On Multi-Message Private Computation

Ali Gholami*, Kai Wan†, Tayyebeh Jahani-Nezhad*, Hua Sun‡, Mingyue Ji§, Giuseppe Caire*
*Technische Universität Berlin, 10587 Berlin, Germany, {a.gholami, caire, t.jahani.nezhad}@tu-berlin.de
†Huazhong University of Science and Technology, China, kai_wan@hust.edu.cn
‡University of North Texas, Denton, TX 76203, USA, hua.sun@unt.edu
§University of Utah, Salt Lake City, UT 84112, USA, mingyue.ji@utah.edu

2024 IEEE International Symposium on Information Theory (ISIT) | 979-8-3503-8284-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISIT57864.2024.10619493

***Abstract*—In a typical formulation of the private information retrieval (PIR) problem, a single user wishes to retrieve one out of K files from N servers without revealing the demanded file index to any server. This paper formulates an extended model of PIR, referred to as multi-message private computation (MM-PC), where instead of retrieving a single file, the user wishes to retrieve $P > 1$ linear combinations of files while preserving the privacy of the demand information. The MM-PC problem is a generalization of the private computation (PC) problem (where the user requests one linear combination of the files), and the multi-message private information retrieval (MM-PIR) problem (where the user requests $P > 1$ files). A baseline achievable scheme repeats the optimal PC scheme by Sun and Jafar P times, or treats each possible demanded linear combination as an independent file and then uses the near optimal MM-PIR scheme by Banawan and Ulukus. In this paper, we propose an achievable MM-PC scheme that significantly improves upon the baseline scheme. Doing so, we design the queries inspired from the structure in the cache-aided scalar linear function retrieval scheme, where they leverage the dependency between messages to reduce the amount of communication. To ensure the decodability of our scheme, we propose a new method to benefit from the existing dependency, referred to as the sign assignment step. In the end, we use Maximum Distance Separable matrices to code the queries, which allows the reduction of download from the servers, while preserving privacy.**

I. Introduction

In the private information retrieval (PIR) problem [1], a user wishes to download a file by sending different queries to a group of N non-colluding servers each storing the same K files, while keeping the identity of the desired file secret from the servers. The information-theoretic capacity is defined as the maximum number of bits of desired information decoded per one bit of downloaded information. The authors in [1] show that the capacity of PIR is given by $\frac{1-1/N}{1-1/N^K}$.

Following the seminal PIR result in [1], a large number of works have considered extended models of PIR. In particular, in [2], [3], the problem of **private computation (PC)** is proposed. In general, linear and multivariate polynomial operations are widely used as fundamental primitives for building the complex queries that support online big-data analysis and data mining procedures. In these scenarios, it is too resource-consuming to locally download all input variables in order to compute the desired output value. Based on this motivation, the PC problem was considered in [2], [3], where instead of retrieving a single file, the user requests a (scalar) linear combination of the files among M possible linear combinations, where each linear combination is called a message. An optimal PC scheme has been proposed in [2]. It is interesting to note that the capacity of the PC problem is exactly the same as that of the PIR problem, which is independent of M. Several extended models of the PC problem have been considered, including PC with coded storages at the servers [4]–[6], private sequential function retrieval [7] (where the user wants to compute a fixed set of linear combinations while hiding the computation order), PC with polynomial functions [8], [9], cache-aided PC [10], single-server PC [11], and more.

Another line of works in PIR is the **multi-message PIR (MM-PIR)** proposed in [12]. Instead of retrieving a single file, in the MM-PIR problem, the user aims to retrieve $P > 1$ files. A near-optimal MM-PIR scheme was proposed in [12]. It is also interesting to note that, even if the requested files are independent, designing the MM-PIR scheme by jointly considering the multi-request (as in [12]) leads to a significant increase in the retrieval rate compared to simply repeating the Sun and Jafar PIR scheme P times. Other works related to MM-PIR include [13], where the problem assuming that the user has private side information is studied, and [14], [15], which consider the MM-PIR problem with side information in the single-server case.

Contributions: In this paper, we formulate a new problem, referred to as the MM-PC problem, which covers the PC and MM-PIR problems as special cases. In this setting, there are N non-colluding servers, each storing a library of M messages with arbitrary linear dependencies, of which K are linearly independent. The user wants to retrieve a set of P linearly independent messages from the servers, while keeping the identity of the requested messages secret from each server.

An achievable scheme by a direct extending of the optimal PC scheme in [2] or the near optimal MM-PIR scheme in [12], is proposed which we refer to as the baseline scheme.

However, the direct combination of the PC scheme in [2] and the MM-PIR scheme in [12] is not possible. We propose an improved scheme to the baseline scheme, by incorporating some ideas of these two schemes with additional novel ideas. More precisely, while each message is divided into multiple symbols and the queries are essentially linear combinations of these symbols, to exploit the dependency between messages, we may need to assign a specific sign to each symbol involved, referred to as sign assignment. To ensure decodability, and inspired from [16], we propose a new sign assignment method which results that some of the queries become linear combinations of others, and then by

Authorized licensed use limited to: University of Florida. Downloaded on March 26,2025 at 16:18:11 UTC from IEEE Xplore. Restrictions apply.

using Maximum Distance Separable (MDS) coding, we can reduce the amount of download, while preserving symmetry and thus privacy. It is essential to mention that the redundancy appears as a result of the sign assignment and the novel index assignment introduced. Numerical evaluations show that the improved scheme provides performance gain with respect to the baseline scheme for a wide range of system parameters.

Notation: For $a \in \mathbb{N}$ the notation $[a]$ represents set $\{1, \dots, a\}$. In addition, we denote the difference of two sets $\mathcal{A}$, $\mathcal{B}$ as $\mathcal{A} \backslash \mathcal{B}$, that means the set of elements which belong to $\mathcal{A}$ but not $\mathcal{B}$.

II. Problem setting

Consider N non-colluding servers with K files which are replicated on all servers. For each $i \in [K]$, the i^{th} file is a vector of large enough size L, denoted by $W_{d_i} \in \mathbb{F}_q^L$, whose symbols take on values over a finite field $\mathbb{F}_q$. Additionally, files are independently and randomly generated with i.i.d. symbols such that

$$H(W_{d_1}) = \cdots = H(W_{d_K}) = L, \tag{1a}$$

$$H(W_{d_1}, \cdots, W_{d_K}) = H(W_{d_1}) + \cdots + H(W_{d_K}). \tag{1b}$$

where $H(\cdot)$ denotes entropy. Note that in this paper, the $\log$ used for information measures in the entropy function is base-q. A user wants to retrieve P of M possible messages from the servers, where each message is a linear combination of the K files. For each $m \in [M]$, the m^{th} message is defined as,

$$W_m := \mathbf{v}_m [W_{d_1}, \dots, W_{d_K}]^T \tag{2a}$$

$$= v_m(1) W_{d_1} + \cdots + v_m(K) W_{d_K}, \tag{2b}$$

where $v_m(i)$ is the i^{th} entry of the coefficient vector $\mathbf{v}_m$ for $i \in [K]$, and all operations are taken in $\mathbb{F}_q$. Without loss of generality, we assume that $M \geq K$ and the first K messages are replicas of the K independent files, i.e., $(W_1, \dots, W_K) = (W_{d_1}, \dots, W_{d_K})$.

Unlike [2] where the user requires only one message, in this paper we formulate the multi-message private computation (MM-PC) problem. In this scenario, the user privately generates a set of P indices $\mathcal{I} = \{\theta_1, \dots, \theta_P\}$, where $\mathcal{I} \subset [M]$ and $\theta_i \neq \theta_j$ for each $i, j \in [P]$ where $i \neq j$. The user wishes to compute $W_{\mathcal{I}} := (W_{\theta_1}, \dots, W_{\theta_P})$ while keeping $\mathcal{I}$ secret from each server. Without loss of generality, we assume that $W_{\theta_1}, \dots, W_{\theta_P}$ are linearly independent; otherwise, we can just reduce P and let the user demand linearly independent combinations. To do so, the user generates N queries $Q_1^{\mathcal{I}}, \dots, Q_N^{\mathcal{I}}$ and sends each $Q_n^{\mathcal{I}}$ to the corresponding server. These queries are generated when the user has no knowledge of the realizations of the messages, so the queries should be independent of the messages, i.e.,

$$I(Q_1^{\mathcal{I}}, \dots, Q_N^{\mathcal{I}}; W_1, \dots, W_M) = 0. \tag{3}$$

where $I(\cdot;\cdot)$ denotes mutual information. Upon receiving $Q_n^{\mathcal{I}}$, each server $n \in [N]$ generates and sends the answer $A_n^{\mathcal{I}}$ which is a function of $Q_n^{\mathcal{I}}$ and $W_1, \dots, W_M$, i.e.,

$$H(A_n^{\mathcal{I}} | Q_n^{\mathcal{I}}, W_1, \dots, W_M) = 0, n \in [N]. \tag{4}$$

Finally, the user must retrieve the desired $W_{\mathcal{I}}$ from the servers' answers $A_n^{\mathcal{I}}$ and the queries $Q_n^{\mathcal{I}}$ with vanishing error[1], i.e.,

$$H(W_{\mathcal{I}} | A_1^{\mathcal{I}}, \dots, A_N^{\mathcal{I}}, Q_1^{\mathcal{I}}, \dots, Q_N^{\mathcal{I}}) = o(L), \tag{5}$$

where $\lim_{L \to \infty} o(L)/L = 0$.

The MM-PC scheme should be designed to keep the demand information $\mathcal{I}$ secret from all servers; i.e., the following privacy constraint must be satisfied,

$$(Q_n^{\mathcal{I}_1}, A_n^{\mathcal{I}_1}, W_1, \dots, W_M) \sim (Q_n^{\mathcal{I}_2}, A_n^{\mathcal{I}_2}, W_1, \dots, W_M), \tag{6}$$

for all $\mathcal{I}_1, \mathcal{I}_2 \in \Omega$ and all servers $n \in [N]$, where Ω is the set of all possible $\mathcal{I}$, and $\sim$ indicates that these two random vectors follow the same distribution.

The MM-PC *rate* denoted by R is defined as the number of symbols recovered collectively from all the demanded messages per one downloaded symbol, $R := \frac{PL}{D}$, where D is the expected value over random queries of the total downloaded symbols from all the servers by the user. The objective is to find the supremum of all achievable rates, denoted by $R^\star$.

III. Main Results

In this section, we present the baseline scheme and the main results for the proposed MM-PC problem. There are two direct solutions to the MM-PC problem, from the PC and MM-PIR schemes in [2], [12], respectively. The first one is to simply use the PC scheme for each demanded message separately, while the second one treats each possible demanded linear combination as an independent message and then uses the MM-PIR scheme. These construct the baseline scheme.

Theorem 1 (Baseline scheme)**.** *For the MM-PC problem, the following rate is achievable,*

$$R_1 = \max \left\{ \frac{1 - \frac{1}{N}}{1 - (\frac{1}{N})^K} + \frac{(P-1)(N-1)}{N^M \left(1 - (\frac{1}{N})^K\right)}, C_{M,P} \right\}, \tag{7}$$

where $C_{M,P}$ represents the achieved rate of the MM-PIR scheme in [12] with M files in the library and P requests from the user.

Theorem 2 (Proposed scheme)**.** *For the MM-PC problem, in case $P < K$, the following rate is achievable,*

$$R_2 = \frac{P \sum_{i=1}^{M-P+1} \alpha_i \binom{M-P}{i-1}}{\sum_{i=1}^{M-P+1} \alpha_i \left(\binom{M-P}{i} - \binom{M-K}{i} + P \binom{M-P}{i-1} \right)}, \tag{8}$$

where $\alpha_{M-P+2} = \cdots = \alpha_M = 0$, $\alpha_{M-P+1} = (N-1)^{M-P}$, and

$$\alpha_i = \frac{1}{N-1} \sum_{m=1}^{P} \binom{P}{m} \alpha_{i+m}, \quad i \in [M-P]. \tag{9}$$

Fig. 1 compares the baseline scheme with the proposed scheme, for the case where $K = 7$, $N = 2$, $M \in \{10, 15\}$, and $P \in \{2, 3, 4, 5, 6\}$. As shown in Fig. 1, when $P = 2$, the baseline scheme is slightly better than the proposed scheme;

[1]The MM-PC scheme proposed in this paper however, has zero probability of error.

Authorized licensed use limited to: University of Florida. Downloaded on March 26,2025 at 16:18:11 UTC from IEEE Xplore. Restrictions apply.

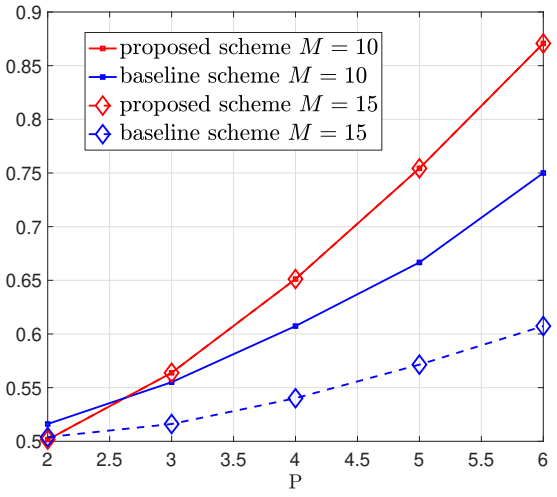

Fig. 1: Comparison of rates. The parameters are $K = 7, N = 2$. P and M change as in the figure.

when $P > 2$, the improvement over the baseline scheme becomes more significant as P increases. It can be seen from Fig. 1 that, the rate of the proposed scheme has very little dependence on M. This is because when M increases while K remains the same, more redundancy appears by our construction and by removing the redundancy, the rate does not change much. Note that in the optimal PC scheme [2], the optimal rate does not depend on M, by a similar reason.

IV. The Proposed Scheme Through an Example

Due to the space limitation and for the sake of clarity, we illustrate the proposed MM-PC scheme through an example, for parameters $M = 5$, $K = 3$, $P = 2$, and $N = 2$. The general scheme and all the proofs (including decodability and privacy) can be found in [17]. The proposed scheme is inspired from the optimal PC scheme in [2] and the near optimal MM-PIR scheme (for $P \leq \frac{M}{2}$) in [12]. However, the direct combination of these two schemes does not work in the MM-PC problem.[2] Instead, we propose a new approach to incorporate the indexing structure and sign assignment in order to reduce transmission by removing redundancy.

In this example, the messages are denoted by letters $\{a, b, c, d, e\}$, where $\{a, b, c\}$ are the independent ones and $\{d, e\}$ are any desired linear combinations of $\{a, b, c\}$ in the given finite field $\mathbb{F}_q$. The demanded messages are $\mathcal{I} = \{a, b\}$. Each message is partitioned into $L = 68$ symbols and the i^{th} symbol of each message is denoted by subscript (index) i, e.g., a_i denotes the i^{th} symbol of message a. There exists

[2]This is because in the PC scheme, each possible demanded linear combination of messages is first treated as a 'message' in the PIR problem; then by applying a smart permutation of indices of 'message symbols' and a sign assignment (-1 or 1) on each symbol, into the M-'message' PIR scheme by Sun and Jafar, some transmissions are redundant which could be removed to reduce transmission; after removing the redundant ones, the number of transmissions by each server is exactly the same as the optimal one for the K-message PIR problem. However, the MM-PIR scheme is built on another permutation of message symbols, where a direct combination of these two schemes will not lead to redundancy to remove.

Round	Stage	server 1	server 2
round 1	stage 1	a_1, b_1, c_1, d_1, e_1	$a_{13}, b_{13}, c_{13}, d_{13}, e_{13}$
	$\vdots$	$\vdots$	$\vdots$
	stage 12	$a_{12}, b_{12}, c_{12}, d_{12}, e_{12}$	$a_{24}, b_{24}, c_{24}, d_{24}, e_{24}$

TABLE I: Round 1 of queries.

a closed formula for the number of symbols, referred to as subpacketization level L, based on the system parameters as $L = N \sum_{i=1}^{M-P+1} \alpha_i \binom{M-P}{i-1}$. We provide the details in the extended version in [17] and some explanation in remark 1.

Step 1: Permutation on the Symbols. A permutation function $\pi(\cdot)$ on the elements in $[L]$ is chosen uniformly at random over all the $L!$ possiblilities. The symbols of every message are permuted by this function. For simplicity, the permuted messages are denoted by the same letters $\{a, b, c, d, e\}$, e.g., message $a = (a_1, a_2, ..., a_{68})$ turns into $a = (a_{\pi(1)}, a_{\pi(2)}, ..., a_{\pi(68)})$. Furthermore, we define the variables $\sigma_i \in \{-1, 1\}, \forall i \in [L]$, referred to as multiplicative factors, each chosen uniformly i.i.d. For all messages, the symbol of position i is multiplied by σ_i. For example, the altered message a eventually turns into $a = (\sigma_1 a_{\pi(1)}, \sigma_2 a_{\pi(2)}, ..., \sigma_{68} a_{\pi(68)})$. For the ease of description, in this example we assume that the permutation is $(1, 2, \ldots, 68)$ and that $\sigma_i = 1, \forall i \in [68]$.

We also perform a relabeling on the message labels $1, ..., M$, such that the first P labels of messages (i.e., $W_1, ..., W_P$) are the demanded messages. Furthermore, we change the set of independent messages, such that they contain the P demanded messages. It is proved in [17] that these actions will not hurt the decodability and privacy of our scheme. In this example, this is already the case and there is no change needed.

Step 2: Number of Stages. The queries to servers are linear combinations of symbols from different messages. They are categorized into multiple *rounds*, where round i contains queries summing i different symbols. Each round itself is also split into multiple *stages*. Each stage of round i contains all $\binom{M}{i}$ choices of i messages from the total M. For instance for a stage of round 2, the queries are of the form $\{a_* + b_*, a_* + c_*, a_* + d_*, a_* + e_*, b_* + c_*, b_* + d_*, b_* + e_*, c_* + d_*, c_* + e_*, d_* + e_*\}$, which covers all $\binom{5}{2} = 10$ ways of choosing 2 messages from the total 5. Note that the subscript $*$ denotes some specific index. The number of stages of round i denoted by α_i, follows (9). The explanation to calculate α_i has appeared is the detailed version of this paper in [17]. For our example we have $\alpha_5 = 0, \alpha_4 = 1, \alpha_3 = 2, \alpha_2 = 5, \alpha_1 = 12$.[3]

Step 3: Initialization. This step corresponds to queries of round 1 (single symbols). Since $\alpha_1 = 12$, from each server the user queries 12 symbols of each message, depicted in Table I.

Step 4: Index Assignment. This is the step to determine symbol indices in the queries. Consider the first stage of round 2 queries to server 1. The queries of the form $\{a_* + c_*, a_* + d_*, a_* + e_*\}$ are used to decode new symbols of message a.

[3]The number of stages calculated here is completely different from that of [12]. The main reason is that in the scheme [12], every query containing symbols of demanded messages contributes to decoding new demanded symbols, while in the proposed scheme because of the special index assignment, designed in cooperation with the sign assignment, this is not possible.

Authorized licensed use limited to: University of Florida. Downloaded on March 26,2025 at 16:18:11 UTC from IEEE Xplore. Restrictions apply.

Round	Stage	Server 1	Server 2	Stage	Server 1	Server 2
round 2	stage 1	$a_{25}-c_{13}$	$a_{28}-c_1$	stage 4	$a_{43}-c_{19}$	$a_{46}-c_7$
		$a_{26}+d_{13}$	$a_{29}+d_1$		$a_{44}+d_{19}$	$a_{47}+d_7$
		$a_{27}+e_{13}$	$a_{30}+e_1$		$a_{45}+e_{19}$	$a_{48}+e_7$
		$b_{25}-c_{14}$	$b_{28}-c_2$		$b_{43}-c_{20}$	$b_{46}-c_8$
		$b_{26}+d_{14}$	$a_{29}+d_2$		$b_{44}+d_{20}$	$a_{47}+d_8$
		$b_{27}+e_{14}$	$b_{30}+e_2$		$a_{20}+b_{19}$	$b_{48}+e_8$
		$a_{14}-b_{13}$	a_2-b_1		a_6-b_5	a_8-b_7
		$c_{26}+d_{25}$	$c_{29}+d_{28}$		$c_{44}+d_{43}$	$c_{47}+d_{46}$
		$c_{27}+e_{25}$	$c_{30}+e_{28}$		$c_{45}+e_{43}$	$c_{48}+e_{46}$
		$d_{27}-e_{26}$	$d_{30}-e_{29}$		$d_{45}-e_{44}$	$d_{48}-e_{47}$
	stage 2	$a_{31}-c_{15}$	$a_{34}-c_3$	stage 5	$a_{49}-c_{21}$	$a_{52}-c_9$
		$a_{32}+d_{15}$	$a_{35}+d_3$		$a_{50}+d_{21}$	$a_{53}+d_9$
		$a_{33}+e_{15}$	$a_{36}+e_3$		$a_{51}+e_{21}$	$a_{54}+e_9$
		$b_{31}-c_{16}$	$b_{34}-c_4$		$b_{49}-c_{22}$	$b_{52}-c_{10}$
		$b_{32}+d_{16}$	$a_{35}+d_4$		$b_{50}+d_{22}$	$a_{53}+d_{10}$
		$b_{33}+e_{16}$	$b_{36}+e_4$		$b_{51}+e_{22}$	$b_{54}+e_{10}$
		$a_{16}-b_{15}$	a_4-b_3		$a_{22}-b_{21}$	$a_{10}-b_9$
		$c_{32}+d_{31}$	$c_{35}+d_{34}$		$c_{50}+d_{49}$	$c_{53}+d_{52}$
		$c_{33}+e_{31}$	$c_{36}+e_{34}$		$c_{51}+e_{49}$	$c_{54}+e_{52}$
		$d_{33}-e_{32}$	$d_{36}-e_{35}$		$d_{51}-e_{50}$	$d_{54}-e_{53}$
	stage 3	$a_{37}-c_{17}$	$a_{40}-c_5$			
		$a_{38}+d_{17}$	$a_{41}+d_5$			
		$a_{39}+e_{17}$	$a_{42}+e_5$			
		$b_{37}-c_{18}$	$b_{40}-c_6$			
		$b_{38}+d_{18}$	$a_{41}+d_6$			
		$b_{39}+e_{18}$	$b_{42}+e_6$			
		$a_{18}-b_{17}$	a_6-b_5			
		$c_{38}+d_{37}$	$c_{41}+d_{40}$			
		$c_{39}+e_{37}$	$c_{42}+e_{40}$			
		$d_{39}-e_{38}$	$d_{42}-e_{41}$			

TABLE II: Round 2 of queries.

Thus, the other symbol involved should be downloaded previously, leading to the queries $\{a_{25}+c_{13}, a_{26}+d_{13}, a_{27}+e_{13}\}$, where $\{25,26,27\}$ are new indices of message a (the first 24 indices are already used in round 1) and the remaining parts are symbols with index 13, already received from queries to server 2 in round 1 and used as side information to decode for new symbols of message a. Similarly in the same stage, for message b the queries would be $\{b_{25}+c_{14}, b_{26}+d_{14}, b_{27}+e_{14}\}$, which use symbols with index 14 as side information.

Now we determine the indices in the remaining queries $\{a_*+b_*, c_*+d_*, c_*+e_*, d_*+e_*\}$. To do so, we use the observation that symbols of a have been added to symbols with index 13 and symbols of b to symbols with index 14. Also, symbols of c have been added to symbols with index 25, symbols of d to symbols with index 26, and symbols of e to symbols with index 27. This structure should be preserved to keep the symmetry of queries regarding indices. This forces the remaining queries to be $\{a_{14}+b_{13}, c_{26}+d_{25}, c_{27}+e_{25}, d_{27}+e_{26}\}$. For the first stage of round 2 queries to server 2, the same process repeats with symbols with indices 1 and 2 acting as side information and decoding new symbols with indices $\{28,29,30\}$ for messages a and b. The other 4 stages of round 2 follow the same procedure. The final queries appear in Table II. We postpone the explanation of minus signs in the table to step 5, since here our focus is just the indices. For now the reader should assume all signs are pluses.

A stage of round 3 contains all $\binom{5}{3}$ ways of choosing 3 messages out of 5, i.e., $\{a_*+b_*+c_*, a_*+b_*+d_*, a_*+b_*+e_*, a_*+c_*+d_*, a_*+c_*+e_*, a_*+d_*+e_*, b_*+c_*+d_*, b_*+c_*+e_*, b_*+d_*+e_*, c_*+d_*+e_*\}$. The queries $\{a_*+c_*+d_*, a_*+c_*+e_*, a_*+d_*+e_*\}$, which are combinations of a with $\{c,d,e\}$, are used to decode new symbols of a. So the remaining parts should be the side information part already received from round 2 queries. Consider the first stage of round 3 queries to server 1. The new indices for a are $\{55,56,57\}$, since the first 54 have already appeared in the first two rounds. The side information part is duplicated from the first stage of round 2 queries to server 2, i.e., $\{c_{29}+d_{28}, c_{30}+e_{28}, d_{30}+e_{29}\}$. Therefore, these queries would be $\{a_{55}+c_{29}+d_{28}, a_{56}+c_{30}+e_{28}, a_{57}+d_{30}+e_{29}\}$. Similarly for decoding new symbols of b, the queries are $\{b_{55}+c_{35}+d_{34}, b_{56}+c_{36}+e_{34}, b_{57}+d_{36}+e_{35}\}$, where the side information parts are duplicated from the second stage of round 2 queries to server 2. One observes that when c and d appear in a query, the index of the other symbol involved is the same, i.e., in $\{a_{55}+c_{29}+d_{28}, b_{55}+c_{35}+d_{34}\}$, both a and b have index 55. Or when a and d appear, the other symbol has index 29. One can verify that the same structure holds for any choice of two messages. To preserve privacy, we keep this structure for all queries. Using this rule, all indices would be determined. Among remaining queries $\{a_*+b_*+c_*, a_*+b_*+d_*, a_*+b_*+e_*, c_*+d_*+e_*\}$, take $a_*+b_*+c_*$. To determine the index of a, we search for a query containing both b and c, e.g., $b_{55}+c_{35}+d_{34}$. Since d has index 34, a should also have index 34. To determine the index for b, since in the query $a_{55}+c_{29}+d_{28}$, containing both a and c, the index for d is 28, b should also have index 28. To determine the index for c, since there has no query containing both a and b already appeared, symbols with index 23 queried in the first round of queries to server 2 are used. Similarly, all other indices are determined. Round 4 queries follow the same logic. Round 3 and 4 queries are depicted in Table III.

Remark 1 (subpacketization L)**.** *By the index assignment, it is evident how many new symbols for demanded messages appear in each stage. Summing the numbers for all stages, determines the subpacketization L required, which in the example equals 68. The general explanation appears in [17].*

The general rule for index assignment is as follows. In a stage of round i, choose any $i-1$ messages from the total M. Any query containing symbols of these messages has the same index for the remaining symbol involved. This indexing structure resembles that of multicast transmissions with alternate sign assignment in the literature of coded caching [16]; therefore inspired from the above scheme, we introduce a new method to leverage the dependency among messages to reduce the number of queries.

Till now, the only operation used in queries is addition. To exploit the dependency between messages, we may need to also use negation, referred to as sign assignment. It is proved in the detailed version of this paper in [17, Theorem 3], that after sign assignment, in a stage of round i, out of $\binom{M}{i}$ total queries, $\binom{M-K}{i}$ of them are redundant and can be written as linear combinations of others. We should point out that this redundancy result is also achieved by the PC scheme in [2], however, due to the fact that we attempt to retrieve multiple messages instead of one, it is not possible to utilize the sign assignment in their scheme. We have designed a completely new sign assignment method which reaches this redundancy while keeping the decodability possible.

Step 5: Sign Assignment. First, each query is first divided into two parts: The first part includes symbols of

Authorized licensed use limited to: University of Florida. Downloaded on March 26,2025 at 16:18:11 UTC from IEEE Xplore. Restrictions apply.

Round	Stage	server 1	server 2
round 3	stage 1	$a_{55}-c_{29}-d_{28}$ $a_{56}-c_{30}-e_{28}$ $a_{57}-d_{30}+e_{29}$ $b_{55}-c_{35}-d_{34}$ $b_{56}-c_{36}-e_{34}$ $b_{57}-d_{36}+e_{35}$ $a_{34}-b_{28}+c_{23}$ $a_{35}-b_{29}-d_{23}$ $a_{36}-b_{30}-e_{23}$ $c_{57}-d_{56}+e_{55}$	$a_{58}-c_{26}-d_{25}$ $a_{59}-c_{27}-e_{25}$ $a_{60}-d_{27}+e_{26}$ $b_{58}-c_{32}-d_{31}$ $b_{59}-c_{33}-e_{31}$ $b_{60}-d_{33}+e_{32}$ $a_{31}-b_{25}+c_{11}$ $a_{32}-b_{26}-d_{11}$ $a_{33}-b_{27}-e_{11}$ $c_{60}-d_{59}+e_{58}$
	stage 2	$a_{61}-c_{41}-d_{40}$ $a_{62}-c_{42}-e_{40}$ $a_{63}-d_{42}+e_{41}$ $b_{61}-c_{47}-d_{46}$ $b_{62}-c_{48}-e_{46}$ $b_{63}-d_{48}+e_{47}$ $a_{46}-b_{40}+c_{24}$ $a_{47}-b_{41}-d_{24}$ $a_{48}-b_{42}-e_{24}$ $c_{63}-d_{62}+e_{61}$	$a_{64}-c_{38}-d_{37}$ $a_{65}-c_{39}-e_{37}$ $a_{66}-d_{39}+e_{38}$ $b_{64}-c_{44}-d_{43}$ $b_{65}-c_{45}-e_{43}$ $b_{66}-d_{45}+e_{44}$ $a_{43}-b_{37}+c_{12}$ $a_{44}-b_{38}-d_{12}$ $a_{45}-b_{39}-e_{12}$ $c_{66}-d_{65}+e_{64}$
round 4	stage 1	$a_{67}-c_{60}+d_{59}-e_{58}$ $b_{67}-c_{66}+d_{65}-e_{64}$ $a_{64}-b_{58}+c_{53}+d_{52}$ $a_{65}-b_{59}+c_{54}+e_{52}$ $a_{66}-b_{60}+d_{54}-e_{53}$	$a_{68}-c_{57}+d_{56}-e_{55}$ $b_{68}-c_{63}+d_{62}-e_{61}$ $a_{61}-b_{55}+c_{50}+d_{49}$ $a_{62}-b_{56}+c_{51}+e_{49}$ $a_{63}-b_{57}+d_{51}-e_{50}$

TABLE III: Rounds 3 and 4 of queries.

independent messages, and the second part includes symbols from dependent messages. So each query q is written as

$$q = (\text{independent symbols}) \pm (\text{dependent symbols}), \quad (10)$$

where in each parenthesis, symbols are ordered based on the label of the message (ranging from 1 to M), from lowest to highest. The signs in each parenthesis are changing alternatively between $+$ and $-$, with the first symbol taking $+$. When the plus sign is used in (10), the sign assignment is called *structure plus* and when minus sign is used, is called *structure minus*. Round 2 queries use structure plus, then round 3 uses minus, and round 4 again uses plus. After these steps, each query is solely randomly multiplied by a $+1$ or -1, uniformly at random. We assume all to be $+1$ in the example. The tables are already depicted with the sign assignment enforced.

As an example, take the queries in the first stage of round 2 to server 1, and assume $d{=}a{+}b$, $e{=}b{+}c$. It is easily verified that $(d_{27}-e_{26})-(a_{27}+e_{13})-(b_{27}+e_{14})+(b_{26}+d_{14})+(c_{26}+d_{25})-(a_{14}-b_{13})-(a_{25}-c_{13})-(b_{25}-c_{14})=0$ holds true, meaning one query is redundant as claimed in [17, Theorem 3].

Step 6: Remove Redundancy. In the first stage of round 2 queries to server 1, as we have seen, one query is redundant. The general statement is that queries containing no symbols from independent messages can be written as linear combinations of those which include independent messages. For example, if we name the 10 queries to server 1 in round 2 from q_1 to q_{10} in the order of appearance, q_{10} is the query we consider as redundant. On the other hand $q_7 = a_{14} - b_{13}$ is also non-useful, since all symbols with indices 13 and 14 have already been queried during round 1. Since directly deleting these queries would jeopardize privacy, we use an MDS (Maximum Distance Separable) coding technique. The final queries in this stage are elements of the vector $\mathbf{q}$ as $\mathbf{q} = \mathbf{G}_{8\times 10} \times [q_1, q_2, ..., q_{10}]^T$, where $\mathbf{G}_{8\times 10}$ is a MDS matrix of size 8×10.[4] Having $\mathbf{q}$, all $q_1, ..., q_{10}$ can be decoded

[4]Since L is large enough, we can represent each of $q_1, q_2, ..., q_{10}$ in some large enough field and then encode them by MDS code, such as Vandermonde or Cauchy matrix.

server 1	server 2
$\mathbf{G}^{(2,1)}_{8\times 10} \times \begin{bmatrix} a_{25}-c_{13} \\ a_{26}+d_{13} \\ \vdots \\ d_{27}-e_{26} \end{bmatrix}$	$\mathbf{G}^{(2,1)}_{8\times 10} \times \begin{bmatrix} a_{28}-c_{1} \\ a_{29}+d_{1} \\ \vdots \\ d_{30}-e_{29} \end{bmatrix}$

TABLE IV: Final queries for the first stage of round 2.

and the reason follows. Since q_{10} can be written as a linear combination of others, we have

$$[q_1, q_2, ..., q_{10}]^T = \mathbf{G}'_{10\times 9} \times [q_1, q_2, ..., q_9]^T, \quad (11)$$

where $\mathbf{G}'_{10\times 9}$ is a full rank matrix of size 10×9. Therefore, $\mathbf{q} = \mathbf{G}^*_{8\times 9} \times [q_1, q_2, ..., q_9]^T$, for some full rank matrix $\mathbf{G}^*_{8\times 9} = \mathbf{G}_{8\times 10}\mathbf{G}'_{10\times 9}$. Thus,

$$\mathbf{q} - q_7 * \mathbf{G}^*_{8\times 9}|_{\{7\}} = \mathbf{G}^*_{8\times 9}|_{[9]\setminus\{7\}} \times [q_1, ..., q_6, q_8, q_9]^T,$$

where $\mathbf{G}^*_{8\times 9}|_{\mathcal{S}}$ denotes the restriction of the matrix to columns in $\mathcal{S}$. Therefore, the values $\{q_1, ..., q_6, q_8, q_9\}$ will be decoded. Since q_7 is already known and q_{10} is dependent to others, all $\{q_1, q_2, ..., q_{10}\}$ are decoded from the 8 queries in $\mathbf{q}$.

This MDS coding technique is done for all the stages in rounds 1 and 2. The size of the MDS matrix in each round is determined by counting the number of redundant queries and the queries that are already known; which are the queries having more than one symbol from the demanded messages. The general explanation of this step appears in [17]. The final queries for the first stage are depicted in Table IV.

Step 7: Shuffling. The order of queries to each server and also the order of the symbols appearing in each query are shuffled, each uniformly at random, to avoid the information leakage from the query orders and the symbol orders.

Remark 2 (rate calculation)**.** *After step* 6*, there are* 3 *queries in each stage of round* 1*,* 8 *in each stage of round* 2*,* 7 *in each stage of round* 3*, and* 2 *in each stage of round* 4*, summing to the total of* 184 *symbols. Since* $L = 68$*, the proposed scheme achieves the rate* $R_2 = 0.74$*, while the baseline scheme achieves* $R_1 = 0.61$*.*

Remark 3 (privacy)**.** *Discussing the privacy intuitively, we note that our design is based on keeping the symmetry of the queries to each server. In every stage, all possible* i*-sums appear, and from the view point of each message, the index structure is symmetric. Besides, using the multiplicative variables* σ_i*, we prove in [17, Appendix C], that the symbols signs appeared in each query have a one to one mapping for different sets of demanded messages; keeping the demanded message indices hidden from the viewpoint of each server.*

Acknowledgement: The work of A. Gholami, T. Jahani-Nezhad, and G. Caire was partially funded by the ERC Advanced Grant N. 789190, CARENET. The work of K. Wan was partially funded by NSFC-12141107. The work of H. Sun was supported in part by NSF under Grant CCF-2007108, Grant CCF-2045656, and Grant CCF-2312228. The work of M. Ji was partially funded by NSF Award 2312227 and CAREER Award 2145835.

Authorized licensed use limited to: University of Florida. Downloaded on March 26,2025 at 16:18:11 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

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

[2] ——, "The capacity of private computation," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3880–3897, 2018.

[3] M. Mirmohseni and M. A. Maddah-Ali, "Private function retrieval," in *2018 Iran Workshop on Communication and Information Theory (IWCIT)*, 2018, pp. 1–6.

[4] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Capacity of private linear computation for coded databases," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2018, pp. 813–820.

[5] ——, "Private linear computation for noncolluding coded databases," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 847–861, 2022.

[6] D. Karpuk, "Private computation of systematically encoded data with colluding servers," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 2112–2116.

[7] B. Tahmasebi and M. A. Maddah-Ali, "Private sequential function computation," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1667–1671.

[8] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Private polynomial function computation for noncolluding coded databases," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1800–1813, 2022.

[9] N. Raviv and D. A. Karpuk, "Private polynomial computation from lagrange encoding," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 553–563, 2020.

[10] Q. Yan and D. Tuninetti, "Robust, private and secure cache-aided scalar linear function retrieval from coded servers," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 968–981, 2022.

[11] A. Heidarzadeh, N. Esmati, and A. Sprintson, "Single-server private linear transformation: The joint privacy case," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 899–911, 2022.

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

[13] S. P. Shariatpanahi, M. J. Siavoshani, and M. A. Maddah-Ali, "Multi-message private information retrieval with private side information," in *2018 IEEE Information Theory Workshop (ITW)*. IEEE, 2018, pp. 1–5.

[14] S. Li and M. Gastpar, "Single-server multi-message private information retrieval with side information," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 173–179.

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

[16] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "On the optimal load-memory tradeoff of cache-aided scalar linear function retrieval," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 4001–4018, 2021.

[17] A. Gholami, K. Wan, H. Sun, M. Ji, and G. Caire, "On multi-message private computation," *arXiv preprint arXiv:2305.05332*, 2023.

Authorized licensed use limited to: University of Florida. Downloaded on March 26,2025 at 16:18:11 UTC from IEEE Xplore. Restrictions apply.