Private Linear Computation for Noncolluding Coded Databases

Sarah A. Obead, *Graduate Student Member, IEEE*, Hsuan-Yin Lin, *Senior Member, IEEE*, Eirik Rosnes, *Senior Member, IEEE*, and Jörg Kliewer, *Senior Member, IEEE*

***Abstract*— Private computation in a distributed storage system (DSS) is a generalization of the private information retrieval (PIR) problem. In such a setting, a user wishes to compute a function of f messages stored in n noncolluding coded databases, i.e., databases storing data encoded with an $[n, k]$ linear storage code, while revealing no information about the desired function to the databases. We consider the problem of private linear computation (PLC) for coded databases. In PLC, a user wishes to compute a linear combination over the f messages while keeping the coefficients of the desired linear combination hidden from the databases. For a DSS setup where data is stored using a code from a particular family of linear storage codes, we derive an outer bound on the PLC rate, which is defined as the ratio of the desired amount of information and the total amount of downloaded information. In particular, the proposed converse is valid for any number of messages and linear combinations, and depends on the rank of the coefficient matrix obtained from all linear combinations. Further, we present a PLC scheme with rate equal to the outer bound and hence settle the PLC capacity for the considered class of linear storage codes. Interestingly, the PLC capacity matches the maximum distance separable coded capacity of PIR for the considered class of linear storage codes.**

***Index Terms*— Capacity, information-theoretic privacy, private computation, private information retrieval.**

I. Introduction

THE problem of private information retrieval (PIR) from public databases, introduced by Chor *et al.* [2], has been the focus of attention for several decades in the computer science community (see, e.g., [3]–[5]). The goal of PIR is to allow a user to privately access an arbitrary message stored in a set of databases, i.e., without revealing any information of the identity of the requested message to each database. If the users do not have any side information on the data stored in the databases, the best strategy is to store the messages in at least two databases while ensuring PIR. Hence, the design of PIR protocols has focused on the case when multiple databases store the messages. This connects to the active and renowned research area of distributed storage systems (DSSs), where the data is encoded by an $[n, k]$ linear code and then distributed and stored across n storage nodes [6], usually referred to as *coded DSSs*. Using coding techniques, coded DSSs possess many practical features and benefits such as high reliability, efficient repairability, robustness, and security [7]. 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 attracted a great deal of attention in the information theory and coding communities. Thus, the renewed interest in PIR primarily focused on the study and design of efficient PIR protocols for coded DSSs (see, e.g., [8]–[15]).

A recently proposed generalization of the PIR problem [16]–[20] addresses *private computation (PC)* for functions of the stored messages, also denoted as private function retrieval [21]. 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. The PC rate, defined as the ratio of the desired amount of information and the total amount of downloaded information is the main performance metric in this line of research. Accordingly, the PC capacity is defined as the maximum of all achievable PC rates over all possible PC protocols. In [16], [21], the capacity and achievable rates for the case of privately computing a given *linear* function, referred to as private linear computation (PLC), were derived as a function of the number of messages and the number of databases, respectively, for the scenario of noncolluding replicated databases. Interestingly, the obtained PLC capacity is equal to the PIR capacity of [10]. The extension to the coded case is addressed in [18]–[20]. In particular, in [18] we proposed a PLC scheme based on maximum distance separable (MDS) coded storage. The presented scheme is able to achieve the MDS-coded PIR capacity, i.e., the capacity of PIR over noncolluding MDS-coded DSSs, established in [12], referred to as the MDS-PIR capacity in the sequel. In [19], private polynomial computation (PPC) over t colluding and systematically coded databases was considered by generalizing the star-product PIR scheme of [14]. An alternative PPC approach was recently proposed in [20] by employing Reed-Solomon coded databases with Lagrange encoding. For low code rates, the scheme improves on the PC

Manuscript received June 10, 2021; revised November 1, 2021; accepted December 21, 2021. Date of publication January 12, 2022; date of current version February 17, 2022. This work was supported in part by U.S. NSF under Grant 1526547, Grant 1815322, and Grant 2107370; and in part by the Research Council of Norway under Grant 240985/F20. An earlier version of this paper was presented in part at the 56th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, in October 2018 [1] [DOI: 10.1109/ALLERTON.2018.8636039]. *(Corresponding author: Sarah A. Obead.)*

Sarah A. Obead and Jörg Kliewer are with the Helen and John C. Hartmann Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: sao23@njit.edu; jkliewer@njit.edu).

Hsuan-Yin Lin and Eirik Rosnes are with Simula UiB, 5006 Bergen, Norway (e-mail: lin@simula.no; eirikrosnes@simula.no).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3142362.

Digital Object Identifier 10.1109/JSAC.2022.3142362

rate of [19]. In [22], we proposed two PPC protocols inspired by the ones from [19], [20] and showed that improved PC rates can be obtained when the number of messages is small. The PC capacity for the case where the candidate functions evaluations are the stored messages plus the evaluation of an arbitrary nonlinear function of them was derived in [23]. The special case of private monomial computation (PMC) was addressed in [24], where the PMC capacity for an asymptotically large field size and under a mild technical condition on the size of the base field was derived. The technical condition on the size of the base field can be shown to be satisfied for a sufficiently large base field. Recently, PC was also extended to the single server scenario (all messages are stored uncoded on a single server) with side information in [25], [26], where the authors derived the capacity with both coded and uncoded side information under two different privacy conditions on the identities of the messages of the desired linear combination.

Finally, a separate but relevant form of PC, the private search (PS) problem [17] considers mapping records replicated over n noncolluding databases to binary search patterns. Each pattern represents the search result of one value out of a set of candidate alphabets. The asymptotic capacity, i.e., the information retrieval rate for PS with a large alphabet size, of privately retrieving one search pattern is found to match the asymptotic capacity of PIR for the special case of *balanced* PS. In a balanced PS scenario, the nonlinearly dependent search patterns are assumed to contain equal amount of information.

In another line of research, for the case of noncolluding databases, two PIR protocols for a DSS where data is stored using a non-MDS linear code, were proposed in [15], and the protocols are shown to achieve the nonasymptotic and asymptotic MDS-PIR capacity, respectively, for a large class of linear codes. The first family of non-MDS codes for which the PIR capacity is known was found in [27], [28]. Further, PIR on linearly-coded databases for the case of colluding databases was also proposed in [13]–[15], [29]. For the PC case with noncolluding databases, however, capacity results for arbitrary linearly-coded DSSs 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. Specifically, we prove a converse bound for the coded PLC capacity for a family of non-MDS storage codes considered in [15] (see Theorem 2). The significance of our PLC converse is that it depends on the rank of the coefficient matrix obtained from all μ candidate linear combinations. As a result, it is valid for any number of messages f and any number of candidate linear combinations μ. Further, a capacity-achieving PLC scheme for DSSs with noncolluding databases, where data is stored using codes from the above-mentioned family of codes, is proposed. Essentially, the proposed PLC scheme both extends the optimal PIR scheme for coded DSSs in [15] and the PLC scheme from MDS-coded DSSs in [18], strictly generalizing the replication-based PC schemes of [16], [21]. As for the optimality of the achievable PLC rate, we prove that the achievable rate matches the PLC converse bound of Theorem 2 and hence settle the coded PLC capacity (see Theorem 3).

This paper extends our previous work in [1] in several aspects and presents these novel results in a comprehensive fashion, highlighted as follows. We present the proof of Lemma 3 required for the proof of the PLC converse bound and provide a comprehensive description of the query generation algorithm, elaborating it with a detailed running example. Further, we prove the optimality of our PLC scheme through Lemma 4 and highlight the privacy-preserving features of our PLC scheme over linearly-coded DSSs.

The remainder of the paper is organized as follows. Section II outlines the notation and basic definitions, then the problem of PLC from coded DSSs and the system model are presented. We derive the converse bound for an arbitrary number of messages and linear combinations in Section III. A capacity-achieving PLC scheme for linearly-coded storage with an MDS-PIR capacity-achieving code is presented in Section IV. Some conclusions are drawn in Section V.

II. Preliminaries

A. Notation

We denote by $\mathbb{N}$ the set of all positive integers and let $\mathbb{N}_0 \triangleq \{0\} \cup \mathbb{N}$, $[a] \triangleq \{1, 2, \ldots, a\}$, and $[a : b] \triangleq \{a, a+1, \ldots, b\}$ for $a, b \in \mathbb{N}$, $a \leq b$. 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, respectively. The notation $\boldsymbol{X} \sim \boldsymbol{Y}$ is used to indicate that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are identically distributed. Random matrices are represented by bold sans serif letters, e.g., $\mathbf{X}$, where X represents its realization. In addition, sets are denoted by calligraphic uppercase letters, e.g., $\mathcal{X}$, and $\mathcal{X}^{\mathrm{c}}$ denotes the complement of a set $\mathcal{X}$ in a universe set. We denote a submatrix of X that is restricted in columns by the set $\mathcal{I}$ by $\mathsf{X}|_{\mathcal{I}}$. For a given index set $\mathcal{S}$, we also write $\mathbf{X}^{\mathcal{S}}$ and $Y_{\mathcal{S}}$ to represent $\left\{\mathbf{X}^{(v)} \colon v \in \mathcal{S}\right\}$ and $\left\{Y_j \colon j \in \mathcal{S}\right\}$, respectively. Furthermore, some constants and functions are also depicted by Greek letters or a special font, e.g., X. The function $\mathsf{H}(X)$ represents the entropy of X, and $\mathsf{I}(X;Y)$ the mutual information between X and Y. The binomial coefficient of a over b, $a, b \in \mathbb{N}_0$, is denoted by $\binom{a}{b}$ where $\binom{a}{b} = 0$ if $a < b$.

We use the customary code parameters $[n, k]$ to denote a code $\mathscr{C}$ over the finite field $\mathbb{F}_p$ of blocklength n and dimension k, where p is a power of a prime number. A generator matrix of $\mathscr{C}$ is denoted by $\mathsf{G}^{\mathscr{C}}$. A set of coordinates of $\mathscr{C}$, $\mathcal{I} \subseteq [n]$, of size k is said to be an *information set* if and only if $\mathsf{G}^{\mathscr{C}}|_{\mathcal{I}}$ is invertible. $(\cdot)^{\mathsf{T}}$ denotes the transpose operator, while $\mathrm{rank}\,(\mathsf{V})$ denotes the rank of a matrix V. The function $\chi(\boldsymbol{x})$ denotes the support of a vector $\boldsymbol{x}$, i.e., the set of indices i such that $x_i \neq 0$, and the linear span of a set of vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_a\}$, $a \in \mathbb{N}$, is denoted by $\mathrm{span}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_a\}$.

We now proceed with a general description for the problem statement of private linear function computation from linearly-coded DSSs.

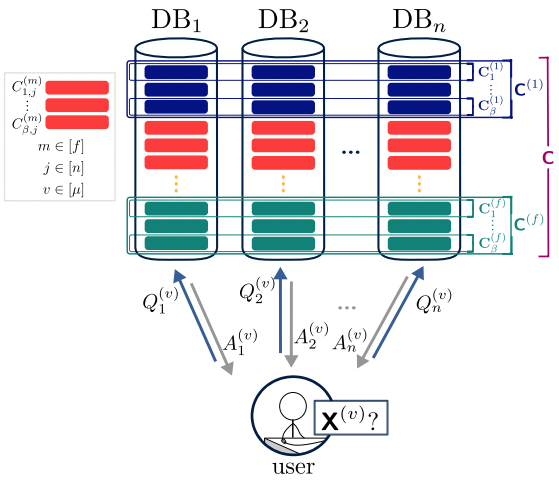

Fig. 1. System model for PLC from an $[n, k]$ coded DSS storing f messages.

B. Problem Statement and System Model

The PLC problem for coded DSSs is described as follows. We consider a DSS that stores in total f independent messages $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(f)}$, where each message $\mathbf{W}^{(m)}, m \in [f]$, consists of L symbols $W_1^{(m)}, \ldots, W_{\mathsf{L}}^{(m)}$ chosen independently and uniformly at random from $\mathbb{F}_p$. Thus, $\mathsf{H}(\mathbf{W}^{(m)}) = \mathsf{L}, \forall\, m \in [f]$ (in p-ary units). Let $\mathsf{L} \triangleq \beta k$, for some $\beta, k \in \mathbb{N}$. The DSS stores the f messages encoded using an $[n, k]$ code as follows. First, the symbols of each message $\mathbf{W}^{(m)}$, $m \in [f]$, are presented as a random $\beta \times k$ matrix over $\mathbb{F}_p$, i.e., $\mathbf{W}^{(m)} = \big(W_{i,j}^{(m)}\big), i \in [\beta], j \in [k]$. Second, let $\boldsymbol{W}_i^{(m)} = \big(W_{i,1}^{(m)}, \ldots, W_{i,k}^{(m)}\big), i \in [\beta]$, denote a message vector corresponding to the i-th row of $\mathbf{W}^{(m)}$. Each $\boldsymbol{W}_i^{(m)}$ is encoded by an $[n, k]$ code $\mathscr{C}$ over $\mathbb{F}_p$ into a length-n codeword $\boldsymbol{C}_i^{(m)} = \big(C_{i,1}^{(m)}, \ldots, C_{i,n}^{(m)}\big)$. The βf generated codewords $\boldsymbol{C}_i^{(m)}$ are then arranged in the array $\mathbf{C} = \big((\mathbf{C}^{(1)})^{\mathsf{T}} | \ldots | (\mathbf{C}^{(f)})^{\mathsf{T}}\big)^{\mathsf{T}}$ of dimensions $\beta f \times n$, where $\mathbf{C}^{(m)} = \big((\boldsymbol{C}_1^{(m)})^{\mathsf{T}} | \ldots | (\boldsymbol{C}_\beta^{(m)})^{\mathsf{T}}\big)^{\mathsf{T}}$. The code symbols $C_{1,j}^{(m)}, \ldots, C_{\beta,j}^{(m)}$, $m \in [f]$, for all f messages are stored on the j-th database, $j \in [n]$.

We consider the case of n noncolluding databases. In PLC, a user wishes to privately compute exactly one *linear* function evaluation $\mathbf{X}^{(v)} = \big(X_1^{(v)}, \ldots, X_{\mathsf{L}}^{(v)}\big)$, out of μ *candidate* linear combinations $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\mu)}$ from the coded DSS. The μ-tuple $\big(X_l^{(1)}, \ldots, X_l^{(\mu)}\big)^{\mathsf{T}}$, $\forall\, l \in [\mathsf{L}]$, is mapped by a deterministic matrix V of size $\mu \times f$ over $\mathbb{F}_p$ by

$$\big(X_l^{(1)}, \ldots, X_l^{(\mu)}\big)^{\mathsf{T}} = \mathsf{V}_{\mu \times f}\big(W_l^{(1)}, \ldots, W_l^{(f)}\big)^{\mathsf{T}}. \tag{1}$$

The user privately selects an index $v \in [\mu]$ and wishes to compute the v-th function while keeping the requested function 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 $\mathcal{L} \triangleq \{\ell_1, \ldots, \ell_r\} \subseteq [\mu]$. In order to retrieve the desired linear combination $\mathbf{X}^{(v)}$, $v \in [\mu]$, from the coded DSS, the user sends a query $Q_j^{(v)}$ to the j-th database for all $j \in [n]$ as illustrated in Fig. 1. Since the queries are generated by the user without any prior knowledge of the realizations of the candidate functions, the queries are independent of the candidate linear combinations. In other words, we have

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

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, by the data processing inequality, $\forall\, v \in [\mu]$,

$$\mathrm{H}\Big(A_j^{(v)} \,\Big|\, Q_j^{(v)}, \boldsymbol{C}_j\Big) = \mathrm{H}\Big(A_j^{(v)} \,\Big|\, Q_j^{(v)}, \mathbf{W}^{[f]}\Big) = 0, \quad \forall\, j \in [n],$$

where $\boldsymbol{C}_j \triangleq \big(C_{1,j}^{(1)}, \ldots, C_{\beta,j}^{(1)}, C_{1,j}^{(2)}, \ldots, C_{\beta,j}^{(f)}\big)^{\mathsf{T}}$ denotes the f coded chunks that are stored in the j-th database.

To preserve user's privacy, the query-answer function must be identically distributed for each possible desired function index $v \in [\mu]$ from the perspective of each database $j \in [n]$. In other words, the queries and answer strings of the scheme corresponding to each database must be independent from the desired function index. Moreover, the user must be able to reliably decode the desired linear function evaluation $\mathbf{X}^{(v)}$. Accordingly, we define a PLC protocol for $[n, k]$ coded DSSs as follows.

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

$$\text{[Privacy]}\ (Q_j^{(v)}, A_j^{(v)}, \mathbf{X}^{[\mu]}) \sim (Q_j^{(v')}, A_j^{(v')}, \mathbf{X}^{[\mu]}), \tag{2a}$$

$$\text{[Recovery]}\ \mathrm{H}\big(\mathbf{X}^{(v)} \,\big|\, A_{[n]}^{(v)}, Q_{[n]}^{(v)}\big) = 0. \tag{2b}$$

From an information-theoretic perspective, the efficiency of a PLC protocol is measured by the *PLC rate*, which is defined as follows.

Definition 1 (PLC Rate and Capacity for Linearly-Coded DSSs): The rate of a PLC scheme, denoted by R*, is defined as the ratio of the desired function size* L *over the total required download cost, i.e.,* $\mathsf{R} \triangleq \frac{\mathsf{L}}{\mathsf{D}}$*, where* D *is the total required download cost. The PLC* capacity *is the maximum of all achievable PLC rates over all possible PLC protocols for a given* $[n, k]$ *storage code.*

C. MDS-PIR Capacity-Achieving Codes

In [15], a PIR protocol for any linearly-coded DSS that uses an $[n, k]$ code to store f messages, named Protocol 1, is proposed. The PIR rate of Protocol 1 can be derived by finding a *PIR achievable rate matrix* of the underlying storage code $\mathscr{C}$, which is defined as follows.

Definition 2 ([15, Def. 10]): Let $\mathscr{C}$ *be an arbitrary* $[n, k]$ *code. A* $\nu \times n$ *binary matrix* $\Lambda_{\kappa,\nu}^{\mathrm{PIR}}(\mathscr{C})$ *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* $\Lambda_{\kappa,\nu}^{\mathrm{PIR}}$ *is* κ*, and*
2) *for each matrix row* $\boldsymbol{\lambda}_i$*,* $i \in [\nu]$*,* $\chi(\boldsymbol{\lambda}_i)$ *always contains an information set of* $\mathscr{C}$*, where* $\chi(\boldsymbol{\lambda}_i)$ *denotes the support of the vector* $\boldsymbol{\lambda}_i$*.*

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

Example 1: Consider a $[4,2]$ code $\mathscr{C}$ with generator matrix $\mathsf{G}^{\mathscr{C}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$. One can verify that $\Lambda_{1,2}^{\text{PIR}} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$ is a valid PIR achievable rate matrix for $\mathscr{C}$ with $(\kappa, \nu) = (1, 2)$. This is true given that, column-wise, the Hamming weight of each column in $\Lambda_{1,2}^{\text{PIR}}$ is $\kappa = 1$. On the other hand, row-wise, $\chi(\boldsymbol{\lambda}_1) = \{1, 3\}$ and $\chi(\boldsymbol{\lambda}_2) = \{2, 4\}$ are two information sets of $\mathscr{C}$. $\triangledown$

In [15], it is shown that the MDS-PIR capacity [12] 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 ([15, Cor. 1]): Consider a DSS that uses an $[n, k]$ code $\mathscr{C}$ to store f messages. If a PIR achievable rate matrix $\Lambda_{\kappa,\nu}^{\text{PIR}}(\mathscr{C})$ 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 3 ([15, Def. 13]): Given an $[n, k]$ code $\mathscr{C}$, if a PIR achievable rate matrix $\Lambda_{\kappa,\nu}^{\text{PIR}}(\mathscr{C})$ with $\frac{\kappa}{\nu} = \frac{k}{n}$ exists, then the code $\mathscr{C}$ is referred to as an MDS-PIR capacity-achieving *code, and the matrix $\Lambda_{\kappa,\nu}^{\text{PIR}}(\mathscr{C})$ is called an* MDS-PIR capacity-achieving *matrix.*

Accordingly, one can easily see that the $[4, 2]$ code $\mathscr{C}$ given in Example 1 is an MDS-PIR capacity-achieving code. 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 [15]. In Section IV, we present a PLC protocol and a general achievable rate for any $\operatorname{rank}(\mathsf{V}) = r$ by using the PIR achievable rate matrix $\Lambda_{\kappa,\nu}^{\text{PIR}}$ of an $[n, k]$ code.

III. Converse Bound

In [27], [28], the PIR capacity for a coded DSS using an MDS-PIR capacity-achieving code is shown to be equal to the MDS-PIR capacity. In this section, we derive a converse bound for the PLC rate (Theorem 2 below) by adapting the converse proof of [28, Thm. 4] to the linearly-coded PLC problem, where the storage code is an MDS-PIR capacity-achieving code. Then, we show that the PLC capacity matches the MDS-PIR capacity (i.e., the PIR capacity for a DSS where data is encoded and stored using an MDS code). The converse is valid for any number of messages f and candidate linear functions μ. The following theorem states an upper bound on the PLC capacity for a coded DSS where data is stored using an MDS-PIR capacity-achieving code.

Theorem 2: Consider a DSS with n noncolluding databases that uses an $[n, k]$ MDS-PIR capacity-achieving code $\mathscr{C}$ to store f messages. Then, the rate R of any PLC protocol is upper bounded by

$$\mathsf{R} \leq \mathsf{C}_{\text{PLC}} \triangleq \left[1 + \sum_{v=1}^{r-1} \left(\frac{k}{n}\right)^v\right]^{-1} = \left(1 - \frac{k}{n}\right) \left[1 - \left(\frac{k}{n}\right)^r\right]^{-1},$$

where r is the rank of the linear mapping from (1)*.*

Note that by simply assuming that the candidate functions are linearly independent linear combinations, i.e., $\mu = r$, the PLC problem reduces to PIR from $[n, k]$ linearly-encoded DSSs. If these linear combinations are also uniformly distributed, the proof of Theorem 2 follows directly from the PIR capacity of [28, Thm. 4]. However, by providing a formal proof for Theorem 2, we confirm that with added computation, i.e., $\mu > r$, we can not achieve a better rate. In the following, we present a general converse proof for *dependent* messages and detail the conditions that lead to this conclusion. Before we proceed with the converse proof, we provide some general results.

1) From the condition of privacy,
$$\mathsf{H}\big(A_j^{(v)} \mid \mathbf{X}^{(v)}, \mathcal{Q}\big) = \mathsf{H}\big(A_j^{(v')} \mid \mathbf{X}^{(v)}, \mathcal{Q}\big), \tag{3}$$
where $v \neq v'$, $v, v' \in [\mu]$, and $\mathcal{Q} \triangleq \big\{Q_j^{(v)} \colon v \in [\mu], j \in [n]\big\}$ denotes the set of all queries. Although this seems to be intuitively true, a proof of this property is still required and can be found in [12].
2) Consider a PLC protocol for a coded DSS that uses an $[n, k]$ code $\mathscr{C}$ to store f messages.
Lemma 1 (Independence of Answers From k Databases Forming an Information Set): For any information set $\mathcal{I} \subseteq [n]$, $|\mathcal{I}| = k$, *of the $[n, k]$ linear storage code $\mathscr{C}$, and for any $v \in [\mu]$,*
$$\mathsf{H}\big(A_{\mathcal{I}}^{(v)} \mid \mathcal{Q}\big) = \sum_{j \in \mathcal{I}} \mathsf{H}\big(A_j^{(v)} \mid \mathcal{Q}\big). \tag{4}$$
Moreover, (4) *is true conditioned on any subset of linear combinations $\mathbf{X}^{\mathcal{V}}$, $\mathcal{V} \subseteq [\mu]$, i.e.,*
$$\mathsf{H}\big(A_{\mathcal{I}}^{(v)} \mid \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) = \sum_{j \in \mathcal{I}} \mathsf{H}\big(A_j^{(v)} \mid \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big). \tag{5}$$
The proof of Lemma 1 is a simple extension of [12, Lem. 1] based on [11, Lem. 1] and is presented in Appendix A.

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

Lemma 2 (Shearer's Lemma [30]): Let $\mathscr{S}$ be a collection of subsets of $[n]$, with each $j \in [n]$ 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 is presented in Appendix B.

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 $\big(\mathbf{X}^{(i_1)}, \ldots, \mathbf{X}^{(i_h)}\big)$ and $\big(\mathbf{W}^{(j_1)}, \ldots, \mathbf{W}^{(j_h)}\big)$ are identically distributed, for some $h \in [r]$. In other words,* $\mathsf{H}\big(\mathbf{X}^{(i_1)}, \ldots, \mathbf{X}^{(i_h)}\big) = h\mathsf{L}$, $h \in [r]$.

Now, we are ready for the converse proof. By [15, 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 $j \in [n]$ 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 $\mathsf{H}\big(A_{[n]}^{(v)} \mid \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \geq \mathsf{H}\big(A_{\mathcal{I}_i}^{(v)} \mid \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big)$, $\forall\, i \in [\nu]$, where $\mathcal{V} \subseteq [\mu]$ is arbitrary.

Let $v \in \mathcal{V}$ and $v' \in \mathcal{V}^{\mathrm{c}} \triangleq [\mu] \setminus \mathcal{V}$. Following similar steps as in the proof given in [12], [31], we get

$$\begin{aligned}
&\nu\, \mathsf{H}\big(A_{[n]}^{(v)} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \\
&\geq \sum_{i=1}^{\nu} \mathsf{H}\big(A_{\mathcal{I}_i}^{(v)} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \\
&\overset{(a)}{=} \sum_{i=1}^{\nu} \left(\sum_{j \in \mathcal{I}_i} \mathsf{H}\big(A_j^{(v)} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \right) \overset{(b)}{=} \sum_{i=1}^{\nu} \left(\sum_{j \in \mathcal{I}_i} \mathsf{H}\big(A_j^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \right) \\
&\overset{(a)}{=} \sum_{i=1}^{\nu} \mathsf{H}\big(A_{\mathcal{I}_i}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \overset{(c)}{\geq} \kappa\, \mathsf{H}\big(A_{[n]}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \\
&= \kappa \Big[\mathsf{H}\big(A_{[n]}^{(v')}, \mathbf{X}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) - \mathsf{H}\big(\mathbf{X}^{(v')} \,\big|\, A_{[n]}^{(v')}, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \Big] \\
&\overset{(d)}{=} \kappa \Big[\mathsf{H}\big(\mathbf{X}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) + \mathsf{H}\big(A_{[n]}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathbf{X}^{(v')}, \mathcal{Q}\big) - 0 \Big] \\
&\overset{(e)}{=} \kappa \Big[\mathsf{H}\big(\mathbf{X}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}\big) + \mathsf{H}\big(A_{[n]}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathbf{X}^{(v')}, \mathcal{Q}\big) \Big],
\end{aligned}$$

where (a) follows from (5); (b) is because of (3); (c) is due to Shearer's Lemma; (d) is from the fact that the v'-th linear combination $\mathbf{X}^{(v')}$ is determined by the answers $A_{[n]}^{(v')}$ and all possible queries $\mathcal{Q}$; and finally, (e) follows from the independence between all possible queries and the messages. Therefore, we can conclude that

$$\begin{aligned}
&\mathsf{H}\big(A_{[n]}^{(v)} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathcal{Q}\big) \\
&\geq \frac{\kappa}{\nu} \mathsf{H}\big(\mathbf{X}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}\big) + \frac{\kappa}{\nu} \mathsf{H}\big(A_{[n]}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathbf{X}^{(v')}, \mathcal{Q}\big) \\
&= \frac{k}{n} \mathsf{H}\big(\mathbf{X}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}\big) + \frac{k}{n} \mathsf{H}\big(A_{[n]}^{(v')} \,\big|\, \mathbf{X}^{\mathcal{V}}, \mathbf{X}^{(v')}, \mathcal{Q}\big), \qquad (6)
\end{aligned}$$

where we have used Definition 3 to obtain (6).

Since there are in total μ linear combinations and $\mathcal{L} \triangleq \{\ell_1, \ldots, \ell_r\} \subseteq [\mu]$ is the set of row indices corresponding to the selected basis for the row space of V, we can recursively use (6) $r-1$ times to obtain

$$\begin{aligned}
&\mathsf{H}\big(A_{[n]}^{(\ell_1)} \,\big|\, \mathbf{X}^{(\ell_1)}, \mathcal{Q}\big) \\
&\geq \sum_{v=1}^{r-1} \Big(\frac{k}{n}\Big)^v \mathsf{H}\big(\mathbf{X}^{(\ell_{v+1})} \,\big|\, \mathbf{X}^{\{\ell_1, \ldots, \ell_v\}}\big) \\
&\quad + \Big(\frac{k}{n}\Big)^{r-1} \mathsf{H}\big(A_{[n]}^{(\ell_r)} \,\big|\, \mathbf{X}^{\{\ell_1, \ldots, \ell_r\}}, \mathcal{Q}\big) \\
&\overset{(a)}{\geq} \sum_{v=1}^{r-1} \Big(\frac{k}{n}\Big)^v \mathsf{H}\big(\mathbf{X}^{(\ell_{v+1})} \,\big|\, \mathbf{X}^{\{\ell_1, \ldots, \ell_v\}}\big) \overset{(b)}{=} \sum_{v=1}^{r-1} \Big(\frac{k}{n}\Big)^v \mathsf{L}, \qquad (7)
\end{aligned}$$

where (a) follows from the nonnegativity of entropy, and (b) holds since $\mathsf{H}\big(\mathbf{X}^{(\ell_{v+1})} \,\big|\, \mathbf{X}^{\{\ell_1, \ldots, \ell_v\}}\big) = \mathsf{H}\big(\mathbf{X}^{(\ell_{v+1})}\big) = \mathsf{L}$ (see Lemma 3). Here, we also remark that the recursive steps follow the same principle of the general converse for DPIR from [17, Thm. 1]. In [17], 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, for the PLC problem, the index permutation of the candidate linear functions intuitively follows from finding a basis for V. Now,

$$\begin{aligned}
\mathsf{L} = \mathsf{H}\big(\mathbf{X}^{(\ell_1)}\big) &\overset{(a)}{=} \mathsf{H}\big(\mathbf{X}^{(\ell_1)} \,\big|\, \mathcal{Q}\big) - \underbrace{\mathsf{H}\big(\mathbf{X}^{(\ell_1)} \,\big|\, A_{[n]}^{(\ell_1)}, \mathcal{Q}\big)}_{=0} \\
&= \mathsf{H}\Big(A_{[n]}^{(\ell_1)} \,\Big|\, \mathcal{Q}\Big) - \mathsf{H}\Big(A_{[n]}^{(\ell_1)} \,\Big|\, \mathbf{X}^{(\ell_1)}, \mathcal{Q}\Big) \\
&\overset{(b)}{\leq} \mathsf{H}\Big(A_{[n]}^{(\ell_1)} \,\Big|\, \mathcal{Q}\Big) - \sum_{v=1}^{r-1} \Big(\frac{k}{n}\Big)^v \mathsf{L}, \qquad (8)
\end{aligned}$$

where (a) follows since any message is independent of the queries $\mathcal{Q}$, and by knowing the answers $A_{[n]}^{(\ell_1)}$ and the queries $\mathcal{Q}$, one can determine $\mathbf{X}^{(\ell_1)}$, and (b) holds because of (7).

Finally, the converse proof is completed by showing that

$$\begin{aligned}
\mathsf{R} = \frac{\mathsf{L}}{\sum_{j=1}^{n} \mathsf{H}\big(A_j^{(\ell_1)}\big)} \leq \frac{\mathsf{L}}{\mathsf{H}\big(A_{[n]}^{(\ell_1)}\big)} &\overset{(a)}{\leq} \frac{\mathsf{L}}{\mathsf{H}\big(A_{[n]}^{(\ell_1)} \,\big|\, \mathcal{Q}\big)} \\
&\overset{(b)}{\leq} \frac{1}{1 + \sum_{v=1}^{r-1} \big(\frac{k}{n}\big)^v} = \mathsf{C}_{\text{PLC}},
\end{aligned}$$

where (a) is due to the fact that conditioning reduces entropy, and we apply (8) to obtain (b).

It can be easily seen that the converse bound of Theorem 2 matches the MDS-PIR capacity $\mathsf{C}_{\text{MDS-PIR}}$ for $f = r$ files given in Theorem 1. The capacity-achieving PLC scheme is provided in the following section.

IV. Private Linear Computation From Coded DSSs

One of the main results 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 [15]. Based on the PLC converse bound of Theorem 2, in this section we construct a capacity-achieving PLC scheme. Our capacity-achieving PLC scheme is also a generalization of the replication-based PLC scheme in [16]. Although the two schemes are build upon a different PIR construction, both schemes adapt the underlying PIR construction for *dependent* virtual messages through an *index assignment* structure. Moreover, in order to optimize the download rate, both schemes deploy a *sign assignment* structure to induce redundancy within the queries of the modified underlying PIR construction. In this section, we first present our modified underlying PIR construction with Algorithm 1 in Section IV-A. Then, we elaborate on the sign assignment procedure in Section IV-C. In Theorem 3 we settle the PLC capacity for a DSS where data is stored using an MDS-PIR capacity-achieving code.

Theorem 3: Consider a DSS with n noncolluding databases that uses an $[n, k]$ MDS-PIR capacity-achieving code $\mathscr{C}$ to store f messages. Then, the PLC capacity is equal to C_{PLC}, where r is the rank of the linear mapping from (1)*.*

We remark that since all MDS codes are MDS-PIR capacity-achieving codes, it follows that if $\mathrm{rank}\,(\mathsf{V}) = f$, then the PLC capacity for an MDS-coded DSS [18] is equal to the MDS-PIR capacity $\mathsf{C}_{\text{MDS-PIR}}$.

We now start by constructing a query generation algorithm for a coded PIR-like scheme, where its *dependent virtual* messages represent the evaluations of the μ candidate linear combinations. A PIR-like scheme achieves a private retrieval of the desired *virtual* message by following three important design principles: 1) Enforcing symmetry across databases. Each database is queried for an equal number of symbols and the query structure does not depend on the individual

database, i.e., the scheme structure is fixed for all databases. 2) Enforcing symmetry across virtual messages. 3) Exploiting side information represented by undesired information downloaded to maintain message symmetry.

Given that the messages are stored using an $[n, k]$ MDS-PIR capacity-achieving code $\mathscr{C}$, we can construct a $\nu \times n$ MDS-PIR capacity-achieving matrix $\Lambda^{\text{PIR}}_{\kappa,\nu}$ of Definition 2, 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 4 ([15]): For a given $\nu \times n$ PIR achievable rate matrix $\Lambda^{\text{PIR}}_{\kappa,\nu}(\mathscr{C}) = (\lambda_{u,j})$, 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 $\mathscr{C}$ as

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

Note that in Definition 4, for each $j \in [n]$, distinct values of $u \in [\nu]$ 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. Moreover, for $j \in [n]$, let $\mathcal{A}_j \triangleq \{a_{i,j} \colon i \in [\kappa]\}$ and $\mathcal{B}_j \triangleq \{b_{i,j} \colon i \in [\nu-\kappa]\}$. Note that the j-th column of $\mathsf{A}_{\kappa\times n}$ contains the row indices of $\Lambda^{\text{PIR}}_{\kappa,\nu}$ whose entries in the j-th column are equal to 1, while $\mathsf{B}_{(\nu-\kappa)\times n}$ contains the remaining row indices of $\Lambda^{\text{PIR}}_{\kappa,\nu}$. Hence, it can be observed that $\mathcal{B}_j = [\nu] \setminus \mathcal{A}_j$, $\forall\, j \in [n]$.

Next, for the sake of illustrating our query generation algorithm, we make use of the following definition.

Definition 5: 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., $\mathcal{S}(u|\mathsf{A}_{\kappa\times n}) \triangleq \{j \in [n] \colon \exists\, a_{i,j} = u, i \in [\kappa]\}$.

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

A. Query Generation for PLC

Before running the main algorithm to generate the 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 symbols queried from each database to appear to be chosen randomly and independently from the desired linear function index. Note that the function is computed separately for the t-th row of all messages, $t \in [\beta]$. Therefore, similar to the PLC scheme in [16] and the MDS-coded PLC scheme in [18], 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 $[\beta]$, and let

$$U^{(v')}_{t,j} \triangleq \boldsymbol{v}_{v'} \boldsymbol{C}_{\pi(t),j}, \quad t \in [\beta],\ j \in [n],\ v' \in [\mu], \tag{9}$$

denote the t-th permuted symbol associated with the v'-th virtual message $\mathbf{X}^{(v')}$ stored in the j-th database, where $\boldsymbol{C}_{t,j} \triangleq \left(C^{(1)}_{t,j}, \ldots, C^{(f)}_{t,j}\right)^{\intercal}$ and $\boldsymbol{v}_{v'}$ represents the v'-th row vector of the matrix $\mathsf{V}_{\mu\times f} = (v_{i,j})$. The permutation $\pi(\cdot)$ is randomly selected privately and uniformly by the user.

2) Preliminaries: The query generation procedure is subdivided into μ rounds, where in each round τ we generate the queries based on the concept of *τ-sums* as defined in the following.

Definition 6 (τ-sum): For $\tau \in [\mu]$, a sum $U^{(v_1)}_{i_1,j} + U^{(v_2)}_{i_2,j} + \cdots + U^{(v_\tau)}_{i_\tau,j}$, $j \in [n]$, of τ distinct symbols is called a τ-sum for any $(i_1, \ldots, i_\tau) \in [\beta]^\tau$, and $\{v_1, \ldots, v_\tau\} \subseteq [\mu]$ 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*. For a requested linear function evaluation indexed by $v \in [\mu]$, a query set $Q^{(v)}_j$, $j \in [n]$, is composed of μ disjoint subsets of queries, each subset of queries is generated by the operations of each round $\tau \in [\mu]$. In a round we generate the queries for all possible $\binom{\mu}{\tau}$ types of τ-sums. For each round $\tau \in [\mu]$ the corresponding query subset is further subdivided into two subsets $Q^{(v)}_j(\mathcal{D};\tau)$ and $Q^{(v)}_j(\mathcal{U};\tau)$. The first subset $Q^{(v)}_j(\mathcal{D};\tau)$ corresponds to τ-sums with a single symbol from the *desired* function evaluation and $\tau - 1$ symbols from the evaluations of *undesired* functions, while the second subset $Q^{(v)}_j(\mathcal{U};\tau)$ corresponds to τ-sums with symbols only from the evaluations of undesired functions. Here, $\mathcal{D}$ is an indicator for "desired function evaluations," while $\mathcal{U}$ an indicator for "undesired functions evaluations." Note that we require $\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}$ distinct instances of each τ-sum type for every query set $Q^{(v)}_j$. To this end, the algorithm will generate κn auxiliary query sets $Q^{(v)}_j(a_{i,j},\mathcal{D};\tau)$, $i \in [\kappa]$, where each query requests a distinct symbol from the desired function evaluation and $\tau - 1$ symbols from undesired functions evaluations, and $(\nu-\kappa)n$ auxiliary query sets $Q^{(v)}_j(b_{i,j},\mathcal{U};\tau)$, $i \in [\nu-\kappa]$, to represent the query sets of symbols from the undesired functions evaluations for each database $j \in [n]$. We utilize these sets to generate the query sets of each round according to the PIR interference matrices $\mathsf{A}_{\kappa\times n}$ and $\mathsf{B}_{(\nu-\kappa)\times n}$.

To illustrate the key concepts of the coded PLC scheme, we use the following example, i.e., Example 2, as a running example for this section.

Example 2: Consider $f = 4$ messages $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$, and $\mathbf{W}^{(4)}$ that are stored in a DSS using the $[4, 2]$ MDS-PIR capacity-achieving code $\mathscr{C}$ given in Example 1 for which $\Lambda^{\text{PIR}}_{1,2} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$, $\mathsf{A}_{1\times 4} = (\,1 \quad 2 \quad 1 \quad 2)$, and $\mathsf{B}_{1\times 4} = (\,2 \quad 1 \quad 2 \quad 1)$, are a PIR achievable rate matrix with $(\kappa, \nu) = (1, 2)$ and the corresponding PIR interference matrices $\mathsf{A}_{1\times 4}$ and $\mathsf{B}_{1\times 4}$, respectively, according to Definition 4. Suppose that the user wishes to obtain a linear function evaluation $\mathbf{X}^{(v)}$ from a set of $\mu = 4$ candidate linear functions, whose $\mathsf{V}_{\mu\times f}$ from (1) is given by

$$\mathsf{V}_{4\times 4} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 \\ 4 & 1 & 0 & 3 \end{pmatrix}.$$

We simplify notation by letting $x_{t,j} = U^{(1)}_{t,j}$, $y_{t,j} = U^{(2)}_{t,j}$, $z_{t,j} = U^{(3)}_{t,j}$, and $w_{t,j} = U^{(4)}_{t,j}$ for all $t \in [\beta]$, $j \in [n]$, where $\beta = \nu^\mu = 16$. First, let the desired linear function index be $v = 1$. $\triangledown$

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 in lines 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 $Q_j^{(v)}(a_{i,j}, \mathcal{D}; 1)$, $i \in [\kappa]$, such that each of these query sets contains $\alpha_1 = \kappa^{\mu-1}$ distinct symbols. Furthermore, to maintain function symmetry, the algorithm asks each database for the same number of distinct symbols of all other linear functions evaluations in $Q_j^{(v)}(a_{i,j}, \mathcal{U}; 1)$, $i \in [\kappa]$, resulting in a total number of $\binom{\mu-1}{1}\kappa^{\mu-1}$ symbols. As a result, the queried symbols in the auxiliary query sets for each database are symmetric with respect to all function evaluation vectors indexed by $v' \in [\mu]$.

In the following steps, we will associate the symbols of undesired functions evaluations in κ groups, each placed in the undesired query sets $Q_j^{(v)}(a_{i,j}, \mathcal{U}; 1)$, $i \in [\kappa]$. Since this procedure produces κ undesired query sets for each database, database symmetry is maintained.

Example 2 (Continued): The initialization step is described in the following. Algorithm 1 starts with $\tau = 1$ to generate auxiliary query sets $Q_j^{(v)}(a_{i,j}, \mathcal{D}; 1)$, $Q_j^{(v)}(a_{i,j}, \mathcal{U}; 1)$, $i \in [\kappa]$, for each database $j \in [n]$. Starting at line 6 of Algorithm 1, since $\nu = 2$, we have the row indicator $u \in [2]$. This indicator is first used to identify the code coordinates pertaining to different entries $u = a_{i,j}$, as specified by the interference matrix $\mathsf{A}_{1\times 4}$. For example, when $u = 1$, following Definition 5, we have $\mathcal{S}(1|\mathsf{A}_{\kappa\times n}) = \{1, 3\}$. In line 7 of Algorithm 1, for $j \in \{1, 3\}$, algorithm `Initial-Round` is invoked to generate the desired and undesired query subsets $Q_j^{(1)}(1, \mathcal{D}; 1)$ and $Q_j^{(1)}(1, \mathcal{U}; 1)$. The set $Q_j^{(1)}(1, \mathcal{D}; 1)$ queries $\alpha_1 = \kappa^{\mu-1} = 1$ distinct instances of the desired function evaluation $x_{t,j}$ and the set $Q_j^{(1)}(1, \mathcal{U}; 1)$ $\alpha_1 = 1$ distinct instances of the remaining linear functions evaluations $y_{t,j}$, $z_{t,j}$, and $w_{t,j}$. To this end, the row indicator u is passed to the subroutine `Initial-Round`, i.e., Algorithm 2, where it is used to determine the indices of the queried symbols. For example, the first auxiliary query set for $u = 1$ generated by Algorithm 2 is given by $Q_j^{(1)}(1, \mathcal{D}; 1) = \{U^{(1)}_{(1-1)\cdot 1+1,j}\} = \{x_{1,j}\}$, $j \in \{1, 3\}$. A similar process is followed for $Q_j^{(1)}(1, \mathcal{U}; 1)$. The same process is then repeated for $u = 2$. By the end of this step, we have queried $\nu\alpha_1 = 2$ distinct instances of the desired function evaluation $x_{t,j}$ and by message symmetry, $\nu\alpha_1 = 2$ distinct instances of the remaining functions evaluations $y_{t,j}$, $z_{t,j}$, and $w_{t,j}$. In total, the first round of queries comprises $n\kappa\alpha_1\mu = 16$ symbols, which can be written in the form $n\binom{\mu}{1}\kappa^{\mu-1+1}(\nu-\kappa)^{1-1}$. The resulting auxiliary query sets for the first round of queries

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

Algorithm 1 `Q-Gen`

Input : v, μ, κ, ν, n, $\mathsf{A}_{\kappa\times n}$, and $\mathsf{B}_{(\nu-\kappa)\times n}$
Output: $Q_1^{(v)}, \ldots, Q_n^{(v)}$

1 **for** $\tau \in [\mu]$ **do**
2 $Q_j^{(v)}(\mathcal{D}; \tau) \leftarrow \emptyset$, $Q_j^{(v)}(\mathcal{U}; \tau) \leftarrow \emptyset$, $j \in [n]$
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** $u \in [\nu]$ **do**
7 **for** $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ **do**
8 $Q_j^{(v)}(u, \mathcal{D}; \tau), Q_j^{(v)}(u, \mathcal{U}; \tau) \leftarrow$ `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** $u \in [\nu]$ **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 $Q_j^{(v)}(u, \mathcal{D}; \tau) \leftarrow$ `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 $Q_j^{(v)}(u, \mathcal{U}; \tau-1) \leftarrow$ `Exploit-SI`$(u, Q_1^{(v)}(u, \mathcal{U}, \tau-1), \ldots, Q_n^{(v)}(u, \mathcal{U}, \tau-1), j, v, \tau)$
22 **end**
23 **end**
24 ▷ Generate the final desired query sets for the following rounds $\tau > 1$
25 **for** $j \in [n]$ **do**
26 $\tilde{Q}_j^{(v)}(\mathcal{U}; \tau-1) \leftarrow \bigcup_{i\in[\nu-\kappa]} Q_j^{(v)}(b_{i,j}, \mathcal{U}; \tau-1)$
27 $\tilde{Q}_j^{(v)}(1, \mathcal{U}; \tau-1), \ldots, \tilde{Q}_j^{(v)}(\kappa, \mathcal{U}; \tau-1) \leftarrow$ `Partition`$\big(\tilde{Q}_j^{(v)}(\mathcal{U}; \tau-1)\big)$
28 **for** $i \in [\kappa]$ **do**
29 $Q_j^{(v)}(a_{i,j}, \mathcal{D}; \tau) \leftarrow$ `SetAddition`$\big(Q_j^{(v)}(a_{i,j}, \mathcal{D}; \tau), \tilde{Q}_j^{(v)}(i, \mathcal{U}; \tau-1)\big)$
30 **end**
31 **end**
32 ▷ Generate the query sets of undesired symbols by forcing message symmetry for the following rounds $\tau > 1$
33 **for** $u \in [\nu]$ **do**
34 **for** $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ **do**
35 $Q_j^{(v)}(u, \mathcal{U}; \tau) \leftarrow$ `M-Sym`$\big(Q_j^{(v)}(u, \mathcal{D}; \tau), j, v, \tau\big)$
36 **end**
37 **end**
38 **end**
39 **for** $u \in [\nu]$ **do**
40 **for** $j \in \mathcal{S}(u|\mathsf{A}_{\kappa\times n})$ **do**
41 $Q_j^{(v)}(\mathcal{D}; \tau) \leftarrow Q_j^{(v)}(\mathcal{D}; \tau) \cup Q_j^{(v)}(u, \mathcal{D}; \tau)$
42 $Q_j^{(v)}(\mathcal{U}; \tau) \leftarrow Q_j^{(v)}(\mathcal{U}; \tau) \cup Q_j^{(v)}(u, \mathcal{U}; \tau)$
43 **end**
44 **end**
45 **end**
46 **for** $j \in [n]$ **do**
47 $Q_j^{(v)} \leftarrow \bigcup_{\tau\in[\mu]} \Big(Q_j^{(v)}(\mathcal{D}; \tau) \cup Q_j^{(v)}(\mathcal{U}; \tau)\Big)$
48 **end**

are shown in Table I(a), where we highlight in red the row indicator $u \in [\nu]$ as specified by the interference matrix $\mathsf{A}_{1\times 4}$, i.e., $u = a_{1,j}$. ▽

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 evaluation $\mathbf{U}^{(v)} = (U_{t,j}^{(v)})$. This is accomplished in lines 16 to 18 by calling the subroutine `Desired-Q`, given in Algorithm 3, to generate $Q_j^{(v)}(a_{i,j}, \mathcal{D}; \tau)$, $i \in [\kappa]$, 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 symbols from the desired linear function evaluation $\mathbf{U}^{(v)}$.

Example 2 (Continued): After successfully generating the queries for $\nu\alpha_1 = 2$ distinct symbols from the desired linear function evaluation in the initiation step, for round $\tau = 2$ we generate the queries for the following $\nu(\alpha_2 - \alpha_1) = 6$ symbols. To this end, subroutine `Desired-Q`, given in Algorithm 3, generates auxiliary query sets $Q_j^{(1)}(a_{i,j}, \mathcal{D}; 2)$ containing distinct symbols from the desired linear function evaluation, following a process similar to Algorithm 2, however with a different method for determining the queried indices. The output of lines 16 to 18 after calling the subroutine `Desired-Q` for $u \in [2]$ is as follows. ▽

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

5) Side Information Exploitation: In lines 20 to 22, we generate the *side information* query sets $Q_j^{(v)}(b_{i',j}, \mathcal{U}; \tau - 1)$, $i' \in [\nu - \kappa]$, from the auxiliary query sets $Q_1^{(v)}(a_{i,1}, \mathcal{U}; \tau - 1), \ldots, Q_n^{(v)}(a_{i,n}, \mathcal{U}; \tau - 1)$, $i \in [\kappa]$, of the previous round $\tau - 1$, $\tau \in [2 : \mu]$, by applying the subroutine `Exploit-SI`, given by Algorithm 4. This subroutine is extended from [16] based on our coded storage scenario. These side information query sets will be exploited by the user to ensure the recovery and privacy of the PLC scheme. Note that in Algorithm 4 the function $\texttt{Reproduce}(j, Q_{j'}^{(v)}(u, \mathcal{U}; \tau - 1))$, $j' \in [n] \setminus \{j\}$, simply reproduces all the queries in the auxiliary query set $Q_{j'}^{(v)}(u, \mathcal{U}; \tau - 1)$ with a different coordinate j.

Next, we update the desired query sets $Q_j^{(v)}(a_{i,j}, \mathcal{D}; \tau)$ in lines 25 to 31. First, the function $\texttt{Partition}\big(\tilde{Q}_j^{(v)}(\mathcal{U}; \tau - 1)\big)$ 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 $\big|\tilde{Q}_j^{(v)}(\mathcal{U}; \tau - 1)\big| = \binom{\mu-1}{\tau-1}\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{(\tau-1)-1} \cdot (\nu - \kappa)$ for each round $\tau \in [2 : \mu]$, which is always divisible by κ. Secondly, we assign the new query set of desired symbols $Q_j^{(v)}(a_{i,j}, \mathcal{D}; \tau)$ for the current round by using an element-wise set addition $\texttt{SetAddition}(Q_1, Q_2)$. The element-wise set addition is defined as $\{q_{i_l} + q_{i'_l} : q_{i_l} \in Q_1, q_{i'_l} \in Q_2, l \in [\rho]\}$ with $|Q_1| = |Q_2| = \rho$, where ρ is an appropriate integer.

Algorithm 2 `Initial-Round`

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

Algorithm 3 `Desired-Q`

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

Algorithm 4 `Exploit-SI`

Input : u, $Q_1^{(v)}(u, \mathcal{U}; \tau - 1), \ldots, Q_n^{(v)}(u, \mathcal{U}; \tau - 1)$, j, v, and τ
Output: $\varphi^{(v)}(u, \mathcal{U}; \tau - 1)$
1 $\varphi^{(v)}(u, \mathcal{U}; \tau - 1) \leftarrow \emptyset$
2 **for** $i \in [\kappa]$ **do**
3 $\quad$ **for** $j' \in [n] \setminus \{j\}$ **do**
4 $\quad\quad$ **if** $u = a_{i,j'}$ **then**
5 $\quad\quad\quad \varphi^{(v)}(u, \mathcal{U}; \tau - 1) \leftarrow \texttt{Reproduce}(j, Q_{j'}^{(v)}(u, \mathcal{U}; \tau - 1))$
6 $\quad\quad\quad$ break
7 $\quad\quad$ **end**
8 $\quad$ **end**
9 **end**

Algorithm 5 `M-Sym`

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

6) Message and Index Symmetry in Rounds $\tau > 1$: In lines 33 to 37, the subroutine `M-Sym`, given in Algorithm 5, is invoked to generate the undesired query sets $Q_j^{(v)}(a_{i,j}, \mathcal{U}; \tau)$ by utilizing message symmetry. This subroutine selects symbols of undesired functions evaluations to generate τ-sums that enforce symmetry in the round queries. The procedure resembles the subroutine `M-Sym` proposed in [16]. In Algorithm 5, Π_τ denotes the set of all possible selections of τ distinct indices in $[\mu]$ and $\texttt{Lexico}(\Pi_\tau)$ 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 symbol can be arbitrary. This is the case since only the function

TABLE I

AUXILIARY QUERY SETS FOR EACH ROUND. HIGHLIGHTED IN RED IS THE ROW INDICATOR $u \in [\nu]$ USED IN DETERMINING THE INDICES OF THE QUERIED SYMBOLS. 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
$Q_j^{(1)}(a_{1,j},\mathcal{D};1)$	$x_{(1-1)\cdot 1+1,1}$	$x_{(2-1)\cdot 1+1,2}$	$x_{(1-1)\cdot 1+1,3}$	$x_{(2-1)\cdot 1+1,4}$
$Q_j^{(1)}(a_{1,j},\mathcal{U};1)$	$y_{(1-1)\cdot 1+1,1}$ $z_{(1-1)\cdot 1+1,1}$ $w_{(1-1)\cdot 1+1,1}$	$y_{(2-1)\cdot 1+1,2}$ $z_{(2-1)\cdot 1+1,2}$ $w_{(2-1)\cdot 1+1,2}$	$y_{(1-1)\cdot 1+1,3}$ $z_{(1-1)\cdot 1+1,3}$ $w_{(1-1)\cdot 1+1,3}$	$y_{(2-1)\cdot 1+1,4}$ $z_{(2-1)\cdot 1+1,4}$ $w_{(2-1)\cdot 1+1,4}$

(a)

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

(b)

j	1	2	3	4
$Q_j^{(1)}(a_{1,j},\mathcal{D};3)$	$x_{4\cdot 2+1,1}+y_{6,1}+z_{4,1}$ $x_{5\cdot 2+1,1}+y_{8,1}+w_{4,1}$ $x_{6\cdot 2+1,1}+z_{8,1}+w_{6,1}$	$x_{4\cdot 2+2,2}+y_{5,2}+z_{3,2}$ $x_{5\cdot 2+2,2}+y_{7,2}+w_{3,2}$ $x_{6\cdot 2+2,2}+z_{7,2}+w_{5,2}$	$x_{4\cdot 2+1,3}+y_{6,3}+z_{4,3}$ $x_{5\cdot 2+1,3}+y_{8,3}+w_{4,3}$ $x_{6\cdot 2+1,3}+z_{8,3}+w_{6,3}$	$x_{4\cdot 2+2,4}+y_{5,4}+z_{3,4}$ $x_{5\cdot 2+2,4}+y_{7,4}+w_{3,4}$ $x_{6\cdot 2+2,4}+z_{7,4}+w_{5,4}$
$Q_j^{(1)}(a_{1,j},\mathcal{U};3)$	$y_{12+1,1}+z_{10+1,1}+w_{8+1,1}$	$y_{12+2,2}+z_{10+2,2}+w_{8+2,2}$	$y_{12+1,3}+z_{10+1,3}+w_{8+1,3}$	$y_{12+2,4}+z_{10+2,4}+w_{8+2,4}$

(c)

j	1	2	3	4
$Q_j^{(1)}(a_{1,j},\mathcal{D};4)$	$x_{7\cdot 2+1,1}+y_{14,1}+z_{12,1}+w_{10,1}$	$x_{7\cdot 2+2,2}+y_{13,2}+z_{11,2}+w_{9,2}$	$x_{7\cdot 2+1,3}+y_{14,3}+z_{12,3}+w_{10,3}$	$x_{7\cdot 2+2,4}+y_{13,4}+z_{11,4}+w_{9,4}$

(d)

indices $(v_1,\ldots,v_\tau)$ are necessary to determine i_z, $\forall z \in [\tau]$. As a result, symmetry over the linear functions is maintained. Moreover, for $Q_j^{(v)}(a_{i,j},\mathcal{U};\tau)$, $i \in [\kappa]$, we obtain for each $\tau \in [2:\mu]$ the remaining τ-sum types, such that each of these query sets contains $\binom{\mu-1}{\tau}\kappa^{\mu-(\tau-1+1)}(\nu-\kappa)^{\tau-1}$ symbols.

Example 2 (Continued): After determining the indices of the desired function evaluations to be queried by each database in round $\tau = 2$, we now deploy side information to preserve the privacy for the desired function evaluation. This is accomplished by generating τ-sums of each possible *type* and enforcing index symmetry. To this end, we first identify the side information available from the previous round, queried from the neighboring databases, to be exploited according to the interference matrix $\mathsf{B}_{1\times 4}$. This process is performed by invoking Algorithm 4, which generates *complement* sets for the undesired query sets of the previous round, i.e., $Q_j^{(1)}(a_{i,j},\mathcal{U};1)$. The output of Algorithm 4 for $u \in [2]$ is as follows.

j	1	2	3	4
$Q_j^{(1)}(1,\mathcal{U};1)$		$y_{1,2}, z_{1,2}, w_{1,2}$		$y_{1,4}, z_{1,4}, w_{1,4}$
$Q_j^{(1)}(2,\mathcal{U};1)$	$y_{2,1}, z_{2,1}, w_{2,1}$		$y_{2,3}, z_{2,3}, w_{2,3}$	

Next, these side information query sets are then partitioned into κ groups to be exploited in different $Q_j^{(1)}(a_{i,j},\mathcal{D};2)$ for $i \in [\kappa]$. The partitioning guarantees that the two sets used in generating the τ-sums in lines 28 to 30 of Algorithm 1 have an equal number of elements. Finally, message and index symmetry is guaranteed by passing the generated auxiliary query sets $Q_j^{(1)}(a_{i,j},\mathcal{D};2)$ to the subroutine `M-Sym`, i.e., Algorithm 5, that generates τ-sums of the remaining *types*. Table I(b) illustrates the final query sets for round $\tau = 2$.

Next, Steps 4) to 6) are repeated for the following rounds, i.e., for $\tau = 3$ and $\tau = 4$. As a result, the queries for $\nu(\alpha_3-\alpha_2) = 6$ and the remaining $\nu(\alpha_4-\alpha_3) = 2$ distinct symbols of the desired linear function evaluation are generated by rounds $\tau = 3$ and $\tau = 4$, respectively. Tables I(c)-(d) illustrate the final query sets for the final rounds. Similar to Table I(a), in Tables I(b)–(d), we highlight with red the row indicator $u = a_{1,j} \in [\nu]$ and with magenta dashed arrows the side information exploitation following the algorithm `Exploit-SI`, i.e., Algorithm 4. In addition, we indicate with cyan arrows the message symmetry enforcement procedure following the algorithm `M-Sym`, i.e., Algorithm 5, and with red the resulting index symmetry in $Q_j^{(1)}(a_{1,j},\mathcal{U};\tau)$ based on the desired linear function indices. ▽

7) Query Set Assembly: Finally, in lines 39 to 48, we assemble each query set from disjoint query subsets obtained in all τ rounds. It can be shown 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 as follows. For the initialization round, $\tau = 1$, from Step 3) above, the total number of queried symbols is given by $\left|Q_j^{(v)}(\mathcal{D};1) \cup Q_j^{(v)}(\mathcal{U};1)\right| = \kappa\left[\kappa^{\mu-1} + \binom{\mu-1}{1}\kappa^{\mu-1}\right] = \binom{\mu}{1}\kappa^{\mu-1+1}(\nu-\kappa)^{1-1}$.

For the following rounds, $\tau \in [2:\mu]$, from Steps 4), 5), and 6) above, we have

$$\begin{aligned}&\left|Q_j^{(v)}(\mathcal{D};\tau) \cup Q_j^{(v)}(\mathcal{U};\tau)\right| \\ &= \kappa\left[\binom{\mu-1}{\tau-1}\kappa^{\mu-\tau}(\nu-\kappa)^{\tau-1} + \binom{\mu-1}{\tau}\kappa^{\mu-\tau}(\nu-\kappa)^{\tau-1}\right] \\ &= \binom{\mu}{\tau}\kappa^{\mu-\tau+1}(\nu-\kappa)^{\tau-1}.\end{aligned}$$

In summary, the total number of queries generated by Algorithm 1 is

$$\sum_{j=1}^{n}\left|Q_j^{(v)}\right| = n\sum_{\tau=1}^{\mu}\binom{\mu}{\tau}\kappa^{\mu-\tau+1}(\nu-\kappa)^{\tau-1}. \tag{10}$$

Remark 1: The practicality of implementing an algorithm is measured by the algorithm's computational complexity, i.e., the number of operations an algorithm performs to complete its task. The computational complexity of Algorithm 1 can be shown to be $\mathcal{O}(n\kappa\mu\nu^{\mu-1})$. To the best of our knowledge, our scheme shares this exponential time complexity with the PIR and PLC schemes of [10], [12], [15], [16].

Example 2 (Continued): In the final step, i.e., Step 7), the auxiliary query subsets are aggregated according to the row indicator $u = a_{i,j}$, $i \in [\kappa]$, to form the final query set for each database. Note that, by utilizing the code coordinates forming an information set in the code array, it can be shown that the side information based on $\mathsf{B}_{(\nu-\kappa)\times n}$ can be decoded. For example, in round 3, since $\{2,4\}$ is an information set of the storage code $\mathscr{C}$, the code symbols $y_{6,1}+z_{4,1}$ and $y_{6,3}+z_{4,3}$ can be obtained by knowing $y_{6,2}+z_{4,2}$ and $y_{6,4}+z_{4,4}$, from which the corresponding symbols $x_{6,1}$ and $x_{6,3}$ can be obtained by canceling the side information. Hence, the symbols from the desired linear function evaluation can be obtained. $\triangledown$

B. Recovery of Desired Function Evaluation

The construction of the capacity-achieving PLC scheme is, so far, a PIR-like scheme that privately retrieve a virtual message from a linearly-coded DSS. This virtual message represents the evaluation of the desired function over coded symbols, however, the user wishes to privately retrieve the evaluation of the desired function over the original information symbols. As a result, due to the fact that we are performing computation over coded storage, the coded PLC scheme includes two extra steps over other uncoded PC schemes. Namely, decoding the desired function evaluation symbols and decoding and canceling the side information. Thus, the correct decoding of the desired function evaluation relies on the correct decoding of the queried symbols from all virtual messages. To this end, in the following, we show that we can reliably recover the desired function evaluation from the queried symbols.

The main argument behind the reliable recovery of the desired function evaluation is the fact that the candidate linear functions and linear coding commute, i.e., evaluating a function over coded symbols is equal to encoding the symbols of the function evaluation. To see that, let $\hat{t} = \pi(t)$ where $t, \hat{t} \in [\beta]$ be the private permutation selected by the user and let $\boldsymbol{g}_j = \left(g_{1,j}, g_{2,j}, \ldots, g_{k,j}\right)^{\intercal}$ be the j-th column of the generator matrix $\mathsf{G}^{\mathscr{C}}$ for the $[n,k]$ linear storage code. One can verify, from (9), that for all $v' \in [\mu]$, we have

$$\begin{aligned}U_{t,j}^{(v')} &= \boldsymbol{v}_{v'}\boldsymbol{C}_{\hat{t},j} = \sum_{i=1}^{f} v_{v',i}C_{\hat{t},j}^{(i)} = \sum_{i=1}^{f} v_{v',i}\sum_{h=1}^{k} W_{\hat{t},h}^{(i)} g_{h,j} \\ &= \sum_{h=1}^{k} g_{h,j}\sum_{i=1}^{f} v_{v',i}W_{\hat{t},h}^{(i)} = \sum_{h=1}^{k} X_{\hat{t},h}^{(v')} g_{h,j},\end{aligned} \tag{11}$$

where $(X_{1,1}^{(v')}, \ldots, X_{1,k}^{(v')}, X_{2,1}^{(v')}, \ldots, X_{\beta,k}^{(v')}) = (X_1^{(v')}, \ldots, X_k^{(v')}, X_{k+1}^{(v')}, \ldots, X_{\mathrm{L}}^{(v')}) = \mathbf{X}^{(v')}$. Note that (11) resembles the process of encoding the segment $\left(X_{\hat{t},1}^{(v')}, \ldots, X_{\hat{t},k}^{(v')}\right)$ of the candidate linear function evaluation $\mathbf{X}^{(v')}$ using the $[n,k]$ storage code. Thus, one can consider the construction of our PLC scheme, so far, as a coded PIR scheme over a virtual coded DSS storing the evaluations of the candidate functions. As a result, using the same $[n,k]$ linear code for decoding the symbols obtained from the answer sets guarantees the reliable retrieval of the desired function evaluation.

C. Sign Assignment and Redundancy Elimination

In contrast to simple PIR solutions, in PLC we have the opportunity to exploit the dependencies induced by performing computations over the same set of messages, i.e., the f independent messages $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(f)}$, while keeping the requested index v private from each database. As shown in the recent PC literature (e.g., [16], [18], [21]), one is able to exploit this dependency to optimize the download cost by trading communication overhead with offline computation performed at the user side. To this end, our proposed PLC scheme is further constructed with two additional procedures: *Sign assignment* and *redundancy elimination*.

After running 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 evaluations is exploited. To this end, an initial sign $\sigma_t^{(v)}$ is first privately generated by the user with a uniform distribution over $\{-1,+1\}$ for all $t \in [\beta]$, i.e., the same selected sign is identically applied to all symbols from different function evaluations with the same index.

Next, depending on the desired linear function index $v \in [\mu]$, we apply a deterministic sign assignment procedure that carefully scales each pre-signed symbol in the query sets, i.e., $\sigma_t^{(v)}U_{t,j}^{(v')}$, $v' \in [\mu]$, by $\{+1,-1\}$. 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 evaluations and/or interference, thus reducing the download rate. On the other hand, the privately selected initial sign for $\sigma_t^{(v)}$, $t \in [\beta]$, acts as a one-time pad that randomizes over the deterministic sign assignment procedure. Here, we adopt a similar sign assignment process over each symbol in the query sets, as introduced in [16, Sec. IV-B].

The sign assigned to each symbol relies on two factors; the position of that symbol within a lexicographically ordered τ-sum query and whether that query contains a symbol from the desired function evaluation. Specifically, let a lexicographically ordered τ-sum query be q, i.e., $q \triangleq \sum_{\ell=1}^{\tau} U^{(v_\ell)}$, $v_1 < \cdots < v_\tau$.[2] Let $\Delta^{(v)}(q)$ denote the position of the symbol associated with the desired function evaluation $\mathbf{X}^{(v)}$ within q, where $\Delta^{(v)}(q) = 0$ indicates that the query does not contain a symbol from the desired function evaluation. The queries generated by Algorithm 1 are sorted by round $\tau \in [\mu]$, then the queries for each round are divided into subgroups indexed by $S(\Delta^{(v)}(q)) \in \{1, 2, \ldots\}$ based on the value of $\Delta^{(v)}(q)$ for each query. Finally, a '$+$' or '$-$' sign is assigned as a function of the subgroup index $S(\cdot)$ and the position of each symbol relative to the desired function evaluation symbol in each query. The details of the sign selection follow [16, Sec. IV-B] and are omitted for brevity. Moreover, we remark that after sign assignment, the recovery condition of the scheme is inherently maintained since it can be seen as a coded PIR scheme as Protocol 1 in [15]. The key idea of redundancy elimination is illustrated with Example 2 below.

Example 2 (Continued): First, without loss of generality, we assume the initial sign assignment $\sigma_t^{(v)} = +1$ is privately selected by the user for all $t \in [\beta]$. Next, we apply the sign assignment process to the query sets for $v = 1$. The resulting queries after sign assignment are shown in Table II. In the following, we show that we can remove some redundant queries from each database and the desired linear function evaluation $\mathbf{X}^{(1)}$ can still be recovered. For example, in the first round ($\tau = 1$), it can be easily seen from $\mathsf{V}_{\mu\times f}$ that the queried symbols of $z_{t,j}$ and $w_{t,j}$ can be generated offline by the user as functions of $x_{t,j}$ and $y_{t,j}$, i.e., $z_{t,j} = x_{t,j} + y_{t,j}$ and $w_{t,j} = 3x_{t,j} + y_{t,j}$ for all $t \in [\beta]$ and $j \in [n]$. Moreover, the coefficient vectors associated with $x_{t,j}$ and $y_{t,j}$ are the two row basis vectors of the coefficient matrix $\mathsf{V}_{\mu\times f}$ ($r = \operatorname{rank}(\mathsf{V}) = 2$). Thus, we can represent the candidate functions evaluations in terms of this basis with a deterministic linear mapping $\hat{\mathsf{V}}_{\mu\times r} = (\hat{v}_{i,l})$ of size $\mu \times r$ as follows:

$$(x_{t,j}, y_{t,j}, z_{t,j}, w_{t,j})^{\intercal} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 3 & 1 \end{pmatrix}}_{\hat{\mathsf{V}}_{\mu\times r}} (x_{t,j}, y_{t,j})^{\intercal}. \tag{12}$$

That is true due to the commutativity of the performed linear functions, i.e., the storage code and the candidate functions, and given that the coefficient matrix $\mathsf{V}_{\mu\times f}$ of the candidate functions is available to the user. Thus, the queries for these symbols, i.e., $z_{t,j}$ and $w_{t,j}$, are redundant and can be removed from the query sets regardless of which function evaluation is desired by the user. Next, in round $\tau = 2$ and for the 1st database, from the deterministic linear mapping $\hat{\mathsf{V}}_{\mu\times r} = (\hat{v}_{i,l})$ of (12), one can verify that

$$\begin{aligned} &\hat{v}_{3,2}(y_{7,1} - w_{3,1}) - \hat{v}_{4,2}(y_{5,1} - z_{3,1}) \\ &\quad - (\hat{v}_{3,1} \cdot \hat{v}_{4,2} - \hat{v}_{4,1} \cdot \hat{v}_{3,2})x_{3,1} - \hat{v}_{4,1}x_{5,1} + \hat{v}_{3,1}x_{7,1} \\ &= 1(y_{7,1} - w_{3,1}) - 1(y_{5,1} - z_{3,1}) - (1 \cdot 1 - 3 \cdot 1)x_{3,1} \\ &\quad -3x_{5,1} + 1x_{7,1} \\ &= 1(y_{7,1} - 3x_{3,1} - 1y_{3,1}) - (y_{5,1} - x_{3,1} - y_{3,1}) \\ &\quad +2x_{3,1} - 3x_{5,1} + x_{7,1} \\ &= (x_{7,1} + y_{7,1}) - (3x_{5,1} + y_{5,1}) = z_{7,1} - w_{5,1}, \end{aligned} \tag{13}$$

and hence we do not need to download the 2-sum $z_{7,1} - w_{5,1}$. Similarly, we can do the same exercise for the other databases. The redundant queries are marked in blue in Table II, shown at the top of the following page, and the indices $t \in [\beta]$ of the desired linear function evaluations are marked in red. This completes the recovery part. The resulting PLC rate becomes $\frac{\nu^{\mu}\cdot k}{\mathsf{D}} = \frac{16\cdot 2}{12\cdot 4} = \frac{2}{3}$, which is equal to the PLC capacity in Theorem 3 with $r = \operatorname{rank}(\mathsf{V}) = 2$. This demonstrates the optimality of the PLC scheme. ▽

From the above example we note the following.

- There is a deterministic linear mapping, i.e., $\hat{\mathsf{V}}_{\mu\times r}$, that captures the dependencies among the candidate linear functions evaluations.
- We maintain the same characteristics of the query construction that facilitate the exploitation of the linear dependencies among the candidate functions evaluations as for the uncoded PLC scheme in [16]. These characteristics include index assignment, sign assignment, and lexicographic ordering of the elements of τ-sums. As a result, some of the queries become redundant and can be removed from the query sets while maintaining the decodability of the desired function evaluation.
- The candidate functions are computed over the coded symbols stored in each database individually. Consequently, from the perspective of the queries of each database, the linear dependency among the symbols of the candidate functions evaluations is present, i.e., the fact that the computation is performed over coded storage is transparent to the redundancy elimination process. This can be seen from (13).
- The number of redundant queries depends on the rank of the coefficient matrix $\mathsf{V}_{\mu\times f}$, i.e., $r = \operatorname{rank}(\mathsf{V})$. This can be clearly observed for the 1-sum symbols where out of the μ symbols, $\mu - r$ can be computed offline given that the symbols of the functions evaluations associated with the r row basis vectors of $\mathsf{V}_{\mu\times f}$ are available.

Based on this insight we can state the following lemma for redundancy elimination.

Lemma 4: For all $v \in [\mu]$, each database $j \in [n]$, and based on the side information available from the databases, any $\binom{\mu-r}{\tau}$ τ-sum types out of all possible $\binom{\mu}{\tau}$ types in each round $\tau \in [\mu - r]$ of the query sets are redundant.

The proof of Lemma 4 is presented in Appendix C. The proof is based on the insight that the redundancy resulting from the linear dependencies between virtual messages is also present with MDS-PIR capacity-achieving codes. Since both repetition and MDS codes are MDS-PIR capacity-achieving codes, Lemma 4 generalizes both [16, Lem. 1] and [18, Lem. 1]. We now make the final modification to our PLC query sets by first directly applying the sign assignment over $\sigma_t^{(v)} U_{t,j}^{(v')}$, $v' \in [\mu]$, and then remove the τ-sums corresponding

[2]Segment and database indices are suppressed here for clarity.

TABLE II

PLC QUERY SETS FOR $v = 1$ AFTER SIGN ASSIGNMENT FOR ROUNDS ONE TO FOUR FOR THE $[4, 2]$ CODE OF EXAMPLE 2, $f = 4$ MESSAGES, AND $\mu = 4$ CANDIDATE LINEAR FUNCTIONS. RED SUBSCRIPTS INDICATE THE INDICES OF THE DESIRED LINEAR FUNCTION EVALUATIONS. THE REDUNDANT QUERIES ARE MARKED IN BLUE

j	1	2	3	4
$Q_j^{(v)}(\mathcal{D};1)$	$x_{1,1}$	$x_{2,2}$	$x_{1,3}$	$x_{2,4}$
$Q_j^{(v)}(\mathcal{U};1)$	$y_{1,1}, z_{1,1}, w_{1,1}$	$y_{2,2}, z_{2,2}, w_{2,2}$	$y_{1,3}, z_{1,3}, w_{1,3}$	$y_{2,4}, z_{2,4}, w_{2,4}$
$Q_j^{(v)}(\mathcal{D};2)$	$x_{3,1} - y_{2,1}$ $x_{5,1} - z_{2,1}$ $x_{7,1} - w_{2,1}$	$x_{4,2} - y_{1,2}$ $x_{6,2} - z_{1,2}$ $x_{8,2} - w_{1,2}$	$x_{3,3} - y_{2,3}$ $x_{5,3} - z_{2,3}$ $x_{7,3} - w_{2,3}$	$x_{4,4} - y_{1,4}$ $x_{6,4} - z_{1,4}$ $x_{8,4} - w_{1,4}$
$Q_j^{(v)}(\mathcal{U};2)$	$y_{5,1} - z_{3,1}$ $y_{7,1} - w_{3,1}$ $z_{7,1} - w_{5,1}$	$y_{6,2} - z_{4,2}$ $y_{8,2} - w_{4,2}$ $z_{8,2} - w_{6,2}$	$y_{5,3} - z_{3,3}$ $y_{7,3} - w_{3,3}$ $z_{7,3} - w_{5,3}$	$y_{6,4} - z_{4,4}$ $y_{8,4} - w_{4,4}$ $z_{8,4} - w_{6,4}$
$Q_j^{(v)}(\mathcal{D};3)$	$x_{9,1} - y_{6,1} + z_{4,1}$ $x_{11,1} - y_{8,1} + w_{4,1}$ $x_{13,1} - z_{8,1} + w_{6,1}$	$x_{10,2} - y_{5,2} + z_{3,2}$ $x_{12,2} - y_{7,2} + w_{3,2}$ $x_{14,2} - z_{7,2} + w_{5,2}$	$x_{9,3} - y_{6,3} + z_{4,3}$ $x_{11,3} - y_{8,3} + w_{4,3}$ $x_{13,3} - z_{8,3} + w_{6,3}$	$x_{10,4} - y_{5,4} + z_{3,4}$ $x_{12,4} - y_{7,4} + w_{3,4}$ $x_{14,4} - z_{7,4} + w_{5,4}$
$Q_j^{(v)}(\mathcal{U};3)$	$y_{13,1} - z_{11,1} + w_{9,1}$	$y_{14,2} - z_{12,2} + w_{10,2}$	$y_{13,3} - z_{11,3} + w_{9,3}$	$y_{14,4} - z_{12,4} + w_{10,4}$
$Q_j^{(v)}(\mathcal{D};4)$	$x_{15,1} - y_{14,1} + z_{12,1} - w_{10,1}$	$x_{16,2} - y_{13,2} + z_{11,2} - w_{9,2}$	$x_{15,3} - y_{14,3} + z_{12,3} - w_{10,3}$	$x_{16,4} - y_{13,4} + z_{11,4} - w_{9,4}$

to the redundant τ-sum types from every round $\tau \in [\mu - r]$. Note that the amount of redundancy is dependent on the rank of the functions matrix, $\text{rank}\,(\mathsf{V}) = r \leq \min\{\mu, f\}$, thus generalizing the MDS-coded PLC case. Finally, we generate the queries $Q_{[n]}^{(v)}$.

D. Privacy

It is worth mentioning that the queries generated by Algorithm 1 inherently satisfy the privacy condition of (2a), which is guaranteed by satisfying the index, message, and database symmetry principles as for all the PIR schemes in [10], [12], [15]. That is, given the fixed and symmetric construction of the queries, there always exists a one-to-one mapping between the queries, $Q_j^{(v)} \leftrightarrow Q_j^{(v')}$, $\forall\, j \in [n]$, in terms of the queried symbols indices $t \in [\beta]$, where $v, v' \in [\mu]$ and $v \neq v'$. Given this one-to-one mapping along with a permutation $\pi(t)$ over these indices privately selected uniformly at random by the user, the queries are indistinguishable and equally likely.

Moreover, after the sign assignment process a one-to-one mapping between the assigned signs is found following a simple sign flipping rule for $\sigma_t^{(v')}$. The rule states that, to map the queries of $Q_j^{(v)}$ to $Q_j^{(v')}$, one should only consider the desired queries, i.e., queries that contain symbols associated with $\mathbf{X}^{(v')}$. For such queries in each round τ, we replace $\sigma_*^{(v')}$ with $-\sigma_*^{(v')}$ for each element to the right of the desired function evaluation symbol $U_*^{(v')}$ in the lexicographically ordered query if the query is sorted in a subgroup indexed with an odd S (see Section IV-C). Next, we flip the sign of elements to the left of the desired function evaluation symbol $U_*^{(v')}$ if the query is sorted in a subgroup indexed with an even S. The proof of the correctness of this rule and thus the privacy after sign assignment follows directly from [16, Sec. VI-B]. For completeness, we also show with Example 2 that the user's privacy is still maintained after the sign assignment process and the removal of redundant queries.

Example 2 (Continued): Here, to show that the queries are identically distributed regardless of the desired function evaluation index $v \in [4]$ we show that there exists a one-to-one mapping from the queries for $v = 1$ to the queries for $v = 3$ for all databases. Without loss of generality, we again assume the initial sign assignment $\sigma_t^{(3)} = +1$ to be privately selected by the user for all $t \in [\beta]$. In Table III, shown at the top of the following page, the queries for $v = 3$ are presented following Algorithm 1 and the sign assignment process. From Tables II and III one can verify that the index and sign mapping

Databases 1 and 3:

$$(3, 2, 5, 9, 6, 4, 11, 8, 13, \sigma_{13}^{(1)}, 15, 14, 12, \sigma_{10}^{(1)}) \xrightarrow{v=3} (5, 3, 2, 6, 4, 9, 13, 11, 8, -\sigma_8^{(3)}, 14, 12, 15, -\sigma_{10}^{(3)}) \tag{14a}$$

Databases 2 and 4:

$$(4, 1, 6, 10, 5, 3, 12, 7, 14, \sigma_{14}^{(1)}, 16, 13, 11, \sigma_9^{(1)}) \xrightarrow{v=3} (6, 4, 1, 5, 3, 10, 14, 12, 7, -\sigma_7^{(3)}, 13, 11, 16, -\sigma_9^{(3)}) \tag{14b}$$

converts the queries for $v = 1$ to the queries for $v = 3$. To see this mapping, compare the τ-sums $x_{t_1,1} - y_{t_2,1}$ and $x_{t'_1,1} - y_{t'_2,1}$ from the queries of the first database of Tables II and III, respectively. It can be seen that the indices $t_1 = 3$ and $t_2 = 2$ of the queries for $v = 1$ convert into the indices $t'_1 = 5$ and $t'_2 = 3$ of the queries for $v = 3$, respectively. Thus, we have the mapping $((t_1, t_2) \to (t'_1, t'_2)) = ((3, 2) \to (5, 3))$ and due to the index symmetry of the query construction this mapping is fixed for all symbols with the corresponding indices. A similar comparison between the remaining τ-sums results in the index and sign mapping of (14a) and (14b). One can similarly verify that there exists a mapping from the queries for $v = 1$ to the queries for $v = 2$ or those for $v = 4$, i.e., $Q_{[n]}^{(1)} \leftrightarrow Q_{[n]}^{(2)}$ and $Q_{[n]}^{(1)} \leftrightarrow Q_{[n]}^{(4)}$. Since a permutation over these indices, i.e., $\pi(t)$ and an initial sign $\sigma_t^{(v)}$ are uniformly and privately selected by the user independently of the desired function evaluation index v, these queries are equally likely and indistinguishable.

Next, to verify the correctness of the sign flipping rule stated above, consider the desired queries of the third round ($\tau = 3$)

TABLE III

PLC QUERY SETS FOR $v = 3$ AFTER SIGN ASSIGNMENT FOR ROUNDS ONE TO FOUR FOR THE $[4, 2]$ CODE OF EXAMPLE 2, $f = 4$ MESSAGES, AND $\mu = 4$ CANDIDATE LINEAR FUNCTIONS. RED SUBSCRIPTS INDICATE THE INDICES OF THE DESIRED LINEAR FUNCTION EVALUATIONS. THE REDUNDANT QUERIES ARE MARKED IN BLUE

j	1	2	3	4
$Q_j^{(v)}(\mathcal{D};1)$	$z_{1,1}$	$z_{2,2}$	$z_{1,3}$	$z_{2,4}$
$Q_j^{(v)}(\mathcal{U};1)$	$x_{1,1}, y_{1,1}, w_{1,1}$	$x_{2,2}, y_{2,2}, w_{2,2}$	$x_{1,3}, y_{1,3}, w_{1,3}$	$x_{2,4}, y_{2,4}, w_{2,4}$
$Q_j^{(v)}(\mathcal{D};2)$	$x_{2,1} - z_{3,1}$ $y_{2,1} - z_{5,1}$ $z_{7,1} - w_{2,1}$	$x_{1,2} - z_{4,2}$ $y_{1,2} - z_{6,2}$ $z_{8,2} - w_{1,2}$	$x_{2,3} - z_{3,3}$ $y_{2,3} - z_{5,3}$ $z_{7,3} - w_{2,3}$	$x_{1,4} - z_{4,4}$ $y_{1,4} - z_{6,4}$ $z_{8,4} - w_{1,4}$
$Q_j^{(v)}(\mathcal{U};2)$	$x_{5,1} - y_{3,1}$ $x_{7,1} - w_{3,1}$ $y_{7,1} - w_{5,1}$	$x_{6,2} - y_{4,2}$ $x_{8,2} - w_{4,2}$ $y_{8,2} - w_{6,2}$	$x_{5,3} - y_{3,3}$ $x_{7,3} - w_{3,3}$ $y_{7,3} - w_{5,3}$	$x_{6,4} - y_{4,4}$ $x_{8,4} - w_{4,4}$ $y_{8,4} - w_{6,4}$
$Q_j^{(v)}(\mathcal{D};3)$	$x_{6,1} - y_{4,1} + z_{9,1}$ $-x_{8,1} - z_{11,1} + w_{4,1}$ $-y_{8,1} - z_{13,1} + w_{6,1}$	$x_{5,2} - y_{3,2} + z_{10,2}$ $-x_{7,2} - z_{12,2} + w_{3,2}$ $-y_{7,2} - z_{14,2} + w_{5,2}$	$x_{6,3} - y_{4,3} + z_{9,3}$ $-x_{8,3} - z_{11,3} + w_{4,3}$ $-y_{8,3} - z_{13,3} + w_{6,3}$	$x_{5,4} - y_{3,4} + z_{10,4}$ $-x_{7,4} - z_{12,4} + w_{3,4}$ $-y_{7,4} - z_{14,4} + w_{5,4}$
$Q_j^{(v)}(\mathcal{U};3)$	$x_{13,1} - y_{11,1} + w_{9,1}$	$x_{14,2} - y_{12,2} + w_{10,2}$	$x_{13,3} - y_{11,3} + w_{9,3}$	$x_{14,4} - y_{12,4} + w_{10,4}$
$Q_j^{(v)}(\mathcal{D};4)$	$x_{14,1} - y_{12,1} + z_{15,1} + w_{10,1}$	$x_{13,2} - y_{11,2} + z_{16,2} + w_{9,2}$	$x_{14,3} - y_{12,3} + z_{15,3} + w_{10,3}$	$x_{13,4} - y_{11,4} + z_{16,4} + w_{9,4}$

for the query sets for $v = 3$ in Table III. For database 1, one can verify that the query $x_{6,1} - y_{4,1} + z_{9,1}$ is sorted in the subgroup indexed by $S = 1$. As S is odd and no element is placed to the right of $z_{9,1}$ the signs are left unchanged. However, for the query $-x_{8,1} - z_{11,1} + w_{4,1}$ which falls in the subgroup indexed by $S = 2$, the sign of the element to the left of $z_{11,1}$, i.e., $x_{8,1}$, is flipped. That is, we change $\sigma_8^{(3)}$ to $-\sigma_8^{(3)}$ and that matches the sign mapping in (14a) for this query. Moreover, due to index symmetry, this mapping also matches the sign assignment for $\sigma_8^{(3)}$ for the query $-y_{8,1} - z_{13,1} + w_{6,1}$.

Finally, for redundancy elimination, we only need to show that for any desired index $v \in [4]$, the removed redundant τ-sums can be chosen to be of the same type. For instance, let us consider the 1st database. In the 2nd round, see Table III, it can be shown that the queries for desired index $v = 3$ satisfy the equation

$$\begin{aligned}&(1 \cdot 1 - 3 \cdot 1)(x_{5,1} - y_{3,1}) - 1(x_{7,1} - w_{3,1}) - 3z_{3,1} - 1z_{5,1} \\ &\qquad +1z_{7,1} \\ &= -2(x_{5,1} - y_{3,1}) - (x_{7,1} - 3x_{3,1} - y_{3,1}) - 3(x_{3,1} + y_{3,1}) \\ &\qquad -(x_{5,1} + y_{5,1}) + (x_{7,1} + y_{7,1}) \\ &= 1(y_{7,1} - (3x_{5,1} + y_{5,1})) = y_{7,1} - w_{5,1},\end{aligned}$$

which implies that the 2-sum $z_{7,1} - w_{2,1}$ can be removed from the download, since $z_{7,1}$ can be obtained from downloading $x_{5,1} - y_{3,1}$, $x_{7,1} - w_{3,1}$, $x_{2,1} - z_{3,1}$, $y_{2,1} - z_{5,1}$, and $y_{7,1} - w_{5,1}$. Hence, the redundant τ-sum type for $v = 3$ can be chosen to be equal to the redundant τ-sum type for $v = 1$ (see (13)). A similar argument can be made for $v = 2$ and $v = 4$, which ensures that the privacy of the scheme is not affected by redundancy elimination. $\triangledown$

E. Achievable PLC Rate

The resulting achievable PLC rate of Algorithm 1 after removing redundant τ-sums according to Lemma 4 becomes

$$\begin{aligned}\mathrm{R} &\overset{(a)}{=} \frac{k\nu^\mu}{n\sum_{\tau=1}^{\mu}\left(\binom{\mu}{\tau} - \binom{\mu-r}{\tau}\right)\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}} \\ &\overset{(b)}{=} \frac{\kappa\nu^\mu}{\nu\sum_{\tau=1}^{\mu}\left(\binom{\mu}{\tau} - \binom{\mu-r}{\tau}\right)\kappa^{\mu-(\tau-1)}(\nu-\kappa)^{\tau-1}} \\ &= \frac{\nu^\mu\left(\frac{\nu-\kappa}{\nu}\right)}{\sum_{\tau=1}^{\mu}\left(\binom{\mu}{\tau} - \binom{\mu-r}{\tau}\right)\kappa^{\mu-\tau}(\nu-\kappa)^{\tau}} \\ &\quad\vdots \\ &\overset{(c)}{=} \frac{\nu^\mu\left(1-\frac{\kappa}{\nu}\right)}{\nu^\mu - \kappa^r\nu^{\mu-r}} = \left(1-\frac{\kappa}{\nu}\right)\left[1-\left(\frac{\kappa}{\nu}\right)^r\right]^{-1}, \qquad (15)\end{aligned}$$

where we recall that $\binom{m}{n} = 0$ if $m < n$; (a) follows from the PLC rate in Definition 1, (10), and Lemma 4; (b) follows from Definition 3; and (c) follows by adapting similar steps as in the proof given in [18]. Note that the rate in (15) matches the converse in Theorem 2, which proves Theorem 3.

V. CONCLUSION

We have provided the capacity of PLC from coded DSSs, where data is encoded and stored using an arbitrary linear code from a large class of linear storage codes. Interestingly, for the considered family of linear storage codes, the capacity of PLC 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 and provides a significant advantage over individually downloading each message via a PIR scheme and combining them offline.

APPENDIX A
PROOF OF LEMMA 1

The proof of Lemma 1 uses the linear independence of the columns of a generator matrix of $\mathscr{C}$ corresponding to an information set. Consider an information set $\mathcal{I}$ of the $[n, k]$ linear storage code $\mathscr{C}$, $|\mathcal{I}| = k$. The content of the databases indexed by $\mathcal{I}$, i.e., $\left(\boldsymbol{C}_j,\, j \in \mathcal{I}\right) = \mathsf{C}|_{\mathcal{I}}$, can be written as $\left((\mathsf{W}^{(1)})^\intercal | \ldots | (\mathsf{W}^{(f)})^\intercal\right)^\intercal \mathsf{G}^{\mathscr{C}}|_{\mathcal{I}} \overset{(a)}{\sim} \left((\mathsf{W}^{(1)})^\intercal | \ldots | (\mathsf{W}^{(f)})^\intercal\right)^\intercal$, where by the construction of any $[n, k]$ linear storage code, if $\mathcal{I}$ is an information set of the code $\mathscr{C}$, then $\mathsf{G}^{\mathscr{C}}|_{\mathcal{I}}$ is

a $k \times k$ invertible matrix. (a) follows from [11, Lem. 1] and the fact that the messages are chosen independently and uniformly at random from $\mathbb{F}_p^{\beta \times k}$. Therefore, the content of any k databases forming an information set is statistically equivalent to the stored messages. Given that the symbols of the messages are independent, then the k columns of $\mathbf{C}|_{\mathcal{I}}$ are also statistically independent. Finally, since $A_j^{(v)}$, $j \in \mathcal{I}$, are deterministic functions (that are composed of the μ candidate linear combinations) of independent random variables $\{\boldsymbol{C}_j \colon j \in \mathcal{I}\}$ and $\mathcal{Q}$, $\{A_j^{(v)},\ j \in \mathcal{I}\}$ are statistically independent, and (4) follows.

Now, given that the candidate linear functions are evaluated element-wise over independent and uniformly distributed symbols, the symbols of each linear combination are also independent and identically distributed (i.i.d.), i.e., for $\mathbf{X}^{(v)} = \big(X_1^{(v)}, \ldots, X_{\mathsf{L}}^{(v)}\big)$, $X_1^{(v)}, \ldots, X_{\mathsf{L}}^{(v)}$ are i.i.d. according to a prototype random variable $X^{(v)}$.

Moreover, due to the commutative property of linear functions, linear computation over linearly-encoded symbols is equivalent to linear encoding of the linear function evaluations. As a result, we can extend the argument of statistical equivalence to linear function evaluations over coded symbols. In other words, the evaluations of linear functions over the content of any k databases, forming an information set $\mathcal{I}$, are statistically equivalent to the evaluations of linear functions over the stored messages. Presenting $\mathbf{X}^{(v)} = \big(X_1^{(v)}, \ldots, X_{\mathsf{L}}^{(v)}\big)$ in the form $\mathbf{X}^{(v)} = \big(X_{i,j}^{(v)}\big)$, $i \in [\beta]$, $j \in [k]$, we have $\big((\mathbf{X}^{(1)})^{\intercal} | \ldots | (\mathbf{X}^{(\mu)})^{\intercal}\big)^{\intercal} \mathsf{G}^{\mathscr{C}}|_{\mathcal{I}} \sim \big((\mathbf{X}^{(1)})^{\intercal} | \ldots | (\mathbf{X}^{(\mu)})^{\intercal}\big)^{\intercal}$. Finally, since we can consider that the storage encoding is applied on individual function evaluations, conditioning on any subset of function evaluations $\mathbf{X}^{\mathcal{V}}$, $|\mathcal{V}| = \mu_v$, is equivalent to reducing the problem to the private computation of $\mu - \mu_v$ linear combinations. That is, $A_j^{(v)}$, $j \in \mathcal{I}$, are deterministic functions (that are composed of the $\mu - \mu_v$ candidate linear combinations) of independent random variables $\{\boldsymbol{C}_j \colon j \in \mathcal{I}\}$ and $\mathcal{Q}$, and $\{A_j^{(v)},\ j \in \mathcal{I}\}$ are still statistically independent. Hence, the statistical independence argument of (5) follows.

Appendix B
Proof of Lemma 3

Since each linear function $\mathbf{X}^{(v)} = \big(X_1^{(v)}, \ldots, X_{\mathsf{L}}^{(v)}\big)$, $v \in [\mu]$, contains L i.i.d. symbols, it is clear that $\forall\, l \in [\mathsf{L}]$,

$$\mathsf{H}\big(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\mu)}\big) = \mathsf{L}\,\mathsf{H}\big(X_l^{(1)}, \ldots, X_l^{(\mu)}\big), \quad \text{and}$$
$$\mathsf{H}\big(\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(f)}\big) = \mathsf{L}\,\mathsf{H}\big(W_l^{(1)}, \ldots, W_l^{(f)}\big).$$

Let $\mathcal{J} \triangleq \{j_1, \ldots, j_h\}$ for some $h \in [r]$. We have

$$\begin{aligned}
&\Pr\left[X_l^{(i_1)}, \ldots, X_l^{(i_h)}\right] \\
&= \sum_{w_l^{\mathcal{J}^{\mathrm{c}}}} \Pr\left[W_l^{\mathcal{J}^{\mathrm{c}}} = w_l^{\mathcal{J}^{\mathrm{c}}}\right] \cdot \Pr\left[X_l^{(i_1)}, \ldots, X_l^{(i_h)} \middle| W_l^{\mathcal{J}^{\mathrm{c}}} = w_l^{\mathcal{J}^{\mathrm{c}}}\right] \\
&= \sum_{w_l^{\mathcal{J}^{\mathrm{c}}}} \Pr\left[W_l^{\mathcal{J}^{\mathrm{c}}} = w_l^{\mathcal{J}^{\mathrm{c}}}\right] \cdot \Pr\left[W_l^{(j_1)}, \ldots, W_l^{(j_h)} \middle| W_l^{\mathcal{J}^{\mathrm{c}}} = w_l^{\mathcal{J}^{\mathrm{c}}}\right] \qquad (16) \\
&= \sum_{w^{\mathcal{H}^{\mathrm{c}}}} \Pr\left[W_l^{\mathcal{J}^{\mathrm{c}}} = w_l^{\mathcal{J}^{\mathrm{c}}}\right] \left(\frac{1}{p}\right)^h = \left(\frac{1}{p}\right)^h, \qquad (17)
\end{aligned}$$

where (16) follows from the fact that there is a linear transformation between $X_l^{(i_1)}, \ldots, X_l^{(i_h)}$ and $W_l^{(j_1)}, \ldots, W_l^{(j_h)}$, and (17) holds since $W_l^{(j_1)}, \ldots, W_l^{(j_h)}$ are i.i.d. over $\mathbb{F}_p$. Hence, $\mathsf{H}\big(X_l^{(i_1)}, \ldots, X_l^{(i_h)}\big) = h$ (in p-ary units), which completes the proof.

Appendix C
Proof of Lemma 4

Here we present the main components needed for the proof of Lemma 4, however the detailed derivations are a direct application of the proof of [16, Lem. 1, Sec. V-B] and thus are omitted. The proof of [16, Lem. 1] is adapted to our setup with the following substitutions.

Let $\mathcal{L} \triangleq \{\ell_1, \ldots, \ell_r\} \subseteq [\mu]$ be the set of candidate linear combination indices corresponding to a basis of the row space of the linear combination coefficient matrix $\mathsf{V}_{\mu \times f}$, where $r = \operatorname{rank}(\mathsf{V}) \leq \min\{\mu, f\}$. Then $X_l^{(\ell_1)}, \ldots, X_l^{(\ell_r)}$ satisfy $\mathsf{H}\big(X_l^{(\ell_1)}, \ldots, X_l^{(\ell_r)}\big) = \mathsf{H}\big(X_l^{[\mu]}\big)$, $\forall\, l \in [\mathsf{L}]$. Assume, without loss of generality, that the rows of the coefficient matrix are ordered such that the first r rows constitute the row basis, i.e., $(X_l^{(1)}, \ldots, X_l^{(r)}) = (X_l^{(\ell_1)}, \ldots, X_l^{(\ell_r)})$, $l \in [\mathsf{L}]$. Note that we can represent the candidate functions evaluations $\big(X_l^{(1)}, \ldots, X_l^{(\mu)}\big)$ in terms of the basis candidate functions evaluations $\big(X_l^{(\ell_1)}, \ldots, X_l^{(\ell_r)}\big)$ for $l \in [\mathsf{L}]$ with a deterministic linear mapping $\hat{\mathsf{V}}_{\mu \times r}$ of size $\mu \times r$ as $\big(X_l^{(1)}, \ldots, X_l^{(\mu)}\big)^{\intercal} = \hat{\mathsf{V}}_{\mu \times r}\big(X_l^{(\ell_1)}, \ldots, X_l^{(\ell_r)}\big)^{\intercal}$. As a result, we have $(\hat{\boldsymbol{v}}_1^{\intercal}, \ldots, \hat{\boldsymbol{v}}_r^{\intercal})^{\intercal} = \mathsf{I}_r$, where I_r is the $r \times r$ identity matrix and $\hat{\boldsymbol{v}}_i$ is the i-th row vector of the deterministic linear mapping matrix $\hat{\mathsf{V}}_{\mu \times r}$.

First, consider the case where the desired function evaluation index $v = 1$. Consider the queries corresponding to undesired τ-sums, i.e., τ-sums that do not involve any symbols from the desired function evaluation $\mathbf{U}^{(1)}$. There are $\binom{\mu-1}{\tau}$ different τ-sum types corresponding to such queries which can be divided into two groups as follows.

- Group 1: $\binom{\mu-1}{\tau} - \binom{\mu-r}{\tau}$ τ-sum types for which the corresponding τ-sums involve at least one element from the set $\{\mathbf{U}^{(2)}, \mathbf{U}^{(3)}, \ldots, \mathbf{U}^{(r)}\}$.
- Group 2: $\binom{\mu-r}{\tau}$ τ-sum types for which the corresponding τ-sums do not involve any element from the set $\{\mathbf{U}^{(2)}, \mathbf{U}^{(3)}, \ldots, \mathbf{U}^{(r)}\}$.

Let $q(U^{(v_{[\tau]})})$ denote a τ-sum as defined in Definition 6 after performing the sign assignment process, i.e., $q(U^{(v_{[\tau]})}) \triangleq \sum_{\ell=1}^{\tau} (-1)^{\ell-1} U^{(v_\ell)}$, where $v_{[\tau]} = \{v_1, \ldots, v_\tau\} \subseteq [\mu]$, $v_1 < \cdots < v_\tau$, are the indices of the functions evaluations, and where the segment indices and the database index are suppressed to simplify the notation. Let the *type* of the τ-sum be presented by the set of distinct indices of functions evaluations involved in the τ-sum, i.e., the type of $q(U^{(v_{[\tau]})})$ is represented by $v_{[\tau]} = \{v_1, \ldots, v_\tau\}$. The key idea is to show that the symbols of the queries corresponding to Group 2 are deterministic linear functions of the queries corresponding to Group 1 when the symbols of the desired function evaluation $\mathbf{U}^{(1)}$ are known. Now, let $q_0 \triangleq q(U^{(v_{[\tau]})})$, where $r < v_1 < \cdots < v_\tau$, denote an arbitrary query corresponding to Group 2. Specifically, we need to show that, when the

symbols of $\mathbf{U}^{(1)}$ queried by the given database are known, i.e., successfully decoded, the query q_0 can be written as a linear function of $\binom{\tau+r-1}{\tau} - 1$ queries corresponding to Group 1. These $\binom{\tau+r-1}{\tau} - 1$ queries contain elements of the row basis functions evaluations and elements included in the τ-sum of q_0 and comprise all the τ-sums of types corresponding to the subsets of size τ of $\mathcal{I} \triangleq [2:r] \cup v_{[\tau]}$, except the type of q_0, i.e., $\{v_1, \ldots, v_\tau\}$. Now, let $\tilde{\mathcal{Q}} \triangleq \left\{q(U^{(\hat{i}_{[\tau]})}) : \hat{i}_{[\tau]} \in \mathcal{T}\right\}$ be a set of queries where there is exactly one query corresponding to each of the $\binom{\tau+r-1}{\tau} - 1$ τ-sum types of Group 1, where $\mathcal{T} \triangleq \left\{\hat{i}_{[\tau]} = \{\hat{i}_1, \hat{i}_2, \ldots, \hat{i}_\tau\} \subset \mathcal{I} \colon \hat{i}_{[\tau]} \neq v_{[\tau]}\right\}$. Finally, assume, without loss of generality, that the subsets of distinct indices $\hat{i}_{[\tau]} \in \mathcal{T}$ are ordered in natural lexicographical order, i.e., $\hat{i}_1 < \hat{i}_2 < \cdots < \hat{i}_\tau$.

Next, from the deterministic linear mapping between the candidate functions evaluations, $\hat{\mathsf{V}}_{\mu\times r}$, we have $U_*^{(v_\ell)} = \hat{v}_{v_\ell,1} U_*^{(1)} + \cdots + \hat{v}_{v_\ell,r} U_*^{(r)}$, $\ell \in [\tau]$, where $(\hat{v}_{v_\ell,1}, \ldots, \hat{v}_{v_\ell,r}) = \hat{\boldsymbol{v}}_{v_\ell}$. Now, we need to show that q_0 is a linear function of the queries of $\tilde{\mathcal{Q}}$ as follows:

$$q_0 = \sum_{\hat{i}_{[\tau]} \in \mathcal{T}} h(U^{(\hat{i}_{[\tau]})}) q(U^{(\hat{i}_{[\tau]})}), \tag{18}$$

where $h(U^{(\hat{i}_{[\tau]})})$ is a linear coefficient calculated as a function of the deterministic linear mapping coefficients represented by the matrix

$$\hat{\mathsf{V}}^*_{(r-1)\times\tau} = \begin{pmatrix} \hat{v}_{v_1,2} & \hat{v}_{v_2,2} & \cdots & \hat{v}_{v_\tau,2} \\ \vdots & \vdots & \ldots & \vdots \\ \hat{v}_{v_1,r} & \hat{v}_{v_2,r} & \cdots & \hat{v}_{v_\tau,r} \end{pmatrix}$$

as outlined in [16, Sec. V-B]. Given the above problem setup, notation, and definitions, one can verify that (18) holds for all queries corresponding to Group 2 (refer to [16, Sec. V-B] for the detailed derivation). Thus, a number of $\binom{\mu-r}{\tau}$ query types in Group 2 are redundant and can be removed from the query set.

REFERENCES

[1] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Capacity of private linear computation for coded databases," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput.*, Oct. 2018, pp. 813–820.

[2] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. 36th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Oct. 1995, pp. 41–50.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–982, Nov. 1998.

[4] W. Gasarch, "A survey on private information retrieval," *Bull. EATCS*, no. 82, pp. 72–107, Feb. 2004.

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

[6] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, Mar. 2011.

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

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

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

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

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

[12] 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.

[13] R. Tajeddine, O. W. Gnilke, and S. El Rouayheb, "Private information retrieval from MDS coded data in distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 11, pp. 7081–7093, Nov. 2018.

[14] 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 Geometry*, vol. 1, no. 1, pp. 647–664, Nov. 2017.

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

[16] H. Sun and S. A. Jafar, "The capacity of private computation," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3880–3897, Jun. 2019.

[17] Z. Chen, Z. Wang, and S. A. Jafar, "The asymptotic capacity of private search," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4709–4721, Aug. 2020.

[18] S. A. Obead and J. Kliewer, "Achievable rate of private function retrieval from MDS coded databases," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2117–2121.

[19] D. Karpuk, "Private computation of systematically encoded data with colluding servers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2112–2116.

[20] N. Raviv and D. A. Karpuk, "Private polynomial computation from Lagrange encoding," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 553–563, 2020.

[21] M. Mirmohseni and M. A. Maddah-Ali, "Private function retrieval," in *Proc. Iran Workshop Commun. Inf. Theory (IWCIT)*, Apr. 2018, pp. 1–6.

[22] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Private polynomial computation for noncolluding coded databases," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1677–1681.

[23] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "On the capacity of private nonlinear computation for replicated databases," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Aug. 2019, pp. 1–5.

[24] Y. Yakimenka, H.-Y. Lin, and E. Rosnes, "On the capacity of private monomial computation," in *Proc. Int. Zurich Sem. Inf. Commun. (IZS)*, Feb. 2020, pp. 31–35.

[25] A. Heidarzadeh and A. Sprintson, "Private computation with side information: The single-server case," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1657–1661.

[26] A. Heidarzadeh and A. Sprintson, "Private computation with individual and joint privacy," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 1112–1117.

[27] H.-Y. Lin, S. Kumar, E. Rosnes, and A. Graell i Amat, "Asymmetry helps: Improved private information retrieval protocols for distributed storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.

[28] H.-Y. Lin, S. Kumar, E. Rosnes, and A. Graell i Amat, "On the fundamental limit of private information retrieval for coded distributed storage," Aug. 2018, *arXiv:1808.09018*.

[29] 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," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2107–2118, Apr. 2019.

[30] J. Radhakrishnan, "Entropy and counting," in *Computational Mathematics, Modelling and Algorithms*, J. C. Misra, Ed. New Delhi, India: Narosa Publishing House, 2003, pp. 146–168.

[31] J. Xu and Z. Zhang, "On sub-packetization and access number of capacity-achieving PIR schemes for MDS coded non-colluding servers," *Sci. China Inf. Sci.*, vol. 61, no. 10, Oct. 2018, Art. no. 100306.