau) \cup \{U^{(v)}_{(u-1)\cdot\alpha_\tau+l,j}\}$
4     $\phi^{(v)}(u,\mathcal{U};\tau) \leftarrow \phi^{(v)}(u,\mathcal{U};\tau) \cup \left(\bigcup_{v'=1}^{\mu}\{U^{(v')}_{(u-1)\cdot\alpha_\tau+l,j}\} \setminus \{U^{(v)}_{(u-1)\cdot\alpha_\tau+l,j}\}\right)$
5 end
```

$Q^{(v)}_j(a_{i,j},\mathcal{D};\tau)$, $i \in [1:\kappa]$, such that each of these query sets contains $(\alpha_\tau - 1) - \alpha_{\tau-1} + 1 = \binom{\mu-1}{\tau-1}\kappa^{\mu-(\tau-1+1)}(\nu-\kappa)^{\tau-1}$ distinct code symbols from the desired linear function $\mathbf{U}^{(v)}$.

Algorithm 3: `Desired-Q`

```
Input  : $u$, $\alpha_\tau$, $j$, $v$, and $\tau$
Output: $\phi^{(v)}(u,\mathcal{D};\tau)$
1 $\phi^{(v)}(u,\mathcal{D};\tau) \leftarrow \emptyset$
2 for $l \in [\alpha_{\tau-1}:\alpha_\tau - 1]$ do
3     $\phi^{(v)}(u,\mathcal{D};\tau) \leftarrow \phi^{(v)}(u,\mathcal{D};\tau) \cup \{U^{(v)}_{l\cdot\nu+u,j}\}$
4 end
```

5) Side Information Exploitation: In Steps 20 to 22, we generate the side information query sets $Q^{(v)}_j(b_{i',j},\mathcal{U};\tau-1)$, $i' \in [1:\nu-\kappa]$, from the auxiliary query sets $Q^{(v)}_1(a_{i,1},\mathcal{U};\tau-1),\ldots,Q^{(v)}_n(a_{i,n},\mathcal{U};\tau-1)$, $i \in [1:\kappa]$, of the previous round $\tau-1$, $\tau \in [2:\mu]$, by applying the subrountine `Exploit-SI`, given by Algorithm 4. This subroutine is extended from [12] based on our coded storage scenario. Note that in Algorithm 4 the function `Reproduce`$(j, Q^{(v)}_{j'}(u,\mathcal{U};\tau-1))$, $j' \in [1:n]\setminus\{j\}$, simply reproduces all the queries in the auxiliary query set $Q^{(v)}_{j'}(u,\mathcal{U};\tau-1)$ with a different coordinate j.

Algorithm 4: `Exploit-SI`

```
Input  : $u$, $Q^{(v)}_1(u,\mathcal{U};\tau-1),\ldots,Q^{(v)}_n(u,\mathcal{U};\tau-1)$, $j$, $v$,
         and $\tau$
Output: $\phi^{(v)}(u,\mathcal{U};\tau-1)$
1 $\phi^{(v)}(u,\mathcal{U};\tau-1) \leftarrow \emptyset$
2 for $i \in [1:\kappa]$ do
3     for $j' \in [1:n]\setminus\{j\}$ do
4         if $u = a_{i,j'}$ then
5             $\phi^{(v)}(u,\mathcal{U};\tau-1) \leftarrow$ Reproduce$(j, Q^{(v)}_{j'}(u,\mathcal{U};\tau-1))$
6             break
7         end
8     end
9 end
```

Next, we update the desired query sets $Q^{(v)}_j(a_{i,j},\mathcal{D};\tau)$ in Steps 25 to 31. First, the function `Partition`$\big(\tilde{Q}^{(v)}_j(\mathcal{U};\tau-1)\big)$ denotes a procedure that divides a set into κ disjoint equally-sized subsets. This is viable since based on the subroutine `Initial-Round` and the following subroutine `M-Sym`, one can show that $\left|\tilde{Q}^{(v)}_j(\mathcal{U};\tau-1)\right| = \binom{\mu-1}{\tau-1}\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{(\tau-1)-1}\cdot(\nu-\kappa)$ for each round $\tau \in [2:\mu]$, which is always divisible by κ. Secondly, we assign the new query set of desired symbols $Q^{(v)}_j(a_{i,j},\mathcal{D};\tau)$ for the current round by using an element-wise set addition `SetAddition`(Q_1,Q_2). The element-wise set addition is defined as $\{q_{i_l}+q_{i'_l} : q_{i_l}\in Q_1, q_{i'_l}\in Q_2, l\in[1:\rho]\}$ with $|Q_1| = |Q_2| = \rho$, where ρ is an appropriate integer. In Steps 33 to 37, the subroutine `M-Sym`, given in Algorithm 5, is invoked to generate the undesired query sets $Q^{(v)}_j(a_{i,j},\mathcal{U};\tau)$ by utilizing message symmetry. This subroutine selects symbols of undesired messages to generate τ-sums that enforce symmetry in the round queries. The procedure resembles the subroutine `M-Sym` proposed in [12]. In Algorithm 5, Π_τ denotes the set of all possible selections of τ distinct indices in $[1:\mu]$ and `Lexico`(Π_τ) denotes the corresponding set of ordered selections (the indices $(v_1,\ldots,v_\tau)$ of a selection of Π_τ are ordered in natural lexicographical order). Further, the notation $U^{(v_x)}_{*,j}$ implies that the row index of the code symbol can be arbitrary. This is the case since only the linear function indices $(v_1,\ldots,v_\tau)$ are necessary to determine i_z, $\forall\, z\in[1:\tau]$. As a result, symmetry over the linear functions is maintained.

Algorithm 5: `M-Sym`

```
Input  : $Q^{(v)}_j(u,\mathcal{D};\tau)$, $j$, $v$, and $\tau$
Output: $\phi^{(v)}(u,\mathcal{U};\tau)$
1 $\phi^{(v)}(u,\mathcal{U};\tau) \leftarrow \emptyset$
2 for $(v_1,\ldots,v_\tau) \in$ Lexico$(\Pi_\tau)$, $v \notin \{v_1,\ldots,v_\tau\}$ do
3     $\phi^{(v)}(u,\mathcal{U};\tau) \leftarrow \phi^{(v)}(u,\mathcal{U};\tau) \cup \{U^{(v_1)}_{i_1,j}+\ldots+U^{(v_\tau)}_{i_\tau,j}\}$
       such that $\forall\, z\in[1:\tau]$,
       $\exists\, U^{(v)}_{i_z,j} + \sum_{x\in[1:\tau], x\neq z} U^{(v_x)}_{*,j} \in Q^{(v)}_j(u,\mathcal{D};\tau)$
4 end
```

6) Query Set Assembly: Finally, in Steps 39 to 48, we assemble each query set from disjoint query subsets obtained in all τ rounds. It can be shown that $Q^{(v)}_j(\mathcal{D};\tau)\cup Q^{(v)}_j(\mathcal{U};\tau)$ contains $\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}$ τ-sums for every τ-sum type.

B. Sign Assignment and Redundancy Elimination

After Algorithm 1, the user will know which row indices of the stored code symbols he/she is going to request. To reduce the total number of downloaded symbols, the linear dependency among the candidate linear functions is exploited. We adopt a similar sign assignment process over $\sigma^{(v)}_t \in \{+1,-1\}$ to each symbol in the query sets: $\sigma^{(v)}_t U^{(v)}_{t,j}$, based on the desired linear function index v as introduced in [12, Sec. 4.2]. Here, the sign $\sigma^{(v)}_t$ is privately generated by the user with a uniform distribution over $\{-1,+1\}$. The intuition behind the sign assignment is to introduce a uniquely solvable equation system from the different τ-sum types given the side information available from all other databases. By obtaining such a system of equations in each round, the user can determine some of the queries offline to decode the desired linear function and/or interference, thus reducing the download rate. Based on this insight we can state the following lemma.

Lemma 1: For all $v \in [1:\mu]$, each database $j \in [1:n]$, and based on the side information available from the databases, there are $\binom{\mu-r}{\tau}$ redundant τ-sum types out of all possible types $\binom{\mu}{\tau}$ in each round $\tau \in [1:\mu-r]$ of the query sets.

Since repetition codes and MDS codes are MDS-PIR capacity-achieving codes, Lemma 1 generalizes [12, Lem. 1] and [14, Lem. 1]. We can extend the proof of [12, Lem. 1] to our scheme based on the insight that the redundancy resulting from the linear dependencies between messages is also present for MDS-PIR capacity-achieving codes. We now make the final modification to our PLC query sets. We first directly apply the sign assignment $\sigma_t^{(v)}$, then remove all τ-sums corresponding to redundant τ-sum types from every round $\tau \in [1:\mu]$ (see Lemma 1). Note that, in our case the redundancy is dependent on the rank of the functions matrix, $\mathrm{rank}(\mathsf{V}) = r \leq \min\{\mu, f\}$, thus generalizing the MDS-coded PLC case. Finally, we generate the queries $Q_{1:n}^{(v)}$.

C. Example

To illustrate the key idea of the scheme we use the following example.

Consider four messages $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$, and $\mathbf{W}^{(4)}$ that are stored in a DSS using a $[4,2]$ MDS-PIR capacity-achieving code $\mathscr{C}$ with generator matrix

$$\mathsf{G}^{\mathscr{C}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \text{ for which } \mathsf{\Lambda}_{1,2} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

is a PIR achievable rate matrix. According to Definition 5 we obtain the PIR interference matrices $\mathsf{A}_{1\times 4} = \begin{pmatrix} 1 & 2 & 1 & 2 \end{pmatrix}$ and $\mathsf{B}_{1\times 4} = \begin{pmatrix} 2 & 1 & 2 & 1 \end{pmatrix}$. Suppose that the user wishes to obtain a linear function $\mathbf{X}^{(v)}$ from a set of $\mu = 3$ candidate linear functions, i.e., $v \in [1:3]$, defined by

$$\mathsf{V}_{\mu\times f} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix}.$$

We simplify notation by letting $x_{t,j} = U_{t,j}^{(1)}$, $y_{t,j} = U_{t,j}^{(2)}$, and $z_{t,j} = U_{t,j}^{(3)}$ for all $t \in [1:\beta]$, $j \in [1:n]$, where $\beta = \nu^{\mu} = 8$. Let the desired linear function index be $v = 1$. For this example, the construction of the query sets is briefly presented in the following steps.

Initialization (Round $\tau = 1$): Algorithm 1 starts with $\tau = 1$ to generate auxiliary query sets for each database holding $\kappa^{\mu-1} = 1$ distinct instances of $x_{t,j}$. By message symmetry this also applies to $y_{t,j}$ and $z_{t,j}$. The auxiliary query sets for the first round are given in Table I(a). Note that the queries for $z_{t,j}$ can be generated offline by the user and thus are later removed from the query sets.

Following Rounds ($\tau > 1$): As can be seen from Table I(b) and (c), using the interference matrices $\mathsf{A}_{1\times 4}$ and $\mathsf{B}_{1\times 4}$, the algorithm further generates the auxiliary query sets $Q_j^{(1)}(a_{1,j}, \mathcal{D}; \tau)$ that contain new symbols to be queried jointly with symbols of the desired linear function during the exploitation of side information for the j-th database, $j \in [1:n]$. After generating the desired auxiliary query sets $Q_j^{(1)}(a_{1,j}, \mathcal{D}; \tau)$, the undesired auxiliary query sets $Q_j^{(1)}(a_{1,j}, \mathcal{U}; \tau)$ are generated by enforcing message symmetry. Finally, we apply the sign assignment scheme to the query sets and remove redundant queries from each database. The resulting queries are shown in Table II.

The PLC rate of the achievable scheme is equal to $\frac{\nu^{\mu}\cdot k}{\mathsf{D}} = \frac{8\cdot 2}{6\cdot 4} = \frac{2}{3}$, which is equal to the PLC capacity in (2) with $\mathrm{rank}(\mathsf{V}) = 2$.

TABLE I
AUXILIARY QUERY SETS FOR EACH ROUND. THE MAGENTA DASHED ARROWS AND THE CYAN ARROWS INDICATE THAT THE EXPLOIT-SI ALGORITHM AND THE M-SYM ALGORITHM ARE USED, RESPECTIVELY.

j	1	2	3	4
$Q_j^{(1)}(a_{1,j}, \mathcal{D}; 1)$	$x_{(1-1)\cdot 1+1,1}$	$x_{(2-1)\cdot 1+1,2}$	$x_{(1-1)\cdot 1+1,3}$	$x_{(2-1)\cdot 1+1,4}$
$Q_j^{(1)}(a_{1,j}, \mathcal{U}; 1)$	$y_{(1-1)\cdot 1+1,1}$ $z_{(1-1)\cdot 1+1,1}$	$y_{(2-1)\cdot 1+1,2}$ $z_{(2-1)\cdot 1+1,2}$	$y_{(1-1)\cdot 1+1,3}$ $z_{(1-1)\cdot 1+1,3}$	$y_{(2-1)\cdot 1+1,4}$ $z_{(2-1)\cdot 1+1,4}$

(a)

j	1	2	3	4
$Q_j^{(1)}(a_{1,j}, \mathcal{D}; 2)$	$x_{1\cdot 2+1,1} + y_{2,1}$ $x_{2\cdot 2+1,1} + z_{2,1}$	$x_{1\cdot 2+2,2} + y_{1,2}$ $x_{2\cdot 2+2,2} + z_{1,2}$	$x_{1\cdot 2+1,3} + y_{2,3}$ $x_{2\cdot 2+1,3} + z_{2,3}$	$x_{1\cdot 2+2,4} + y_{1,4}$ $x_{2\cdot 2+2,4} + z_{1,4}$
$Q_j^{(1)}(a_{1,j}, \mathcal{U}; 2)$	$y_{4+1,1} + z_{2+1,1}$	$y_{4+2,2} + z_{2+2,2}$	$y_{4+1,3} + z_{2+1,3}$	$y_{4+2,4} + z_{2+2,4}$

(b)

j	1	2	3	4
$Q_j^{(1)}(a_{1,j}, \mathcal{D}; 3)$	$x_{3\cdot 2+1,1} + y_{6,1} + z_{4,1}$	$x_{3\cdot 2+1,2} + y_{5,2} + z_{3,2}$	$x_{3\cdot 2+1,3} + y_{6,3} + z_{4,3}$	$x_{3\cdot 2+1,4} + y_{5,4} + z_{3,4}$

(c)

TABLE II
QUERY SETS FOR A CODED DSS THAT USES THE $[4,2]$ CODE OF EXAMPLE IV-C TO STORE $f = 4$ MESSAGES.

j	1	2	3	4
$Q_j^{(1)}(\mathcal{D}; 1)$	$x_{1,1}$	$x_{2,2}$	$x_{1,3}$	$x_{2,4}$
$Q_j^{(1)}(\mathcal{U}; 1)$	$y_{1,1}$	$y_{2,2}$	$y_{1,3}$	$y_{2,4}$
$Q_j^{(1)}(\mathcal{D}; 2)$	$x_{3,1} + y_{2,1}$ $x_{5,1} + z_{2,1}$	$x_{4,2} + y_{1,2}$ $x_{6,2} + z_{1,2}$	$x_{3,3} + y_{2,3}$ $x_{5,3} + z_{2,3}$	$x_{4,4} + y_{1,4}$ $x_{6,4} + z_{1,4}$
$Q_j^{(1)}(\mathcal{U}; 2)$	$y_{5,1} + z_{3,1}$	$y_{6,2} + z_{4,2}$	$y_{5,3} + z_{3,3}$	$y_{6,4} + z_{4,4}$
$Q_j^{(1)}(\mathcal{D}; 3)$	$x_{7,1} + y_{6,1} + z_{4,1}$	$x_{8,2} + y_{5,2} + z_{3,2}$	$x_{7,3} + y_{6,3} + z_{4,3}$	$x_{8,4} + y_{5,4} + z_{3,4}$

D. Achievable Rate

The resulting achievable rate of Algorithm 1 after removing redundant τ-sums according to Lemma 1 is given as

$$\begin{aligned}
\mathsf{R} &\overset{(a)}{=} \frac{k\nu^{\mu}}{n\sum_{\tau=1}^{\mu}\left(\binom{\mu}{\tau} - \binom{\mu-r}{\tau}\right)\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}} \\
&\overset{(b)}{=} \frac{\kappa\nu^{\mu}}{\nu\sum_{\tau=1}^{\mu}\left(\binom{\mu}{\tau} - \binom{\mu-r}{\tau}\right)\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}} \\
&= \frac{\nu^{\mu}\left(\frac{\nu-\kappa}{\nu}\right)}{\sum_{\tau=1}^{\mu}\left(\binom{\mu}{\tau} - \binom{\mu-r}{\tau}\right)\kappa^{\mu-\tau}(\nu-\kappa)^{\tau}} \\
&\;\;\vdots \\
&\overset{(c)}{=} \frac{\nu^{\mu}\left(1-\frac{\kappa}{\nu}\right)}{\nu^{\mu} - \kappa^{r}\nu^{\mu-r}} \\
&= \left(1 - \frac{\kappa}{\nu}\right)\left[1 - \left(\frac{\kappa}{\nu}\right)^{r}\right]^{-1},
\end{aligned}$$

where we define $\binom{m}{n} \triangleq 0$ if $m < n$; (a) follows from the PLC rate in Definition 2, the fact that $Q_j^{(v)}(\mathcal{D};\tau) \cup Q_j^{(v)}(\mathcal{U};\tau)$

contains $\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}$ τ-sums for every τ-sum type, and Lemma 1; (b) follows from Definition 4; and (c) follows by adapting similar steps as in the proof given in [14].

V. Converse Proof

In [18], [19], the PIR capacity of MDS-PIR capacity-achieving codes is shown to be equal to the MDS-PIR capacity. In this section, we adapt the converse proof of [19, Thm. 4] to the scenario of the linearly-coded PLC problem. We will show that the PLC capacity of MDS-PIR capacity-achieving codes is equal to (2). Before we proceed with the converse proof, we give some general results that are useful for the proof.

1) From the condition of privacy,
$$\mathsf{H}\bigl(A_j^{(v)}\,\big|\,\mathbf{X}^{(v)},\mathcal{Q}\bigr)=\mathsf{H}\bigl(A_j^{(v')}\,\big|\,\mathbf{X}^{(v)},\mathcal{Q}\bigr),\qquad(3)$$
where $v\neq v'$, $v,v'\in[1:\mu]$, and $\mathcal{Q}\triangleq\bigl\{Q_j^{(v)}\colon v\in[1:\mu],j\in[1:n]\bigr\}$ denotes the set of all possible queries made by the user. Although this seems to be intuitively true, a proof of this property is still required and can be found in [21, Lem. 3].

2) Consider a PLC protocol for a coded DSS that uses an $[n,k]$ code $\mathscr{C}$ to store f messages. For any subset of linear combinations $\mathcal{V}\subseteq[1:\mu]$ and for any information set $\mathcal{I}$ of $\mathscr{C}$, we have
$$\mathsf{H}\bigl(A_{\mathcal{I}}^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)=\sum_{j\in\mathcal{I}}\mathsf{H}\bigl(A_j^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr).\qquad(4)$$
The proof uses the linear independence of the columns of a generator matrix of $\mathscr{C}$ corresponding to an information set, and can be seen as a simple extension of [8, Lem. 2] or [21, Lem. 4]. This argument applies to the case of PLC due to the fact that $A_{\mathcal{I}}^{(v)}$ is still a deterministic function of independent random variables $\{\boldsymbol{C}_j\colon j\in\mathcal{I}\}$ and $\mathcal{Q}$, where $\boldsymbol{C}_j\triangleq\bigl(C_{1,j}^{(1)},\ldots,C_{\beta,j}^{(1)},C_{1,j}^{(2)},\ldots,C_{\beta,j}^{(f)}\bigr)^{\mathsf{T}}$ denotes the f coded chunks that are stored in the j-th database.

Next, we state Shearer's Lemma, which represents a very useful entropy method for combinatorial problems.

Lemma 2 (Shearer's Lemma [22]): Let $\mathscr{S}$ be a collection of subsets of $[1:n]$, with each $j\in[1:n]$ included in at least κ members of $\mathscr{S}$. For random variables $Z_1,\ldots,Z_n$, we have
$$\sum_{\mathcal{S}\in\mathscr{S}}\mathsf{H}(Z_{\mathcal{S}})\geq\kappa\,\mathsf{H}(Z_1,\ldots,Z_n).$$

For our converse proof for the coded PLC problem, we also need the following lemma, whose proof has been omitted due to lack of space.

Lemma 3: Consider the linear mapping $\mathsf{V}=(v_{i,j})$ defined in (1) with $\operatorname{rank}(\mathsf{V})=r$ where $v_{i_1,j_1},\ldots,v_{i_r,j_r}$ are the entries corresponding to the pivot elements of V. It follows that $\bigl(\mathbf{X}^{(i_1)},\ldots,\mathbf{X}^{(i_h)}\bigr)$ and $\bigl(\mathbf{W}^{(j_1)},\ldots,\mathbf{W}^{(j_h)}\bigr)$ are identically distributed, for some $h\in[1:r]$. In other words, $\mathsf{H}\bigl(\mathbf{X}^{(i_1)},\ldots,\mathbf{X}^{(i_h)}\bigr)=h\mathsf{L}$, $h\in[1:r]$.

Now, we are ready for the converse proof. By [17, Lem. 2], since the code $\mathscr{C}$ is MDS-PIR capacity-achieving, there exist ν information sets $\mathcal{I}_1,\ldots,\mathcal{I}_\nu$ such that each coordinate $j\in[1:n]$ is included in exactly κ members of $\mathscr{I}=\{\mathcal{I}_1,\ldots,\mathcal{I}_\nu\}$ with $\frac{\kappa}{\nu}=\frac{k}{n}$.

Applying the chain rule of entropy we have
$$\mathsf{H}\bigl(A_{[1:n]}^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\geq\mathsf{H}\bigl(A_{\mathcal{I}_i}^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr),\quad\forall\,i\in[1:\nu].$$

Let $v\in\mathcal{V}$ and $v'\in\mathcal{V}^{\mathsf{c}}\triangleq[1:\mu]\setminus\mathcal{V}$. Following similar steps as in the proof given in [8], [21], we get
$$\begin{aligned}
\nu\,\mathsf{H}\bigl(A_{[1:n]}^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)
&\geq\sum_{i=1}^{\nu}\mathsf{H}\bigl(A_{\mathcal{I}_i}^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\\
&=\sum_{i=1}^{\nu}\Biggl(\sum_{j\in\mathcal{I}_i}\mathsf{H}\bigl(A_j^{(v)}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\Biggr) &(5)\\
&=\sum_{i=1}^{\nu}\Biggl(\sum_{j\in\mathcal{I}_i}\mathsf{H}\bigl(A_j^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\Biggr) &(6)\\
&=\sum_{i=1}^{\nu}\mathsf{H}\bigl(A_{\mathcal{I}_i}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr) &(7)\\
&\geq\kappa\,\mathsf{H}\bigl(A_{[1:n]}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr) &(8)\\
&=\kappa\Bigl[\mathsf{H}\bigl(A_{[1:n]}^{(v')},\mathbf{X}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\\
&\qquad-\mathsf{H}\bigl(\mathbf{X}^{(v')}\,\big|\,A_{[1:n]}^{(v')},\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\Bigr]\\
&=\kappa\Bigl[\mathsf{H}\bigl(\mathbf{X}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\\
&\qquad+\mathsf{H}\bigl(A_{[1:n]}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathbf{X}^{(v')},\mathcal{Q}\bigr)-0\Bigr] &(9)\\
&=\kappa\Bigl[\mathsf{H}\bigl(\mathbf{X}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}}\bigr)+\mathsf{H}\bigl(A_{[1:n]}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathbf{X}^{(v')},\mathcal{Q}\bigr)\Bigr], &(10)
\end{aligned}$$
where (5) and (7) follow from (4); (6) is because of (3); (8) is due to Shearer's Lemma; (9) is from the fact that the v'-th linear combination $\mathbf{X}^{(v')}$ is determined by the answers $A_{[1:n]}^{(v')}$ and all possible queries $\mathcal{Q}$; and finally, (10) follows from the independence between all possible queries and the messages. Therefore, we can conclude that
$$\begin{aligned}
&\mathsf{H}\bigl(A_{[1:n]}^{(v)}\big|\,\mathbf{X}^{\mathcal{V}},\mathcal{Q}\bigr)\\
&\quad\geq\ \frac{\kappa}{\nu}\,\mathsf{H}\bigl(\mathbf{X}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}}\bigr)+\frac{\kappa}{\nu}\,\mathsf{H}\bigl(A_{[1:n]}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathbf{X}^{(v')},\mathcal{Q}\bigr)\\
&\quad=\ \frac{k}{n}\,\mathsf{H}\bigl(\mathbf{X}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}}\bigr)+\frac{k}{n}\,\mathsf{H}\bigl(A_{[1:n]}^{(v')}\,\big|\,\mathbf{X}^{\mathcal{V}},\mathbf{X}^{(v')},\mathcal{Q}\bigr),\qquad(11)
\end{aligned}$$
where we have used Definition 4 to obtain (11).

Since there are in total μ linear combinations and $\mathcal{L}\triangleq\{\ell_1,\ldots,\ell_r\}\subseteq[1:\mu]$ is the set of row indices corresponding to the selected basis for the row space of V, we can recursively use (11) $r-1$ times to obtain
$$\begin{aligned}
&\mathsf{H}\bigl(A_{[1:n]}^{(\ell_1)}\,\big|\,\mathbf{X}^{(\ell_1)},\mathcal{Q}\bigr)\\
&\quad\geq\sum_{v=1}^{r-1}\Bigl(\frac{k}{n}\Bigr)^{v}\,\mathsf{H}\bigl(\mathbf{X}^{(\ell_{v+1})}\,\big|\,\mathbf{X}^{\{\ell_1,\ldots,\ell_v\}}\bigr)\\
&\qquad\qquad+\Bigl(\frac{k}{n}\Bigr)^{r-1}\,\mathsf{H}\bigl(A_{[1:n]}^{(\ell_r)}\,\big|\,\mathbf{X}^{\{\ell_1,\ldots,\ell_r\}},\mathcal{Q}\bigr)
\end{aligned}$$

$$
\begin{aligned}
&\geq \sum_{v=1}^{r-1} \Bigl(\frac{k}{n}\Bigr)^{v} \mathsf{H}\bigl(\mathbf{X}^{(\ell_{v+1})} \bigm| \mathbf{X}^{\{\ell_1,\ldots,\ell_v\}}\bigr) && (12)\\
&= \sum_{v=1}^{r-1} \Bigl(\frac{k}{n}\Bigr)^{v} \mathsf{L}, && (13)
\end{aligned}
$$

where (12) follows from the nonnegativity of entropy. (13) holds since it follows from Lemma 3 that $\mathsf{H}\bigl(\mathbf{X}^{(\ell_{v+1})} \bigm| \mathbf{X}^{\{\ell_1,\ldots,\ell_v\}}\bigr) = \mathsf{H}\bigl(\mathbf{X}^{(\ell_{v+1})}\bigr) = \mathsf{L}$. Here, we also remark that the recursive steps follow the same principle of the general converse for dependent PIR (DPIR) from [15, Thm. 1]. In [15], the authors claim that the general converse for the DPIR problem strongly depends on the chosen permutation of the indices of the candidate functions. However, the best index permutation for the candidate linear functions for the PLC problem results from finding a basis for V. Now,

$$
\begin{aligned}
\mathsf{L} &= \mathsf{H}\Bigl(\mathbf{X}^{(\ell_1)}\Bigr)\\
&= \mathsf{H}\bigl(\mathbf{X}^{(\ell_1)} \bigm| \mathcal{Q}\bigr) - \underbrace{\mathsf{H}\bigl(\mathbf{X}^{(\ell_1)} \bigm| A^{(\ell_1)}_{[1:n]}, \mathcal{Q}\bigr)}_{=0} && (14)\\
&= \mathsf{I}\bigl(\mathbf{X}^{(\ell_1)}; A^{(\ell_1)}_{[1:n]} \bigm| \mathcal{Q}\bigr)\\
&= \mathsf{H}\Bigl(A^{(\ell_1)}_{[1:n]} \Bigm| \mathcal{Q}\Bigr) - \mathsf{H}\Bigl(A^{(\ell_1)}_{[1:n]} \Bigm| \mathbf{X}^{(\ell_1)}, \mathcal{Q}\Bigr)\\
&\leq \mathsf{H}\Bigl(A^{(\ell_1)}_{[1:n]} \Bigm| \mathcal{Q}\Bigr) - \sum_{v=1}^{r-1} \Bigl(\frac{k}{n}\Bigr)^{v} \mathsf{L}, && (15)
\end{aligned}
$$

where (14) follows since any message is independent of the queries $\mathcal{Q}$, and knowing the answers $A^{(\ell_1)}_{[1:n]}$ and the queries $\mathcal{Q}$, one can determine $\mathbf{X}^{(\ell_1)}$; (15) holds because of (13).

Finally, the converse proof is completed by showing that

$$
\begin{aligned}
\mathsf{R} &= \frac{\mathsf{L}}{\sum_{j=1}^{n} \mathsf{H}\Bigl(A^{(\ell_1)}_{j}\Bigr)}\\
&\leq \frac{\mathsf{L}}{\mathsf{H}\Bigl(A^{(\ell_1)}_{[1:n]}\Bigr)} && (16)\\
&\leq \frac{\mathsf{L}}{\mathsf{H}\Bigl(A^{(\ell_1)}_{[1:n]} \Bigm| \mathcal{Q}\Bigr)} && (17)\\
&\leq \frac{1}{1 + \sum_{v=1}^{r-1} \bigl(\frac{k}{n}\bigr)^{v}} = \mathsf{C}_{\mathrm{PLC}}, && (18)
\end{aligned}
$$

where (16) holds because of the chain rule of entropy, (17) is due to the fact that conditioning reduces entropy, and we apply (15) to obtain (18).

VI. Conclusion

We have provided the capacity of PLC for DSSs, where data is encoded and stored using an arbitrary linear code from the class of MDS-PIR capacity-achieving codes. Interestingly, the capacity is equal to the corresponding PIR capacity. Thus, privately retrieving arbitrary linear combinations of the stored messages does not incur any overhead in rate compared to retrieving a single message from the databases.

References

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

[2] S. Yekhanin, "Private information retrieval," *Commun. ACM*, vol. 53, no. 4, pp. 68–73, Apr. 2010.

[3] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, HI, USA, Jun. 29 – Jul. 4, 2014, pp. 856–860.

[4] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, China, Jun. 14–19, 2015, pp. 2842–2846.

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

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

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

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

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

[10] S. Kumar, E. Rosnes, and A. Graell i Amat, "Private information retrieval in distributed storage systems using an arbitrary linear code," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Germany, Jun. 25–30, 2017, pp. 1421–1425.

[11] M. Mirmohseni and M. A. Maddah-Ali, "Private function retrieval," Nov. 2017, arXiv:1711.04677v2 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1711.04677

[12] H. Sun and S. A. Jafar, "The capacity of private computation," Oct. 2017, arXiv:1710.11098v3 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1710.11098

[13] D. Karpuk, "Private computation of systematically encoded data with colluding servers," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, CO, USA, Jun. 17–22, 2018, pp. 2112–2116.

[14] S. A. Obead and J. Kliewer, "Achievable rate of private function retrieval from MDS coded databases," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, CO, USA, Jun. 17–22, 2018, pp. 2117–2121.

[15] Z. Chen, Z. Wang, and S. Jafar, "The asymptotic capacity of private search," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, CO, USA, Jun. 17–22, 2018, pp. 2122–2126.

[16] S. Kumar, H.-Y. Lin, E. Rosnes, and A. Graell i Amat, "Achieving maximum distance separable private information retrieval capacity with linear codes," Dec. 2017, arXiv:1712.03898v3 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1712.03898

[17] H.-Y. Lin, S. Kumar, E. Rosnes, and A. Graell i Amat, "An MDS-PIR capacity-achieving protocol for distributed storage using non-MDS linear codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, CO, USA, Jun. 17–22, 2018, pp. 966–970.

[18] ——, "Asymmetry helps: Improved private information retrieval protocols for distributed storage," to be pres. at *IEEE Inf. Theory Workshop*, Guangzhou, China, Nov. 25–29, 2018. [Online]. Available: https://arxiv.org/abs/1806.01342

[19] ——, "On the fundamental limit of private information retrieval for coded distributed storage," Aug. 2018, arXiv:1808.09018v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1808.09018

[20] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, A.-L. Horlemann-Trautmann, D. Karpuk, and I. Kubjas, "t-private information retrieval schemes using transitive codes," Dec. 2017, arXiv:1712.02850v1 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1712.02850

[21] J. Xu and Z. Zhang, "On sub-packetization of capacity-achieving PIR schemes for MDS coded databases," Dec. 2017, arXiv:1712.02466v2 [cs.IT]. [Online]. Available: http://arxiv.org/abs/1712.02466

[22] J. Radhakrishnan, "Entropy and counting," in *Computational mathematics, modelling and algorithms*, J. C. Misra, Ed. Narosa Publishing House, 2003, pp. 146–168.