Fundamental Limits of Multi-Message Private Computation

Ali Gholami, *Student Member, IEEE,* Kai Wan, *Member, IEEE,* Tayyebeh Jahani-Nezhad, *Member, IEEE,* Hua Sun, *Member, IEEE,* Mingyue Ji, *Member, IEEE,* and Giuseppe Caire, *Fellow, IEEE*

***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 a new MM-PC scheme that significantly improves upon the baseline schemes. In doing so, we design the queries inspired by the structure in the cache-aided scalar linear function retrieval scheme by Wan *et al.*, which leverages the dependency between linear functions to reduce the amount of communications. 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. By the proposed schemes, we characterize the capacity within a multiplicative factor of 2.**

***Index Terms*—Private computation, multi-message private information retrieval, multiple linear combinations**

I. Introduction

In the private information retrieval (PIR) problem [2], 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 [2] show that the capacity of PIR is given by $\frac{1-1/N}{1-1/N^K}$.

Following the seminal PIR result in [2], a large number of works have considered extended models of PIR. In particular, in [3], [4], 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 is considered in [3], [4], 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 [3]. 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 [5]–[7], private sequential function retrieval [8] (where the user wants to compute a fixed set of linear combinations while hiding the computation order), PC with polynomial functions [9], [10], cache-aided PC [11], single-server PC [12], PC with arbitrary non-linear file dependencies [13], Private Monomial Computation (PMC) (where they consider arbitrary multivariate monomials of messages [14]), and more.

Another line of work in PIR is the **multi-message PIR (MM-PIR)** proposed in [15]. 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 has been proposed in [15]. 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 [15]) 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 [16], where the problem assuming that the user has private side information is studied, and [17], [18], which consider the MM-PIR problem with side information in the single-server case.

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-

A short version of this paper was presented at the 2024 IEEE International Symposium on Information Theory [1].

A. Gholami, T. Jahani-Nezhad, and G. Caire are with the Electrical Engineering and Computer Science Department, Technische Universität Berlin, 10587 Berlin, Germany (e-mail: {a.gholami, caire, t.jahani.nezhad}@tu-berlin.de). The work of G. Caire, A. Gholami, and T. Jahani-Nezhad was supported by the Gottfried Wilhelm Leibniz-Preis 2021 of the German Science Foundation (DFG).

K. Wan is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, 430074 Wuhan, China, (e-mail: kai_wan@hust.edu.cn). The work of K. Wan was partially funded by the National Natural Science Foundation of China (NSFC-12141107), the Key Research and Development Program of Wuhan under Grant 2024050702030100, and Wuhan "Chen Guang" Pragram under Grant 2024040801020211.

H. Sun is with the Department of Electrical Engineering, University of North Texas, Denton, TX 76203, USA (email: hua.sun@unt.edu). The work of Hua Sun was supported in part by NSF under Grant CCF-2045656 and Grant CCF-2312228.

M. Ji is with the Department of Electrical and Computer Engineering in the University of Florida, Gainesville, FL, 32611, USA (email: mingyueji@ufl.edu). The work of M. Ji was supported in part by NSF CAREER Award 2145835 and NSF Award 231222.

colluding servers, each storing a library of K messages, with the possibility of querying M linear combinations of these K messages. 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.

Two recent problems are similar to our formulated MM-PC problem. The private linear transformation (PLT) problem has been considered in [19]–[22]. In the PLT problem, the user also wants to retrieve P linear combinations of $\tilde{K} < K$ files while preserving the privacy of the indices of the $\tilde{K}$ files. The private distributed computing problem has been considered in [23]–[30], where the user wants to compute a matrix multiplication AB_i where $B_1, B_2, \dots$ are matrices with uniform i.i.d. elements while preserving the privacy of the index i. In our considered problem, the results for the above two problems cannot be applied (or are highly inefficient).[1] A very recent work on private multiple linear computation appeared in [31], where the problem is to compute multiple linear combinations of some messages, which are replicated on multiple servers, by considering the case of colluding and non-responsive servers. While keeping the privacy of the requests, the scheme in [31] attains a tradeoff between the communication and computation costs, where the communication cost also includes the upload cost. However, when applied to the MM-PC problem considered here, the scheme of [31] achieves the rate $\frac{N-1}{N}$, which can also be achieved by repeating the PIR scheme in [32] P times. The main challenge addressed in our work is how to improve the repetition strategy.

Contributions: An achievable scheme by a direct extending of the optimal PC scheme in [3] or the near optimal MM-PIR scheme in [15], is proposed which we refer to as the baseline scheme.

However, the direct combination of the PC scheme in [3] and the MM-PIR scheme in [15] is not possible. Hence, we propose a new scheme that improves over the baseline scheme, by leveraging some features of the optimal PC and near optimal MM-PIR schemes and incorporating some non-trivial 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 by [33], we propose a new sign assignment method which makes some of the queries linear combinations of others, and then by 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 novel sign and index assignment method. Numerical evaluations show that the improved scheme provides large performance gains with respect to the baseline scheme for a wide range of system parameters.

[1]More precisely, the PLT schemes cannot be applied to our problem, since in our problem the linear combinations are over all files, and we aim to preserve the privacy of the coefficient matrix instead of chosen files. The private distributed computing schemes are very inefficient to be applied to our problem, since we should treat each possible set of linear combinations as an "independent" demand matrix, and thus there is a huge number of such possible demand matrices.

Notation: For $a \in \mathbb{N}$ the notation $[a]$ represents set $\{1, \dots, a\}$, and notation $[a : b]$ for $a, b \in \mathbb{N}$ represents set $\{a, a+1, \dots, b\}$. 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

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

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., $(W_1, \dots, W_K) = (W_{d_1}, \dots, W_{d_K})$. For the sake of future convenience, each message in $W_1, \dots, W_K$ is called an independent message; each other message is called a dependent message, since it is a linear combination of independent messages.

Unlike [3] where the user requires only one message, in the MM-PC problem, 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.[2] 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.,

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

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.,

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

[2]So in this paper we only need to consider the case $P \leq K$. In addition, if $P = K$, the optimal solution is trivially to download all the files.

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[3], 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,

$$R:=\frac{PL}{D}, \tag{7}$$

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 to 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.

Theorem 1 (Baseline scheme)**.** *For the MM-PC problem, in case $P\le K$, the following rate is achievable,*

$$R_1=\max\left\{\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^K},C_{M,P}\right\}, \tag{8}$$

where $C_{M,P}$ represents the achieved rate of the MM-PIR scheme in [15] with M files in the library and P requests from the user. When $P\ge\frac{M}{2}$,

$$C_{M,P}=\frac{1}{1+\frac{M-P}{PN}}, \tag{9}$$

and when $P\le\frac{M}{2}$,

$$C_{M,P}=\frac{\sum_{i=1}^{P}\gamma_i r_i^{M-P}[(1+\frac{1}{r_i})^M-(1+\frac{1}{r_i})^{M-P}]}{\sum_{i=1}^{P}\gamma_i r_i^{M-P}[(1+\frac{1}{r_i})^M-1]}, \tag{10}$$

in which

$$r_i=\frac{e^{j2\pi(i-1)/P}}{N^{1/P}-e^{j2\pi(i-1)/P}},\forall i\in[P], \tag{11}$$

and $\gamma=(\gamma_1,...,\gamma_P)^T$ is the solution of the system of equations

$$\begin{bmatrix} r_1^{-P} & r_2^{-P} & \dots & r_P^{-P}\\ r_1^{-P+1} & r_2^{-P+1} & \dots & r_P^{-P+1}\\ \vdots & \vdots & \dots & \vdots\\ r_1^{-1} & r_2^{-1} & \dots & r_P^{-1}\end{bmatrix}\begin{bmatrix}\gamma_1\\ \gamma_2\\ \vdots\\ \gamma_P\end{bmatrix}=\begin{bmatrix}(N-1)^{M-P}\\ 0\\ \vdots\\ 0\end{bmatrix}. \tag{12}$$

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

The first rate in (8) is achieved by simply repeating the single-message private computation scheme (PC scheme) in [3]. The second rate in (8) is achieved by treating each possible demanded linear combination as an independent message, and then using the MM-PIR scheme in [15]. The following theorem shows the order optimality of the baseline scheme.

Theorem 2 (order-optimality of the baseline scheme)**.** *The baseline scheme in Theorem 1 is order-optimal within a multiplicative gap of 2.*

The order optimality proof could be found in Appendix C.

Remark 1 (asymptotic optimality of the baseline scheme)**.** *Based on Theorem 2, the gap between the optimal scheme and the baseline scheme is bounded by $\frac{1}{1-\frac{1}{N}}$, which for large N converges to 1. This shows the asymptotic optimality of the baseline scheme for large N.*

Even though the baseline scheme (which treats each linear function as one file and uses the optimal PIR scheme P times) is order-optimal within 2, it can be further improved by carefully leveraging the connection among the linear functions. The achieved rate of our improved scheme is listed in the following theorem, and its description is presented in Section V.

Theorem 3 (Proposed scheme)**.** *For the MM-PC problem, in case $P\le K$, the following rate is achievable,*

$$R_2=\frac{P\sum_{i=1}^{M-P+1}\alpha_i\binom{M-P}{i-1}}{\sum_{i=1}^{M-P+1}\alpha_i\left(\binom{M-P}{i}-\binom{M-K}{i}+P\binom{M-P}{i-1}\right)}, \tag{13}$$

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

$$\alpha_i=\frac{1}{N-1}\sum_{m=1}^{P}\binom{P}{m}\alpha_{i+m},\quad i\in[1:M-P]. \tag{14}$$

The proofs of decodability and privacy of the proposed scheme for Theorem 3 are provided in Appendices A and B, respectively. Fig. 1 compares the baseline scheme with the proposed scheme, for the case where $K=7$, $N=2$, $M\in\{10,15\}$, and $P\in[2:6]$. As shown in Fig. 1, when $P=2$, the baseline scheme is slightly better than the proposed scheme; when $P>2$, the improvement over the baseline scheme becomes more significant as P increases. Note that the rate of the proposed scheme has very little dependence on M, since the solid red line and the dashed red line almost coincide. This can be also observed in Fig. 2, sweeping on M does not change much the rate for the proposed scheme.

Remark 2. *When $P=1$, the rate of the proposed scheme in Theorem 3 equals that of PC capacity $\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^K}$ in [3]. However, the scheme itself would be different from that of [3], since the sign assignment method is different. Therefore, the proposed scheme for $P=1$ introduces a new capacity-achieving PC scheme.*

Remark 3. *When $K\to\infty$, the asymptotic capacities of the PIR and also the PC problems are both $1-1/N$. When $K\gg P$, the asymptotic capacity of the MM-PIR is also $1-1/N$. In addition, when $K\gg P$ (which also means that $M\gg P$), the*

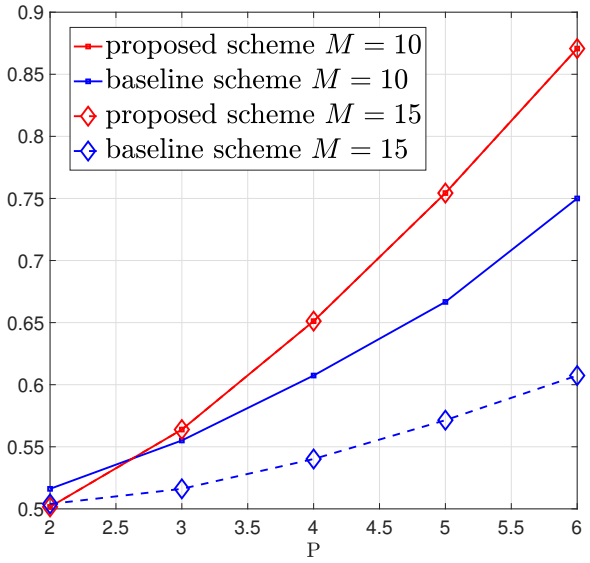


Fig. 1: Comparison of the rates. Red lines represent the proposed scheme and blue lines the baseline scheme, for values $K = 7, N = 2$. P and M changes as in the figure.

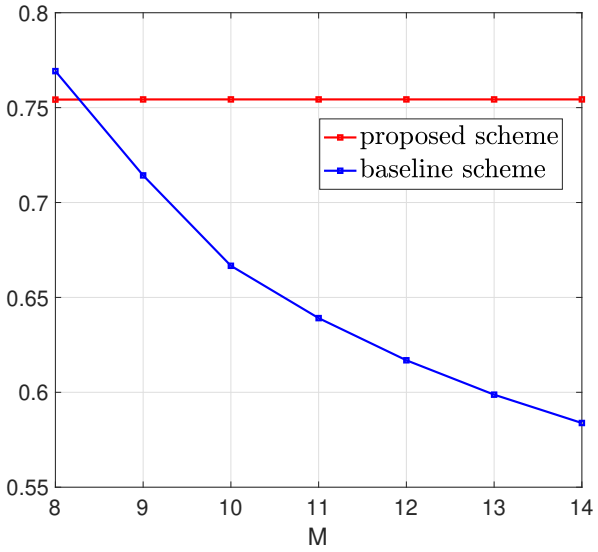


Fig. 2: Comparison of the rates. Red line represents the proposed scheme and blue line the baseline scheme, for values $K = 7, N = 2$, $P = 5$ and M changes as in the figure. The dependency of the proposed scheme on M is almost zero.

asymptotic achieved rate by the baseline scheme of this paper (see the first term in (8), $\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^K}$) is also $1-1/N$. Note that the capacity of the MM-PIR problem is also a converse bound of the considered MM-PC problem. Hence, when $K \gg P$ the asymptotic capacity of the considered MM-PC problem is also $1-1/N$.

IV. New Proposed MM-PC Scheme Through an Example

For the sake of clarity, we illustrate the scheme through an example and provide the general description afterwards.

The step-by-step example in this section considers $M = 5$, $K = 3$, $P = 2$, and $N = 2$. The messages are denoted by letters $\{a, b, c, d, e\}$, where $\{a, b, c\}$ are the independent files and $\{d, e\}$ are any desired linear combinations of the independent files in the given finite field $\mathbb{F}_q$. In this example, the demanded files 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. Note that the general relation between the system parameters and the subpacketization level L follows $L = N\sum_{i=1}^{M-P+1}\alpha_i\binom{M-P}{i-1}$, where the exact calculation appears in Appendix A.

Step 1: Permutation and Relabeling. A permutation function $\pi(\cdot)$ on $[L]$ is chosen uniformly at random over all the $L!$ possibilities. The symbols of every message are permuted by π. 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 message a is transformed into $a = (\sigma_1 a_{\pi(1)}, \sigma_2 a_{\pi(2)}, ..., \sigma_{68} a_{\pi(68)})$. 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. We prove in Appendices A and B that these actions will not hurt the decodability and privacy. In this example, this is already the case and there is no change needed.

Step 2: Query Structure and Number of Repetitions. 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 symbol index. The number of stages of round i denoted by α_i, follows (14). The explanation to calculate α_i is provided in Section V, Step 2. For our example we have $\alpha_5 = 0, \alpha_4 = 1, \alpha_3 = 2, \alpha_2 = 5, \alpha_1 = 12$.[4]

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. Since the general structure of queries is known from Step 2, i.e., the number of stages in each round and that each stage of round i contains all possible i-sums, we need to determine the symbol indices for each query. The index assignment is inspired by the delivery phase of the

[4]The number of stages calculated here is completely different from that of [15]. The main reason is that in the scheme [15], 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.

This article has been accepted for publication in IEEE Transactions on Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2025.3543231

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 queries.

coded caching scheme in [34]. Note that the construction of this coded caching scheme is completely symmetric over files if the number of files is equal to the number of users and each user requests a distinct file.

Consider the first stage of round 2 queries to server 1. There are $\binom{5}{2} = 10$ queries of the form $\{a_*+b_*, a_*+c_*, a_*+d_*, a_*+e_*, b_*+c_*, b_*+d_*, b_*+e_*, c_*+d_*, c_*+e_*, d_*+e_*\}$.

Let us first determine the indices of the queries in $\{a_* + c_*, a_* + d_*, a_* + e_*\}$, where the indices of c_*, d_*, e_* in $\{a_* + c_*, a_* + d_*, a_* + e_*\}$ should be the same (treated as the side information to decode new symbols of message a). In addition, these symbols c_*, d_*, e_* should have been downloaded previously. Hence, we can let these queries be $\{a_{25} + c_{13}, a_{26} + d_{13}, a_{27} + e_{13}\}$, 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. Similarly in the same stage, for the demanded message b the queries should be $\{b_{25}+c_{14}, b_{26}+d_{14}, b_{27}+e_{14}\}$, which use symbols c_{14}, d_{14}, e_{14} as side information.

We then determine the indices in the remaining queries $\{a_* + b_*, c_* + d_*, c_* + e_*, d_* + e_*\}$. Recall that the symbols which are treated as side information to decode a are with index 13, and the symbols which are treated as side information to decode b are with index 14. Hence $a_* + b_*$ should be $a_{14} + b_{13}$. In addition, the symbols which are treated as side information to (virtually) decode c, d, e are with indices $25, 26, 27$, respectively. Thus these remaining queries should 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 using 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, given in Table II.

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 $\{a_* + c_* + d_*, a_* + c_* + e_*, a_* + d_* + e_*\}$, are used to decode new symbols of a. Consider the first stage of round 3 queries to server 1. The new indices for the symbols of the demanded message a are $\{55, 56, 57\}$, since the first 54 symbols of a 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 together in a query, the indices of the other involved symbols are the same, i.e., in $\{a_{55} + c_{29} + d_{28}, b_{55}+c_{35}+d_{34}\}$, both a and b have index 55. When a and d appear together, the indices of the other involved symbols are both 29; see $\{a_{55}+c_{29}+d_{28}, a_{57}+d_{30}+e_{29}\}$. One can verify that the same structure holds for any two messages appearing together. It is also interesting to see that this phenomenon also exists in the coded caching scheme in [34], which is the core to preserve the privacy.

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

TABLE II: Round 2 of queries.

Among the remaining queries $\{a_*+b_*+c_*, a_*+b_*+d_*, a_*+b_*+e_*, c_*+d_*+e_*\}$, first consider $a_*+b_*+c_*$. 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 $b_{55} + c_{35} + d_{34}$ and $a_* + b_* + c_*$ both contain b and c, the indices of a and d should be the same, which is 34. To determine the index of b, since in the query $a_{55} + c_{29} + d_{28}$ containing both a and c, b should have the same index as d_{28}, which is 28. To determine the index of c, there has not been any query containing both a and b in this stage yet. Note that a_{34} and b_{28} have already been recovered

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

TABLE III: Rounds 3 and 4 of queries.

by the user from the previous transmissions. Hence, we use symbol c_{23} transmitted in the first round of queries to server 2, such that the whole sum $a_{34}+b_{28}+c_{23}$ could be recovered by the previous rounds and thus redundant, which could be removed later in Step 6 to reduce the transmissions. Let us then consider the indices of $a_*+b_*+d_*$. a_* should have the same index as c_{35} in $b_{55}+c_{35}+d_{34}$. b_* should have the same index as c_{29} in $a_{55}+c_{29}+d_{28}$. d_* should have the same index as c_{23} in $a_{34}+b_{28}+c_{23}$. Thus we fix $a_*+b_*+d_*=a_{35}+b_{29}+d_{23}$. Similarly, we can subsequently determine the indices of $a_*+b_*+e_*, c_*+d_*+e_*$ as $a_{36}+b_{30}+e_{23}, c_{57}+d_{56}+e_{55}$.

By this approach, Round 3 and 4 queries are obtained as shown in Table III.

So far, 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 will be proved later that by the proposed 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 [3]. 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 [3] to achieve the same amount of redundancy. Instead, we propose a new sign assignment approach resulting in this amount of redundancy while guaranteeing the decodability.

Step 5: Sign assignment. The messages in each query should be sorted based on the lexicographic order of the messages. In round 2, for each query with one symbol from $\{a,b,c\}$ and one symbol from $\{d,e\}$, a plus sign is used in between the symbols; for each other query in this round, a minus sign is used. For instance, the queries in the first stage of round 2 sent to Server 1 would be as Table IV.

Among these queries, the query $q_{10}=d_{27}-e_{26}$ can be written as a linear combination of the other queries, thus being redundant. To show this, suppose $d=a+b$ and $e=b+c$; one

$$
\begin{aligned}
q_1 &= a_{25}-c_{13}\\
q_2 &= a_{26}+d_{13}\\
q_3 &= a_{27}+e_{13}\\
q_4 &= b_{25}-c_{14}\\
q_5 &= b_{26}+d_{14}\\
q_6 &= b_{27}+e_{14}\\
q_7 &= a_{14}-b_{13}\\
q_8 &= c_{26}+d_{25}\\
q_9 &= c_{27}+e_{25}\\
q_{10} &= d_{27}-e_{26}
\end{aligned}
$$

TABLE IV: First stage of round 2 queries to Server 1 after sign assignment.

can check that the equation $q_{10}=q_3+q_6-q_5-q_8+q_7+q_1+q_4$ holds.

In general for sign assignment, each query is first divided into two parts. The first part contains symbols of independent messages and the second part symbols of dependent messages, which are called independent symbols and dependent symbols, respectively. So each query q is written as

$$q=(\text{independent symbols})\pm(\text{dependent symbols}), \quad (15)$$

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 (15), the sign assignment is called *structure plus*; when minus sign is used, it is called *structure minus*. Round 2 queries use structure plus, then round 3 uses minus, and round 4 again uses plus. This is the most non-trivial step in the proposed sign assignment. Note that besides alternating signs in each parenthesis in (15), we also alternate the signs between the two parenthesis in each query according to the round numbers. The latter sign alternating is needed to ensure the decodability of the scheme which will be proved in Appendix A. After these steps, each query is solely randomly multiplied by a $+1$ or -1, uniformly at random, referred to as switching random variables. This is to ensure the existence of mapping of queries for two different sets of demanded messages by a choice of $\{\sigma_i\}$ and these switching RVs, which is required in the privacy proof in Appendix B. We assume all to be $+1$ in the example. After the sign assignment, the queries follow Tables VI and VII. Notice that the multiplicative matrices $\mathbf{G}$ in the tables are due to the following step.

Step 6: Reducing Download. In the first stage of round 2, as pointed out in the previous step, among the queries $\{q_1,q_2,...,q_{10}\}$, q_{10} is redundant. On the other hand, since both a_{14} and b_{13} are downloaded in the first and second stages of round 1 from Server 2, $q_7=a_{14}-b_{13}$ is also redundant. However, it is not possible to simply delete this query since it jeopardizes the symmetry and consequently the privacy. Instead, we use a coding strategy as follows. Instead of sending the 10 queries $\{q_1,q_2,...,q_{10}\}$, the 8 queries $\mathbf{q}=\mathbf{G}_{8\times 10}[q_1,q_2,...,q_{10}]^T$ are sent, where $\mathbf{G}_{8\times 10}$ is an MDS matrix of size 8×10. By receiving all 8 queries in $\mathbf{q}$, the user is able to decode all non-redundant queries. Similarly, all queries are depicted in Tables V, VI, and VII.

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 4 (independence of the proposed rate of M)**.** *As shown in Fig. 2, the proposed rate is almost independent of M. This is due to the fact that in every stage, the queries only containing symbols of the $M-K$ dependent messages are redundant and their burden is removed in Step 6. On the other hand, part of the remaining queries include the ones containing two or more symbols from the demanded messages and the $M-K$ dependent messages. These are part of useless queries and their burden is also removed in Step 6 as in the example. The redundancy removement helps with eliminating the effect of $M-K$ dependent messages in the scheme, which has proved to be effective in Fig. 2. Note that for the original private computation problem in [3] (i.e., the case of $P=1$), it was proved that the optimal rate is absolutely independent of M, which is also due to the redundancy removement.*

Remark 5 (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 6 (Privacy)**.** *Intuitively, the privacy of the proposed scheme follows from the fact that the scheme yields symmetric 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 Appendix B that the symbols signs appeared in each query have a one to one mapping for different sets of demanded messages; keeping the demanded messages hidden from the viewpoint of each server.*

Remark 7 (Outline of the proposed scheme)**.** *After the initialization steps (Steps 1-3), the proposed scheme in Step 4 designs the queries similar to the delivery phase of coded caching in terms of designing the indices of symbols (whose detailed explanation will be provided in Lemma 1). Then using the sign assignment strategy in Step 5, we let some transmitted messages be linear combinations of others, such that this redundancy could be removed by using an MDS matrix in Step 6. As a result, the number of transmissions is reduced. Note that Step 2 is designed such that the number of side information queries needed is satisfied.*

Remark 8. *(Why $\alpha_5 = 0$) Note that in each informative query (i.e., the query which contains symbols from the demanded messages), there exists only one new symbol from all the demanded messages, which has not been decoded. In this example, since there are totally 5 messages, 2 of which are demanded, summation of 5 symbols (each from a different message) has two demanded symbols and cannot contribute to decoding any demanded symbols. This is the reason the scheme continues till round 4.*

Remark 9. *(Number of stages) The last round is round 4 with 1 stage. These queries are of the form $a_*+c_*+d_*+e_*$ where $c_*+d_*+e_*$ is treated as side information, or of the form $b_*+c_*+d_*+e_*$ where again $c_*+d_*+e_*$ is treated as side information, or of the form $a_*+b_*+\{c_*+d_*, c_*+e_*, d_*+e_*\}$ where $\{c_*+d_*, c_*+e_*, d_*+e_*\}$ are treated as side information. Based on this observation, a stage of round 4 needs 2 stages of round 3 and 1 stage of round 2, to get the side information. Similarly, for a stage of round 3, the queries are of the form $a_*+\{c_*+d_*, c_*+e_*, d_*+e_*\}$ where $\{c_*+d_*, c_*+e_*, d_*+e_*\}$ are treated as side information, or of the form $b_*+\{c_*+d_*, c_*+e_*, d_*+e_*\}$ where $\{c_*+d_*, c_*+e_*, d_*+e_*\}$ are treated as side information, or of the form $a_*+b_*+\{c_*, d_*, e_*\}$ where $\{c_*, d_*, e_*\}$ are treated as side information. Based on this observation, a stage of round 3 needs 2 stages of round 2 and 1 stage of round 1, to get the side information. By a similar argument, round 2 needs 2 stages of round 1 to get the side information. Considering all this together, the number of stages in each round is determined as $\alpha_5=0, \alpha_4=1, \alpha_3=2, \alpha_2=5, \alpha_1=12$. For instance, for the 12 stages of round 1, 10 of them are used as side information in round 2, which has 5 stages and each of which needs 2 stages of round 1 as side information; the remaining 2 are used in round 3 with 2 stages, since each of which needs 1 stage of round 1 as side information.*

V. New Proposed MM-PC Scheme: The General Case

In this section, following the main idea of the example in Section IV, we describe the general MM-PC scheme proposed in this paper. Note that each message is divided into L symbols. The j^{th} symbol of W_i is denoted by $W_i(j)$. The proofs of decodability and privacy of the proposed scheme are provided in Appendices A and B, respectively.

Step 1: Permutation and Relabeling. In this step, the symbols in each message are permuted by a single permutation function $\pi(\cdot)$ over $[L]$ and multiplied by the multiplicative variable $\sigma_i \in \{+1,-1\}$ for the symbol index $i \in [L]$. We denote the alternated message of W_m by u_m as follows.

$$u_m(i) := \sigma_i W_m(\pi(i)), m \in [M], i \in [L]. \tag{16}$$

Both the permutation function π and the multiplicative variables σ_i are uniformly and independently distributed. These functions are independent of message label $m \in [M]$.

Furthermore, we change the initial labeling of the messages such that the first P labels are the demanded messages; i.e., $(\theta_1, \theta_2, ..., \theta_P) = (1, 2, ..., P)$. We expand the new basis with $K-P$ more independent messages with the new labels from $P+1$ to K, and then label the others (which are the new dependent ones) from $K+1$ to M. Notice that this is possible with the assumption that the demanded messages are independent. This relabeling (or permutation on messages) is done privately by the user and unknown to the servers.

Step 2: Number of Stages. This step is inspired from the second MM-PIR scheme in [15] (for case $P \leq \frac{M}{2}$). The query structure to each server is split into $M-P+1$ *rounds*, where each round i contains the queries summing i different symbols. Each round may also be split into multiple *stages*. Each stage of round i queries contains all $\binom{M}{i}$ possible choices of messages; i.e., summations with the form $u_{j_1}(*)+u_{j_2}(*)+$

Round	Stage	Server 1	Server 2
round 1	stage 1	$\mathbf{G}_{3\times5}^{(1,1)} \times \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ e_1 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,1)} \times \begin{bmatrix} a_{13} \\ b_{13} \\ c_{13} \\ d_{13} \\ e_{13} \end{bmatrix}$
	stage 2	$\mathbf{G}_{3\times5}^{(1,2)} \times \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \\ e_2 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,2)} \times \begin{bmatrix} a_{14} \\ b_{14} \\ c_{14} \\ d_{14} \\ e_{14} \end{bmatrix}$
	stage 3	$\mathbf{G}_{3\times5}^{(1,3)} \times \begin{bmatrix} a_3 \\ b_3 \\ c_3 \\ d_3 \\ e_3 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,3)} \times \begin{bmatrix} a_{15} \\ b_{15} \\ c_{15} \\ d_{15} \\ e_{15} \end{bmatrix}$
	stage 4	$\mathbf{G}_{3\times5}^{(1,4)} \times \begin{bmatrix} a_4 \\ b_4 \\ c_4 \\ d_4 \\ e_4 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,4)} \times \begin{bmatrix} a_{16} \\ b_{16} \\ c_{16} \\ d_{16} \\ e_{16} \end{bmatrix}$
	stage 5	$\mathbf{G}_{3\times5}^{(1,5)} \times \begin{bmatrix} a_5 \\ b_5 \\ c_5 \\ d_5 \\ e_5 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,5)} \times \begin{bmatrix} a_{17} \\ b_{17} \\ c_{17} \\ d_{17} \\ e_{17} \end{bmatrix}$
	stage 6	$\mathbf{G}_{3\times5}^{(1,6)} \times \begin{bmatrix} a_6 \\ b_6 \\ c_6 \\ d_6 \\ e_6 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,6)} \times \begin{bmatrix} a_{18} \\ b_{18} \\ c_{18} \\ d_{18} \\ e_{18} \end{bmatrix}$
	stage 7	$\mathbf{G}_{3\times5}^{(1,7)} \times \begin{bmatrix} a_7 \\ b_7 \\ c_7 \\ d_7 \\ e_7 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,7)} \times \begin{bmatrix} a_{19} \\ b_{19} \\ c_{19} \\ d_{19} \\ e_{19} \end{bmatrix}$
	stage 8	$\mathbf{G}_{3\times5}^{(1,8)} \times \begin{bmatrix} a_8 \\ b_8 \\ c_8 \\ d_8 \\ e_8 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,8)} \times \begin{bmatrix} a_{20} \\ b_{20} \\ c_{20} \\ d_{20} \\ e_{20} \end{bmatrix}$
	stage 9	$\mathbf{G}_{3\times5}^{(1,9)} \times \begin{bmatrix} a_9 \\ b_9 \\ c_9 \\ d_9 \\ e_9 \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,9)} \times \begin{bmatrix} a_{21} \\ b_{21} \\ c_{21} \\ d_{21} \\ e_{21} \end{bmatrix}$
	stage 10	$\mathbf{G}_{3\times5}^{(1,10)} \times \begin{bmatrix} a_{10} \\ b_{10} \\ c_{10} \\ d_{10} \\ e_{10} \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,10)} \times \begin{bmatrix} a_{22} \\ b_{22} \\ c_{22} \\ d_{22} \\ e_{22} \end{bmatrix}$
	stage 11	$\mathbf{G}_{3\times5}^{(1,11)} \times \begin{bmatrix} a_{11} \\ b_{11} \\ c_{11} \\ d_{11} \\ e_{11} \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,11)} \times \begin{bmatrix} a_{23} \\ b_{23} \\ c_{23} \\ d_{23} \\ e_{23} \end{bmatrix}$
	stage 12	$\mathbf{G}_{3\times5}^{(1,12)} \times \begin{bmatrix} a_{12} \\ b_{12} \\ c_{12} \\ d_{12} \\ e_{12} \end{bmatrix}$	$\mathbf{G}_{3\times5}^{(1,12)} \times \begin{bmatrix} a_{24} \\ b_{24} \\ c_{24} \\ d_{24} \\ e_{24} \end{bmatrix}$

TABLE V: Round 1 of queries. Informative queries are denoted with black and side information queries with red.

Round	Stage	Server 1	Server 2
round 2	stage 1	$\mathbf{G}_{8\times10}^{(2,1)} \times \begin{bmatrix} a_{25}-c_{13} \\ a_{26}+d_{13} \\ a_{27}+e_{13} \\ b_{25}-c_{14} \\ b_{26}+d_{14} \\ b_{27}+e_{14} \\ a_{14}-b_{13} \\ c_{26}+d_{25} \\ c_{27}+e_{25} \\ d_{27}-e_{26} \end{bmatrix}$	$\mathbf{G}_{8\times10}^{(2,1)} \times \begin{bmatrix} a_{28}-c_1 \\ a_{29}+d_1 \\ a_{30}+e_1 \\ b_{28}-c_2 \\ a_{29}+d_2 \\ b_{30}+e_2 \\ a_2-b_1 \\ c_{29}+d_{28} \\ c_{30}+e_{28} \\ d_{30}-e_{29} \end{bmatrix}$
	stage 2	$\mathbf{G}_{8\times10}^{(2,2)} \times \begin{bmatrix} a_{31}-c_{15} \\ a_{32}+d_{15} \\ a_{33}+e_{15} \\ b_{31}-c_{16} \\ b_{32}+d_{16} \\ b_{33}+e_{16} \\ a_{16}-b_{15} \\ c_{32}+d_{31} \\ c_{33}+e_{31} \\ d_{33}-e_{32} \end{bmatrix}$	$\mathbf{G}_{8\times10}^{(2,2)} \times \begin{bmatrix} a_{34}-c_3 \\ a_{35}+d_3 \\ a_{36}+e_3 \\ b_{34}-c_4 \\ a_{35}+d_4 \\ b_{36}+e_4 \\ a_4-b_3 \\ c_{35}+d_{34} \\ c_{36}+e_{34} \\ d_{36}-e_{35} \end{bmatrix}$
	stage 3	$\mathbf{G}_{8\times10}^{(2,3)} \times \begin{bmatrix} a_{37}-c_{17} \\ a_{38}+d_{17} \\ a_{39}+e_{17} \\ b_{37}-c_{18} \\ b_{38}+d_{18} \\ b_{39}+e_{18} \\ a_{18}-b_{17} \\ c_{38}+d_{37} \\ c_{39}+e_{37} \\ d_{39}-e_{38} \end{bmatrix}$	$\mathbf{G}_{8\times10}^{(2,3)} \times \begin{bmatrix} a_{40}-c_5 \\ a_{41}+d_5 \\ a_{42}+e_5 \\ b_{40}-c_6 \\ a_{41}+d_6 \\ b_{42}+e_6 \\ a_6-b_5 \\ c_{41}+d_{40} \\ c_{42}+e_{40} \\ d_{42}-e_{41} \end{bmatrix}$
	stage 4	$\mathbf{G}_{8\times10}^{(2,4)} \times \begin{bmatrix} a_{43}-c_{19} \\ a_{44}+d_{19} \\ a_{45}+e_{19} \\ b_{43}-c_{20} \\ b_{44}+d_{20} \\ a_{20}+b_{19} \\ a_6-b_5 \\ c_{44}+d_{43} \\ c_{45}+e_{43} \\ d_{45}-e_{44} \end{bmatrix}$	$\mathbf{G}_{8\times10}^{(2,4)} \times \begin{bmatrix} a_{46}-c_7 \\ a_{47}+d_7 \\ a_{48}+e_7 \\ b_{46}-c_8 \\ a_{47}+d_8 \\ b_{48}+e_8 \\ a_8-b_7 \\ c_{47}+d_{46} \\ c_{48}+e_{46} \\ d_{48}-e_{47} \end{bmatrix}$
	stage 5	$\mathbf{G}_{8\times10}^{(2,5)} \times \begin{bmatrix} a_{49}-c_{21} \\ a_{50}+d_{21} \\ a_{51}+e_{21} \\ b_{49}-c_{22} \\ b_{50}+d_{22} \\ b_{51}+e_{22} \\ a_{22}-b_{21} \\ c_{50}+d_{49} \\ c_{51}+e_{49} \\ d_{51}-e_{50} \end{bmatrix}$	$\mathbf{G}_{8\times10}^{(2,5)} \times \begin{bmatrix} a_{52}-c_9 \\ a_{53}+d_9 \\ a_{54}+e_9 \\ b_{52}-c_{10} \\ a_{53}+d_{10} \\ b_{54}+e_{10} \\ a_{10}-b_9 \\ c_{53}+d_{52} \\ c_{54}+e_{52} \\ d_{54}-e_{53} \end{bmatrix}$

TABLE VI: Round 2 of queries. Informative queries, useless queries, and side information queries are denoted with black, blue, and red respectively.

Round	Stage	Server 1	Server 2
round 3	stage 1	$\mathbf{G}_{7\times10}^{(3,1)} \times \begin{bmatrix} a_{55}-c_{29}-d_{28} \\ a_{56}-c_{30}-e_{28} \\ a_{57}-d_{30}+e_{29} \\ b_{55}-c_{35}-d_{34} \\ b_{56}-c_{36}-e_{34} \\ b_{57}-d_{36}+e_{35} \\ a_{34}-b_{28}+c_{23} \\ a_{35}-b_{29}-d_{23} \\ a_{36}-b_{30}-e_{23} \\ c_{57}-d_{56}+e_{55} \end{bmatrix}$	$\mathbf{G}_{7\times10}^{(3,1)} \times \begin{bmatrix} a_{58}-c_{26}-d_{25} \\ a_{59}-c_{27}-e_{25} \\ a_{60}-d_{27}+e_{26} \\ b_{58}-c_{32}-d_{31} \\ b_{59}-c_{33}-e_{31} \\ b_{60}-d_{33}+e_{32} \\ a_{31}-b_{25}+c_{11} \\ a_{32}-b_{26}-d_{11} \\ a_{33}-b_{27}-e_{11} \\ c_{60}-d_{59}+e_{58} \end{bmatrix}$
	stage 2	$\mathbf{G}_{7\times10}^{(3,2)} \times \begin{bmatrix} a_{61}-c_{41}-d_{40} \\ a_{62}-c_{42}-e_{40} \\ a_{63}-d_{42}+e_{41} \\ b_{61}-c_{47}-d_{46} \\ b_{62}-c_{48}-e_{46} \\ b_{63}-d_{48}+e_{47} \\ a_{46}-b_{40}+c_{24} \\ a_{47}-b_{41}-d_{24} \\ a_{48}-b_{42}-e_{24} \\ c_{63}-d_{62}+e_{61} \end{bmatrix}$	$\mathbf{G}_{7\times10}^{(3,2)} \times \begin{bmatrix} a_{64}-c_{38}-d_{37} \\ a_{65}-c_{39}-e_{37} \\ a_{66}-d_{39}+e_{38} \\ b_{64}-c_{44}-d_{43} \\ b_{65}-c_{45}-e_{43} \\ b_{66}-d_{45}+e_{44} \\ a_{43}-b_{37}+c_{12} \\ a_{44}-b_{38}-d_{12} \\ a_{45}-b_{39}-e_{12} \\ c_{66}-d_{65}+e_{64} \end{bmatrix}$
round 4	stage 1	$\mathbf{G}_{2\times5}^{(3,2)} \times \begin{bmatrix} a_{67}-c_{60}+d_{59}-e_{58} \\ b_{67}-c_{66}+d_{65}-e_{64} \\ a_{64}-b_{58}+c_{53}+d_{52} \\ a_{65}-b_{59}+c_{54}+e_{52} \\ a_{66}-b_{60}+d_{54}-e_{53} \end{bmatrix}$	$\mathbf{G}_{2\times5}^{(3,2)} \times \begin{bmatrix} a_{68}-c_{57}+d_{56}-e_{55} \\ b_{68}-c_{63}+d_{62}-e_{61} \\ a_{61}-b_{55}+c_{50}+d_{49} \\ a_{62}-b_{56}+c_{51}+e_{49} \\ a_{63}-b_{57}+d_{51}-e_{50} \end{bmatrix}$

TABLE VII: Rounds 3 and 4 of queries. Informative queries, useless queries, and side information queries are denoted with black, blue, and red respectively.

$... + u_{j_i}(*), \forall \{j_1, j_2, ..., j_i\} \subset [M]$. The symbol indices $*$ will be carefully chosen, explained in Step 4. In each round, the number of stages will be determined as follows. Consider a stage of round i queries to server 1. The queries are partitioned based on the number of symbols from the demanded messages involved. For the queries containing only 1 symbol from the demanded messages, there are $\binom{P}{1} = P$ types; for each type, one stage of round $i-1$ is needed to provide the side information part. Note that these P stages of round $i-1$ (used for providing side information) are from the other $N-1$ servers, i.e., servers 2 to N, for the sake of privacy. Generally, for the queries containing $i_1 \in [\min\{i, P\}]$ symbols from the demanded messages, there are $\binom{P}{i_1}$ types; for each type, one stage of round $i - i_1$ is needed from the other $N-1$ servers to provide the side information part.

The number of stages in each round j queries to each server is denoted by α_j, for $j \in [M - P + 1]$. The number of stages of round j to servers 2 to N would be $(N-1)\alpha_j$. $\binom{P}{1}\alpha_{j+1}$ of these stages will be used as the side information in α_{j+1} stages of round $j+1$ queries to server 1. $\binom{P}{2}\alpha_{j+2}$ of these stages will be used as the side information in α_{j+2} stages of round $j+2$ queries to server 1, and so on, leading to the equation (14). Furthermore, as seen in the example, only queries containing one symbol from the demanded messages contribute to decoding new demanded symbols. Thus, after round $M - P + 1$, since each query would have at least two symbols from the demanded messages, the scheme continues til round $M - P + 1$; $\alpha_j = 0, \forall j \in [M - P + 2 : M]$.

After the general structure of the queries is set, the next step would be to determine which indices should be used for the symbols in each query.

Step 3: Initialization. In this step, the queries of round 1 (single symbols) are downloaded from the servers. Let $\texttt{new}(u_m)$ be a function that starting from $u_m(1)$, returns the next symbol index of u_m each time it is called, i.e., the first time the function $\texttt{new}(u_m)$ is called, it returns $u_m(1)$, next time it returns $u_m(2)$ and so on. Starting from server 1, the functions $\texttt{new}(u_1), \ldots, \texttt{new}(u_M)$ are called as the queries to the server. This is one stage of round 1 and should be repeated α_1 times in total for each server.

Next we determine the indices of symbols in the queries. Notice that the ultimate goal of index assignment, is to exploit the redundancy between messages and reduce the total number of queries, hence increasing the rate.

Step 4: Index assignment. The indexing structure follows the following lemma.

Lemma 1 (Index structure)**.** *In a stage in round i, for any set of $i-1$ messages (assumed to be $\{u_1, u_2, \ldots, u_{i-1}\}$) and any other two messages (assumed to be u_{i_1}, u_{i_2}), in the queries with the form $u_{i_1}(k_1) + u_1(*) + \cdots + u_{i-1}(*)$ and $u_{i_2}(k_2) + u_1(*) + \cdots + u_{i-1}(*)$, it should have $k_1 = k_2$ (i.e., the symbol indices of u_{i_1}, u_{i_2} in the two queries are the same.).*

The indexing structure in Lemma 1 is inspired from the delivery phase of the seminal coded caching scheme in [34].

Our objective is to design the index assignment satisfying Lemma 1. To accomplish this, we divide the queries in each stage into three groups: (1) informative queries, (2) side information queries, and (3) useless queries. The description of the design comes as follows.

4.1: Informative Queries. These queries are used to decode new symbols of demanded messages. Each informative query only contains one symbol from demanded messages, which is added to some side information obtained from the previous round, and thus can be decoded using this side information. Formally, these queries for round i are with the form $q_\gamma = u_\theta(*) + u_{j_1}(*) + ... + u_{j_{i-1}}(*)$ where $\theta \in [P]$ and $\{j_1, ..., j_{i-1}\} \subseteq [M] \setminus [P]$ and γ denotes the set of message indices, i.e. $\gamma = \{\theta, j_1, ..., j_{i-1}\}$. The part $u_{j_1}(*) + ... + u_{j_{i-1}}(*)$ is treated as side information directly obtained from some stage in round $i-1$ dedicated for the usage of side information for u_θ. The symbol $u_\theta(*)$ is a previously not-decoded symbol for u_θ, i.e. $\texttt{new}(u_\theta)$. So the queries involving u_θ is the set $\{q_{\theta \cup \gamma'} = \texttt{new}(u_\theta) + u_{j_1}(*) + ... + u_{j_{i-1}}(*) : \forall \gamma' = \{j_1, ..., j_{i-1}\} \subseteq [M] \setminus [P]\}$ where we choose γ' in a lexicographic order. By the structure of queries, in each stage of round i, $\binom{M-P}{i-1}$ new symbols of each demanded message is decoded, which equals the number of ways of choosing the set $\{j_1, ..., j_{i-1}\} \subseteq [M] \setminus [P]$. Note that for any given $\gamma' = \{j_1, ..., j_{i-1}\} \in [M] \setminus [P]$, in the set of queries $\{q_{\theta \cup \gamma'} = u_\theta(*) + u_{j_1}(*) + ... + u_{j_{i-1}}(*) : \forall \theta \in [P]\}$, all $u_\theta(*)$ where $\theta \in [P]$ have the same index, since for each u_θ the queries have been built on the lexicographic order of γ', consequently satisfying the index structure in Lemma 1.

Let us go back to the example in Section IV. In one stage of round 2, we first determine the query $a_* + c_*$, then $a_* + d_*$, and then $a_* + e_*$. These queries would be $a_{25} + c_{13}$, $a_{26} + d_{13}$, and $a_{27} + e_{13}$ for the first stage of round 2 queries to server 1, where a_{25}, a_{26}, a_{27} are new symbols of message a, and c_{13}, d_{13}, e_{13} have been downloaded symbols in round 1 treated as the side information in round 2.

4.2: Side Information Queries. These queries do not contain any symbols from demanded messages. Consider the query $q_\gamma = u_{j_1}(*) + \cdots + u_{j_i}(*)$ in round i where $\gamma = \{j_1, ..., j_i\} \in [M] \setminus [P]$. To determine the symbol index for u_k where $k \in \gamma$, by Lemma 1, the index of this symbol should be determined by any informative query in the same stage (determined in Step 4.1) containing symbols of messages $\gamma \setminus \{k\}$, $q_{\theta \cup \gamma \setminus \{k\}}$ for any $\theta \in [P]$; i.e., the index should be the same as the symbol index of the demanded message u_θ in $q_{\theta \cup \gamma \setminus \{k\}}$. By the definition, the number of the side information queries in a stage in round i is $\binom{M-P}{i}$.

Let us go back to the example in Section IV. In the first stage of round 2, the symbol indices in $c_* + d_*$ are determined based on the informative queries $a_{25} + c_{13}$ and $a_{26} + d_{13}$. So c is added to a symbol with index 25 and d is added to a symbol with index 26; i.e., the resulting query is $c_{26} + d_{25}$.

4.3: Useless Queries. These queries contain more than one symbol from the demanded messages. Starting with the queries containing two demanded messages, consider the query $q_\gamma = u_{\theta_1}(*) + u_{\theta_2}(*) + u_{j_1}(*) + ... + u_{j_{i-2}}(*)$ where $\gamma = \{\theta_1, \theta_2, j_1, \ldots, j_{i-2}\}$, and $\theta_1, \theta_2 \in [P], j_1, ...j_{i-2} \in [M] \setminus [P]$. The part $u_{j_1}(*) + ... + u_{j_{i-2}}(*)$ is a side information obtained from a stage in round $i-2$. Therefore, it remains to determine the indices of u_{θ_1} and u_{θ_2}. To determine the index of u_{θ_1}, based on Lemma 1, the index of this symbol

should be determined by any informative query in the same stage (determined in Step 4.1) containing symbols of messages $\{u_{\theta_2}, u_{j_1}, ..., u_{j_{i-2}}\}$, $q_{\gamma'}$ where $\gamma' = \theta_1' \cup \{\theta_2, j_1, \dots, j_{i-2}\}$ for any $\theta_1' \in [M] \setminus [P]$; i.e., the index should be the same as the symbol index of $u_{\theta_1'}(*)$ in $q_{\gamma'}$. It is important to note that since $u_{\theta_1'}(*)$ comes from a side information query in a stage of round $i-1$, u_{θ_1} with the same symbol index has also appeared in the same stage in an informative query, and thus has already been decoded there. Consequently, these queries cannot contribute to decoding new symbols for demanded messages, nor serve as side information. We observe that to determine symbol indices containing two symbols from the demanded messages, queries containing one are used. Similarly, to determine symbol indices for queries containing three symbols from demanded messages, queries containing two are used, with a similar process explained. This process continues until all queries in this group have been indexed.

Let us go back to the example in Section IV. In the first stage of round 1, to determine the indices in $a_* + b_*$, we need to check the queries $a_{25} + c_{13}$ and $b_{25} + c_{14}$. So a is added to a symbol with index 13 and b to a symbol with index 14; i.e., the resulting query is $a_{14} + b_{13}$.

As a result, by Step 4, the indices of all the symbols are determined. The next step would be to assign the signs ($+1$ or -1) to symbols in the queries, such that there would be some queries being linear combinations of other queries.

Step 5: Sign assignment. The sign assignment step, from round 2 to the last round, includes two sub-steps: (1) choosing between structure plus or minus and (2) performing random sign switching, which are described as follows.

5.1: Structure Plus/Minus. Each query is first divided into two parts. The first part contains symbols from independent messages and the second part symbols from dependent messages. So each query q is written as

$$q = (\text{independent symbols}) \pm (\text{dependent symbols}). \quad (17)$$

The sign $+$ is referred to as *structure plus* and the sign $-$ is referred to as *structure minus*. In round 2, a structure plus is used in each query. The structure is successively switched for the next rounds, i.e. for round 3, a structure minus is used in each query; for round 4, a structure plus is used in each query; and so on. Additionally, in each parenthesis, after ordering the symbols based on the lexicographic order of the corresponding messages, the first symbol is assigned by a plus sign and this successively alternates until the last symbol in the parenthesis. In other words, if the independent symbols in (17) are $u_{i_1}(*), u_{i_2}(*), \dots, u_{i_j}(*)$ where $i_1 < i_2 < \cdots < i_j$, then (independent symbols) in (17) should be

$$(\text{independent symbols}) = (u_{i_1}(*) - u_{i_2}(*) + u_{i_3}(*) - \cdots).$$

Similarly, if the dependent symbols in (17) are $u_{k_1}(*), u_{k_2}(*), \dots, u_{k_j}(*)$ where $k_1 < k_2 < \cdots < k_j$, then (dependent symbols) in (17) should be

$$(\text{dependent symbols}) = (u_{k_1}(*) - u_{k_2}(*) + u_{k_3}(*) - \cdots).$$

5.2: Random Sign Switching: In this step, each query solely is multiplied by $+1$ or -1, uniformly and independently at random.

Remark 10. *As studied in [33] in the cache-aided scalar linear function retrieval problem, in order to reduce the load in the delivery phase of a caching system in which each user requests a linear combination of messages, it is needed that symbols get multiplied by a minus or a plus based on certain rules. For sign assignment, we are inspired from the sign assignment in [33]. Particularly, the caching scheme in [33] always uses the structure plus between independent symbols and dependent symbols. This is natural since they have one stage (and also only one round) of delivery. However, since we have multiple delivery stages and rounds, which are are inter-connected; i.e. a side information query in one stage is used in another stage, to ensure the decodability of the scheme, we have to use the plus and minus structures alternatively in rounds.*

Lemma 2. *By the end of Step 5, each stage of round i has $\binom{M-K}{i}$ linearly redundant queries from the total $\binom{M}{i}$ queries, and can be written as linear combinations of the others. Linearly redundant queries are those which do not contain any symbols from independent messages.*

The proof is given in Appendix D.

By the end of Step 5, we summarize that in each stage there are two disjoint sets of redundant queries: the set of useless queries and the set of linearly redundant queries. Furthermore,

- The useless queries are redundant since they are the summation of some side information and some symbols of demanded messages which are all previously decoded.
- The set of linearly redundant queries by Lemma 2 are among the side information queries, which are some linear combinations of all remaining queries.

Hence, we further reduce the amount of download summations by removing the redundancy. However, removing these queries directly from the set of queries jeopardizes privacy. Step 6 introduces a way to reduce download while preserving privacy.

For a stage of round $i \in [M-P+1]$, the number of informative queries, side information queries, and useless queries are $n_{iq}^{(i)} = P\binom{M-P}{i-1}, n_{sq}^{(i)} = \binom{M-P}{i}, n_{uq}^{(i)} = \binom{M}{i} - n_{iq}^{(i)} - n_{sq}^{(i)}$, respectively. The number of linearly redundant queries is $n_{rq}^{(i)} = \binom{M-K}{i}$.

Step 6: Reducing Download. For each round i and each stage s, if the queries are $q_1, \dots, q_{\binom{M}{i}}$, we denote $\mathbf{q}^{(i,s)} = [q_1, \dots, q_{\binom{M}{i}}]^T$. We multiply $\mathbf{q}^{(i,s)}$ on the left by the MDS matrix $\mathbf{G}^{(i,s)}$ of size $r \times \binom{M}{i}$, where r is defined as $r = \binom{M}{i} - n_{uq}^{(i)} - n_{rq}^{(i)} = P\binom{M-P}{i-1} + \binom{M-P}{i} - \binom{M-K}{i}$, to reach the final set of queries in this stage as the elements of $\mathbf{q}_f^{(i,s)}$,

$$\mathbf{q}_f^{(i,s)} := \mathbf{G}^{(i,s)}_{r \times \binom{M}{i}} \mathbf{q}^{(i,s)}. \quad (18)$$

This is done for all rounds i and stages s.

The reason we can decode all $\binom{M}{i}$ queries in $\mathbf{q}^{(i,s)}$ by $\mathbf{q}_f^{(i,s)}$ is as follows. We first partition $\mathbf{q}^{(i,s)}$ into three parts as

$$\mathbf{q}^{(i,s)} = \begin{bmatrix} \mathbf{q}_1^{(i,s)} \\ \mathbf{q}_2^{(i,s)} \\ \mathbf{q}_3^{(i,s)} \end{bmatrix}, \quad (19)$$

where $\mathbf{q}_3^{(i,s)}, \mathbf{q}_2^{(i,s)}, \mathbf{q}_1^{(i,s)}$ represent linearly redundant queries, useless queries, and other queries, respectively. Since $\mathbf{q}_3^{(i,s)}$ is a linear combination of the other two, there exists a full rank matrix $\mathbf{G}'$ such that

$$\mathbf{q}^{(i,s)} = \begin{bmatrix} \mathbf{q}_1^{(i,s)} \\ \mathbf{q}_2^{(i,s)} \\ \mathbf{q}_3^{(i,s)} \end{bmatrix} = \mathbf{G}' \begin{bmatrix} \mathbf{q}_1^{(i,s)} \\ \mathbf{q}_2^{(i,s)} \end{bmatrix}. \tag{20}$$

Thus, (18) turns into

$$\mathbf{q}_f^{(i,s)} = \mathbf{G}^* \begin{bmatrix} \mathbf{q}_1^{(i,s)} \\ \mathbf{q}_2^{(i,s)} \end{bmatrix}, \tag{21}$$

for some full rank matrix $\mathbf{G}^* = \mathbf{G}'^{(i,s)}_{r\times\binom{M}{i}}\mathbf{G}'$. Since the queries in $\mathbf{q}_2^{(i,s)}$ have already been decoded from the previous rounds, together with $\mathbf{q}_f^{(i,s)}$ we can decode $\mathbf{q}_1^{(i,s)}$.

Step 7: Shuffling. Finally, we shuffle the order of queries sent to each server and also, shuffle the order of the messages appearing in each query. The shufflings are uniformly and independently at random. This is to prevent servers from guessing any orders between messages and queries.

Decodability and rate. Intuitively, the decodability simply follows since the informative queries are composed of the desired symbol added to some previously downloaded side information; the most-non-trivial step to guarantee this is the alternative structure plus and structure minus across different rounds. The overall rate is computed as the ratio of the number of informative queries to all queries. The formal proof of the decodability and rate computation is given in Appendix A.

Privacy. Intuitively, privacy is satisfied since the queries are symmetric with respect to each message through the index assignment structure. Besides, the sign assignment step does not reveal the identity of the demanded messages since there is a mapping of symbol signs for different demand scenarios with the help of random variables involved, including the multiplicative factors in Step 1 and sign switching variables in Step 5.2. As a consequence, all possible symbol signs for different demand scenarios will be equally likely. Furthermore, the MDS coding step trivially does not jeopardise privacy. The formal proof of the privacy is given in Appendix B.

VI. Conclusion

In this paper, we studied the multi-message private computation problem which is an extension to the PC problem of [3] and the MM-PIR of [15]. Our design is based on breaking the scheme into multiple rounds and stages such that round i corresponds to queries in the form of summations of i different symbols. By designing the index and sign of each symbol involved, we were able to reduce the amount of downloaded summations since some of the queries are linear combinations of the others. Furthermore, to use this redundancy while preserving privacy, we used an MDS coding method so that each server cannot distinguish between the redundant and non-redundant queries. Numerical evaluations demonstrated that the rate of the proposed scheme has significant improvements over the baseline scheme for a wide range of system parameters, thus inheriting the order-optimality of the baseline scheme within a multiplicative factor of 2. It is also important to point out that the rate of the proposed scheme has very little dependence on M, as suggested by Fig. 2, while this is not the case for the baseline scheme. This is important since we expect that as long as K is fixed, changing only the number of possible linear combinations should not affect the rate for an order optimal scheme. We observe the same behaviour for the optimal PC scheme in [3].

On-going works include deriving the converse bound specifically for the MMPC problem and designing new MMPC schemes with low subpacketization level.

Appendix A
Proof of Decodability and Rate Calculation

By the end of Step 4 (index assignment), it is straightforward to decode the new symbols of demanded messages, since these new symbols only exist in informative queries which are built by the addition of these symbols to some already known side information. But after Step 5 (sign assignment), some symbol signs alter to a minus. Since in each stage, the informative and useless queries are build up using some side information from earlier rounds, we should check if after the sign assignment step, these side information queries remain consistent regarding the symbol signs. For the sake of simplicity, we assume other than the first P labels, the other $K-P$ independent messages are labeled from from $P+1$ to K. Also for the sake of simplicity, we denote symbols just by the message letter and not using $(*)$ in front of it.

In round i, for some informative query $q = u_\theta + q_{si}$ where $\theta \in [P]$, the side information part q_{si} within this query should remain consistent on symbol signs compared to the corresponding query in round $i-1$ after sign assignment. Without loss of generality, assume we use structure plus for round $i-1$ and structure minus for round i. Also assume from the $i-1$ symbols in q_{si}, v of them are symbols of independent messages; i.e., $q_{si} = u_{j_1} + ... + u_{j_v} + u_{j_{v+1}} + ... + u_{j_{i-1}}$, where $\{j_1, ...j_v\} \subset [P+1:K], j_{v+1}, ...j_{i-1} \subset [K+1:M]$. If v is even, then after sign assignment for query q_{si} in round $i-1$, u_{j_1} would have a plus sign and u_{j_2} a minus sign and so on, until a minus sign for u_{j_v}. Since structure plus is used for this round, $u_{j_{v+1}}$ starts with a plus sign and the other signs follow the alternating structure; leading to $q'_{si} = u_{j_1} - u_{j_2} + ... - u_{j_v} + u_{j_{v+1}} - u_{j_{v+2}} + ... \pm u_{j_{i-1}}$, where q'_{si} is q_{si} after sign assignment. In sign assignment for the query q in round i, u_θ starts with a plus sign, u_{j_1} would have a minus sign, u_{j_2} a plus sign up until u_{j_v} with a plus sign. Then, since structure minus is used in this round, $u_{j_{v+1}}$ would start with a minus sign and so on; leading to $q' = u_\theta - u_{j_1} + u_{j_2} - ... + u_{j_v} - u_{j_{v+1}} + u_{j_{v+2}} - ... \pm u_{j_{i-1}}$, where q' is q after sign assignment. It is evident that $q' = u_\theta - q'_{si}$, and therefore, the signs are consistent after sign assignment and q'_{si} can be cancelled out to decode for u_θ. We can similarly prove the case for v being odd. This completes the proof of consistency for informative queries.

We should prove the consistency for useless queries too. Consider the useless query $q = u_{\theta_{l_1}} + ... + u_{\theta_{l_n}} + q_{si}$ in round i with n symbols from the demanded messages, i.e. $\{\theta_{l_1}, ..., \theta_{l_n}\} \subset [P]$ and the side information part has v symbols from demanded messages, i.e. $q_{si} = u_{j_1} + ... +$

$u_{j_v} + u_{j_{v+1}} + ... + u_{j_{i-n}}$, where $\{j_1, ...j_v\} \in [P+1 : K], j_{v+1}, ...j_{i-n} \in [K+1 : M]$. Assume without loss of generality, in round $i-n$ structure plus is used for sign assignment. For the case v is odd, after sign assignment for query q_{si}, u_{j_1} would have a plus sign, u_{j_2} a minus and so on, until u_{j_v} with a plus sign. $u_{j_{v+1}}$ would have a plus sign and the rest change their signs alternatively, leading to $q'_{si} = u_{j_1} - ... + u_{j_v} + u_{j_{v+1}} - ... \pm u_{j_{i-n}}$, where q'_{si} is q_{si} after sign assignment. There are two cases for n, both of which need to be checked. For the case n is even, for round i structure plus will be used again. After sign assignment for the query q, $u_{\theta_{l_1}}$ would have a plus sign, $u_{\theta_{l_2}}$ a minus sign and so on, up to $u_{\theta_{l_n}}$ with a minus sign. Also, u_{j_1} would have a plus, u_{j_2} a minus, up until u_{j_v} with a plus. Furthermore, $u_{j_{v+1}}$ would have a plus sign and the rest change their signs alternatively, leading to $q' = u_{\theta_{l_1}} - ... - u_{\theta_{l_n}} + u_{j_1} - ... + u_{j_v} + u_{j_{v+1}} - ... \pm u_{j_{i-n}}$, where q' is q after sign assignment. Thus, it is evident that $q' = u_{\theta_{l_1}} - u_{\theta_{l_2}} + ... - u_{\theta_{l_n}} + q'_{si}$. Therefore, again the signs remain consistent after sign assignment. For the case n is odd, similarly it will be resulted that $q' = u_{\theta_{l_1}} - u_{\theta_{l_2}} + ... + u_{\theta_{l_n}} - q'_{si}$, where again the consistency is evident. For the case v is even, one can verify the sign consistency similarly. Therefore, we have proved the consistency of signs after sign assignment. Notice that in the proof, for convenience, we have assumed the sign switching variables in Step 5.2 are all 1 and this does not jeopardize the generality, since only the relative symbol signs are important. This completes the proof of decodability.

To calculate the rate, we first calculate the message length L. To do so, we count the number of informative queries corresponding to each demanded message, since these are the only queries that generate new indices, which as calculated in Step 4.1, equals $\binom{M-P}{i-1}$ for a stage in round i. Furthermore, the number of stages in round i is α_i which follows (14). Therefore, collectively from all servers, for each message, $N\alpha_i\binom{M-P}{i-1}$ new symbols appear in round i. Therefore,

$$L = N \sum_{i=1}^{M-P+1} \alpha_i \binom{M-P}{i-1}. \tag{22}$$

Next, we calculate the total download D from all servers. Based on Step 6, in a stage in round i, a total of $r = \binom{M-P}{i} - \binom{M-K}{i} + \binom{P}{1}\binom{M-P}{i-1}$ symbols is downloaded. Therefore,

$$D = N \sum_{i=1}^{M-P+1} \alpha_i \left(\binom{M-P}{i} - \binom{M-K}{i} + P\binom{M-P}{i-1} \right). \tag{23}$$

For the rate defined in (7), using (22) and (23), we get the rate in Theorem 3.

Appendix B
Proof of Privacy

To prove privacy, we must show no matter the choice of $\mathcal{I}$, the realization of the queries for each server has the same probability space. Notice that after the index assignment step, the queries to each server are completely symmetrical. This is because the queries are partitioned to multiple stages, and in each stage of round i, all the possible $\binom{M}{i}$ types of queries appear. Besides, the indexing structure in Lemma 1 is also symmetrical from the viewpoint of each message, as discussed earlier. To proceed, we first state the following lemma.

Lemma 3. *In a stage of queries to one server, the symbol indices appearing are disjoint from those of other stages in the same server.*

Proof. We go through all 3 types of queries in a stage. Notice that since the side information queries to a server duplicate the new symbol indices of demanded messages in the same stage, and since these new indices do not appear in the same server in any other stage by definition, these queries have completely disjoint indices compared to other stages in the same server. Furthermore, the side information parts of informative and useless queries have also disjoint indices, since these parts are duplicated from queries to other servers and are used only once in queries to each server, so they do not appear anywhere else in the same server. Additionally, the symbols of demanded messages in useless queries duplicate the new indices of demanded messages in the same server, indicating they do not appear twice in queries to the same server. □

With Lemma 3 and the symmetry of indices from the perspective of each message, it is readily concluded that for any two choices of demanded messages $\mathcal{I}_1$ and $\mathcal{I}_2$ where $\mathcal{I}_1 \neq \mathcal{I}_2$, the indices of symbols in queries to one server have a one to one mapping by a choice of permutation function π.

The proposed scheme has two permutations: one on symbol indices and the other on message indices, where the latter is referred to as *relabeling* as stated in the first step of the scheme. So far we have shown that the permutation function π on symbol indices preserves privacy. To complete the proof, it only remains to show that the sign assignment step does not jeopardize the symmetry of the queries, in the sense that it does not reveal the private relabeling of the messages, otherwise some information on the requested messages would be leaked. We indicate this by showing that the signs of symbols in queries to one server for two choices of demanded messages $\mathcal{I}_1$ and $\mathcal{I}_2$ where $\mathcal{I}_1 \neq \mathcal{I}_2$, have an one to one mapping by a particular choice of multiplicative variables $\sigma_i, i \in [L]$ and sign switching variables in Step 5.2. Remember that these variables are chosen by the user and private to the server.

We now introduce an algorithm, by which the sign mapping from $\mathcal{I}_1$ to $\mathcal{I}_2$ will be possible. By each step, the necessary explanations are immediately followed. Notice that since we have proved the one to one mapping of indices, we do not present the indices for ease of understanding.

We indicate the multiplicative variables in the setting $\mathcal{I}_1$ with σ_is and in the setting $\mathcal{I}_2$ with σ'_is. Based on a fixed choice of σ_is, we choose the values of σ'_is such that the symbol signs in corresponding queries match. The algorithm is as follows.

Step 1. *Choose the messages with randomly chosen labels $j_1, j_2, ..., j_i$. Compare the query containing these messages when $\mathcal{I}_2$, i.e. $q_1^{(2)} = \pm\sigma'_{j_2 j_3 ... j_i} W_{j_1} \pm \sigma'_{j_1 j_3 ... j_i} W_{j_2} \pm ... \pm \sigma'_{j_1 j_2 ... j_{i-1}} W_{j_i}$, to the query when $\mathcal{I}_1$, i.e. $q_1^{(1)} = \pm\sigma_{j_2 j_3 ... j_i} W_{j_1} \pm \sigma_{j_1 j_3 ... j_i} W_{j_2} \pm ... \pm \sigma_{j_1 j_2 ... j_{i-1}} W_{j_i}$. Simply choose σ'_is in $q_1^{(2)}$ such that the sign of each symbol matches with the corresponding one in $q_1^{(1)}$.*

Step 2. *All the variables σ'_i that were fixed in Step 1, appear also in some other queries, but not together. Go through all these queries, and fix other σ'_is involved relative to the other already-fixed variable in Step 1.*

Consider the query containing message labels $j_0, j_2, ..., j_i$, i.e. $q_2^{(2)} = \pm\sigma'_{j_2 j_3 ... j_i} W_{j_0} \pm \sigma'_{j_0 j_3 ... j_i} W_{j_2} \pm ... \pm \sigma'_{j_0 j_2 ... j_{i-1}} W_{j_i}$. The already-fixed variable $\sigma'_{j_2 j_3 ... j_i}$ appears in this query too. Compare this query to its corresponding one when $\mathcal{I}_1$, i.e. $q_2^{(1)} = \pm\sigma_{j_2 j_3 ... j_i} W_{j_0} \pm \sigma_{j_0 j_3 ... j_i} W_{j_2} \pm ... \pm \sigma_{j_0 j_2 ... j_{i-1}} W_{j_i}$. Fix the other variables $\sigma'_{j_0 j_3 ... j_i}, ..., \sigma'_{j_0 j_2 ... j_{i-1}}$, relative to the already-fixed $\sigma'_{j_2 j_3 ... j_i}$ such that either $q_2^{(2)} = q_2^{(1)}$ or $q_2^{(2)} = -q_2^{(1)}$.

After fixing these queries (fixing the σ'_is inside), one has the concern whether the fixed σ'_is are consistent among the other queries they appear in simultaneously. For example, take the queries $q_1^{(2)}$ and $q_2^{(2)}$ fixed in Steps 1 and 2. In $q_1^{(2)}$ the message labels $j_1, j_2, ..., j_i$ and in $q_2^{(2)}$ the message labels $j_0, j_2, ..., j_i$ appear. In these two queries, the variables $\sigma'_{j_1 j_3 ... j_i}$ and $\sigma'_{j_0 j_3 ... j_i}$ are fixed. We should check in the query containing both of these together, i.e. containing message labels $j_0, j_1, j_3, ..., j_i$, whether their relative values remains consistent. In general we should prove, and this will also be needed in the following steps, whether any two variables of σ'_is, when fixed in two different queries, maintain a correct relative value concerning in the query in which both of them appear. This is proved in the following lemma.

Lemma 4. *The already-fixed variables $\sigma'_{j_1 j_3 ... j_i}$ and $\sigma'_{j_0 j_3 ... j_i}$, fixed in queries $q_1^{(2)}$ and $q_2^{(2)}$, maintain a correct relative sign when they appear together in another query.*

Proof. We prove the lemma for one setting of the labels j_0, j_1, j_2 for each case of $\mathcal{I}_1$ and $\mathcal{I}_2$, since all other ones is proved similarly. For simplicity, assume that every $\sigma_i = 1$ when $\mathcal{I}_1$. Assume when $\mathcal{I}_1$, the messages with labels j_0, j_1, j_2 are all in the independent set with the ordering $j_0 < j_1 < j_2$. Additionally, assume among independent messages in $j_1, j_2, ..., j_i$, there is an odd number of messages between j_1 and j_2. Moreover among independent messages in $j_0, j_2, ..., j_i$, there is again an odd number of messages between j_0 and j_2. Based on this setting, after sign assignment we have the following queries for the three set of labels $\{j_1, j_2, ..., j_i\}$, $\{j_0, j_2, ..., j_i\}$, and $\{j_0, j_1, j_3, ..., j_i\}$ respectively,

$$q_1^{(1)} = W_{j_1} + W_{j_2} \pm \cdots \tag{24}$$
$$q_2^{(1)} = W_{j_0} + W_{j_2} \pm \cdots \tag{25}$$
$$q_3^{(1)} = W_{j_0} - W_{j_1} \pm \cdots \tag{26}$$

For $\mathcal{I}_2$, we consider the case where labels j_1 and j_2 are among the independent messages which have odd number of independent messages in between based on the ordering among $j_1, j_2, ..., j_i$. Additionally we assume j_0 is among dependent messages. With this setting, if we assume the relative sign between W_{j_0} and W_{j_2} in $q_2^{(2)}$ is minus, then,

$$q_1^{(2)} = \sigma'_{j_2 j_3 ... j_i} W_{j_1} + \sigma'_{j_1 j_3 ... j_i} W_{j_2} \pm \cdots \tag{27}$$
$$q_2^{(2)} = \sigma'_{j_2 j_3 ... j_i} W_{j_0} - \sigma'_{j_0 j_3 ... j_i} W_{j_2} \pm \cdots \tag{28}$$
$$q_3^{(2)} = \sigma'_{j_1 j_3 ... j_i} W_{j_0} + \sigma'_{j_0 j_3 ... j_i} W_{j_1} \pm \cdots \tag{29}$$

To fix the variables in $q_1^{(2)}$ and $q_2^{(2)}$, we should set $\sigma'_{j_2 j_3 ... j_i} = \sigma'_{j_1 j_3 ... j_i} = 1$ and $\sigma'_{j_0 j_3 ... j_i} = -1$. This leads to $q_3^{(2)} = W_{j_0} - W_{j_1} \pm ...$, which as can be seen, automatically matches with $q_3^{(1)}$. So the relative signs remain consistent and the lemma is proved. □

Remark 11. *The reason why only the relative values of σ'_is are important, is because of the sign switching variables of Step 5.2 in the scheme. When the relative signs of symbols are correct, to match these signs between two corresponding queries of different labelings $\mathcal{I}_1$ and $\mathcal{I}_2$, we only need to multiply the whole query with a -1 or a $+1$.*

In Step 2, we fixed all the queries that are within 1 message distance from the first randomly chosen query $q_1^{(2)}$; meaning the queries in Step 2 have $i-1$ messages in common with that of $q_1^{(2)}$ and are only different in 1 message. In Step 3, we fix the queries with distance 2 from $q_1^{(2)}$.

Step 3. *Consider all the queries with distance 2 from $q_1^{(2)}$. Fix the variables σ'_is within these queries relative to the already-fixed ones in the first two steps.*

Consider the query containing messages with labels $j'_1, j'_2, j_3, ..., j_i$, which is in distance 2 from $q_1^{(2)}$. The variables $\sigma'_{j'_2, j_3, ..., j_i}$ and $\sigma'_{j'_1, j_3, ..., j_i}$ have been already fixed in Step 2 of the algorithm, and they both appear in the mentioned query. We should prove their relative value remains correct in this new query. This is proved in the following lemma.

Lemma 5. *The already-fixed values of σ'_is within queries in previous steps, maintain the correct relative values in Step 3.*

Proof. Take two queries containing the message labels $j'_1, j_2, ..., j_i$ and $j'_2, j_2, ..., j_i$. These queries are fixed in Step 2, so the values for $\sigma'_{j'_2, j_3, ..., j_i}$ and $\sigma'_{j'_1, j_3, ..., j_i}$, are already fixed in these two queries. Exactly like the proof in Lemma 4, the relative signs of these variables remain correct in the query with labels $j'_1, j'_2, j_3, ..., j_i$, which contains both variables. □

The rest of the algorithm is evident, as follows.

Step 4. *Each time increase the distance of queries from $q_1^{(2)}$ by one, and fix the not-yet-fixed σ'_is within these queries. Continue this process until the last step, where the distance is i. Then, all the queries will be exhausted and fixed.*

The correctness of relative signs of the already-fixed variables in each step is proved similar to the previous steps.

It is immediately resulted by our algorithm, that if the mapping of symbol signs from the setting $\mathcal{I}_2$ to $\mathcal{I}_1$ is done by the values $\{\sigma_i'^*\}$ and sign switching variables in vector $\mathbf{s}$, then there would be another set of answers $\{-\sigma_i'^*\}$ and $-\mathbf{s}$ and there exists no other set of answers. This proves that the mapping from all possible setting to the setting $\mathcal{I}_1$, is uniformly random, thus hiding the private labeling in Step 1 of the scheme. This completes the proof the privacy.

$$
\begin{aligned}
q_1 &= a_{55} - c_{29} - d_{28} \\
q_2 &= a_{56} - c_{30} - e_{28} \\
q_3 &= a_{57} - d_{30} + e_{29} \\
q_4 &= b_{55} - c_{35} - d_{34} \\
q_5 &= b_{56} - c_{36} - e_{34} \\
q_6 &= b_{57} - d_{36} + e_{35} \\
q_7 &= a_{34} - b_{28} + c_{23} \\
q_8 &= a_{35} - b_{29} - d_{23} \\
q_9 &= a_{36} - b_{30} - e_{23} \\
q_{10} &= c_{57} - d_{56} + e_{55}
\end{aligned}
$$

TABLE VIII: First stage of round 3 queries to Server 1 for the demand set $\{a, b\}$.

$$
\begin{aligned}
&\sigma_{55} c_{\pi(55)} - \sigma_{29} e_{\pi(29)} - \sigma_{28} a_{\pi(28)} \\
&\sigma_{56} c_{\pi(56)} - \sigma_{30} e_{\pi(30)} - \sigma_{28} b_{\pi(28)} \\
&\sigma_{57} c_{\pi(57)} - \sigma_{30} a_{\pi(30)} + \sigma_{29} b_{\pi(29)} \\
&\sigma_{55} d_{\pi(55)} - \sigma_{35} e_{\pi(35)} - \sigma_{34} a_{\pi(34)} \\
&\sigma_{56} d_{\pi(56)} - \sigma_{36} e_{\pi(36)} - \sigma_{34} b_{\pi(34)} \\
&\sigma_{57} d_{\pi(57)} - \sigma_{36} a_{\pi(36)} + \sigma_{35} b_{\pi(35)} \\
&\sigma_{34} c_{\pi(34)} - \sigma_{28} d_{\pi(28)} + \sigma_{23} e_{\pi(23)} \\
&\sigma_{35} c_{\pi(35)} - \sigma_{29} d_{\pi(29)} - \sigma_{23} a_{\pi(23)} \\
&\sigma_{36} c_{\pi(36)} - \sigma_{30} d_{\pi(30)} - \sigma_{23} b_{\pi(23)} \\
&\sigma_{57} e_{\pi(57)} - \sigma_{56} a_{\pi(56)} + \sigma_{55} b_{\pi(55)}
\end{aligned}
$$

TABLE IX: First stage of round 3 queries to Server 1 for demand set $\{c, d\}$.

A. Example for Privacy Using the Proposed Algorithm

Let us return to the example in section IV and illustrate the proof of the privacy. Consider the case that the demands are $\{a, b\}$, the lexicographic order of the messages is as (a, b, c, d, e), the permutation function π is the identity permutation which maps each input to itself, the multiplicative variables are all 1; $\sigma_i = 1, \forall i \in [68]$, and the sign switching variables in Step 5.2 of the scheme are all 1. For this case, the first stage of queries in round 3 for Server 1 (before Step 6 of the scheme) is as in Table VIII.

Now consider another case where the demands are $\{c, d\}$ and the lexicographic order of the messages is as (c, d, e, a, b). For this case, again consider the first stage of queries in round 3 for Server 1. Before Step 6 of the scheme, the queries would be as in Table IX.

Our goal is to depict the one-to-one mapping of these two sets of queries with a choice of permutation function π and the multiplicative variables σ_is. To do so, we first reorder the queries in Table IX so that the corresponding lines have symbols from the same messages, making it easier to find the mapping. The reordered queries would be as Table X.

Now the same lines in Tables X and VIII are corresponding queries. By the following index mapping, the indices in corresponding queries would be the same.

$$
\begin{aligned}
q'_1 &= -\sigma_{23} a_{\pi(23)} + \sigma_{35} c_{\pi(35)} - \sigma_{29} d_{\pi(29)} \\
q'_2 &= -\sigma_{28} a_{\pi(28)} + \sigma_{55} c_{\pi(55)} - \sigma_{29} e_{\pi(29)} \\
q'_3 &= -\sigma_{34} a_{\pi(34)} + \sigma_{55} d_{\pi(55)} - \sigma_{35} e_{\pi(35)} \\
q'_4 &= -\sigma_{23} b_{\pi(23)} + \sigma_{36} c_{\pi(36)} - \sigma_{30} d_{\pi(30)} \\
q'_5 &= -\sigma_{28} b_{\pi(28)} + \sigma_{56} c_{\pi(56)} - \sigma_{30} e_{\pi(30)} \\
q'_6 &= -\sigma_{34} b_{\pi(34)} + \sigma_{56} d_{\pi(56)} - \sigma_{36} e_{\pi(36)} \\
q'_7 &= -\sigma_{30} a_{\pi(30)} + \sigma_{29} b_{\pi(29)} + \sigma_{57} c_{\pi(57)} \\
q'_8 &= -\sigma_{36} a_{\pi(36)} + \sigma_{35} b_{\pi(35)} + \sigma_{57} d_{\pi(57)} \\
q'_9 &= -\sigma_{56} a_{\pi(56)} + \sigma_{55} b_{\pi(55)} + \sigma_{57} e_{\pi(57)} \\
q'_{10} &= \sigma_{34} c_{\pi(34)} - \sigma_{28} d_{\pi(28)} + \sigma_{23} e_{\pi(23)}
\end{aligned}
$$

TABLE X: First stage of round 3 queries to Server 1 for demand set $\{c, d\}$ after reordering.

$$
\begin{aligned}
&\pi(23) = 55, \pi(28) = 56, \pi(34) = 57, \\
&\pi(29) = 28, \pi(35) = 29, \pi(55) = 30, \\
&\pi(30) = 34, \pi(36) = 35, \pi(56) = 36.
\end{aligned} \tag{30}
$$

Now that the indices are the same, and it remains to choose the σ_is such that the corresponding queries are either equal or equal with a -1 factor. As the algorithm suggests in Step 1, we first compare q_1 to q'_1. It is obvious that if $\sigma_{23} = -1, \sigma_{35} = -1, \sigma_{29} = +1$, then $q'_1 = q_1$.

Following Step 2 of the algorithm, we compare q'_4 to q_4. Since $\sigma_{23} = -1$, we set the other σ_is in q'_4 such that $q'_4 = q_4$, which leads to $\sigma_{36} = -1, \sigma_{30} = +1$. Next we compare q'_{10} to q_{10}. Since $\sigma_{23} = -1$, we choose $\sigma_{34} = -1, \sigma_{28} = -1$ such that $q'_{10} = -q_{10}$. After that we compare q'_3 to q_3, and since $\sigma_{35} = -1$, we choose $\sigma_{34} = -1, \sigma_{55} = -1$ which leads $q'_3 = q_3$. Comparing q'_8 to q_8 when $\sigma_{35} = -1$, we set $\sigma_{36} = -1, \sigma_{57} = -1$ to get $q'_8 = q_8$. Then we compare q'_2 to q_2 and since $\sigma_{29} = +1$, we set $\sigma_{28} = -1, \sigma_{55} = -1$ so that $q'_2 = q_2$. In the end of Step 2 we compare q'_7 to q_7. Since $\sigma_{29} = +1$, we choose $\sigma_{30} = +1, \sigma_{57} = -1$ which leads $q'_7 = -q_7$. It is noteworthy to see that in Step 2 each of the fixed σ_is has actually been fixed two times, which both match and this is a result of Lemma 4.

There are three remaining queries q'_5, q'_6, q'_9 for Step 3 of the algorithm. By setting $\sigma_{56} = -1$, we get $q'_5 = q_5, q'_6 = q_6, q'_9 = q_9$. At the end of Step 3, there are no σ_is left to set.

In the end, if we denote the sign switching variables in Step 5.2 of the scheme for the 10 queries $q'_1, ..., q'_{10}$ by $s_1, ..., s_{10}$, then by choosing $s_7 = -1, s_{10} = -1$ and the other ones equal to 1, we get that $q'_i = q_i, \forall i \in [10]$. Naturally if we multiply all fixed σ_is and s_is by -1, we get another set of answers that leads to $q'_i = q_i, \forall i \in [10]$. In general for any two sets of demands, we would have one choice of the mapping function π and two choices for the random variables such that the queries are the same.

APPENDIX C
PROOF OF THEOREM 2

To prove Theorem 2, we first note that the capacity of the MM-PIR problem in [15] for K total messages is an upper bound (i.e., converse bound) to our problem, since this setting assumes independency among all messages and the MM-PC problem allows for requesting not only messages themselves, but also their linear combinations. For the case $P \leq \frac{K}{2}$, the upper bound for the MM-PIR problem would be $R^\star \leq R_u = \frac{1-\frac{1}{N}}{1-(\frac{1}{N})^{\lfloor \frac{K}{P} \rfloor}}$. Note that the achieved rate in (8) is no less than $\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^K}$, which is achieved by using the PC scheme in [3] P times. Thus

$$
\frac{R_u}{R_1} \leq \frac{\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^{\lfloor \frac{K}{P} \rfloor}}}{\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^K}} = \frac{1-(\frac{1}{N})^K}{1-(\frac{1}{N})^{\lfloor \frac{K}{P} \rfloor}} \leq \frac{1}{1-\frac{1}{N}} \leq 2. \tag{31}
$$

For the case $P \geq \frac{K}{2}$, the capacity of MM-PIR follows $R_u = \frac{1}{1+\frac{K-P}{PN}}$. Thus

$$\frac{R_u}{R_1} \leq \frac{\frac{1}{1+\frac{K-P}{PN}}}{\frac{1-\frac{1}{N}}{1-(\frac{1}{N})^K}} \leq \frac{1}{1-\frac{1}{N}} \leq 2. \tag{32}$$

Appendix D
Proof of Lemma 2

We first point out that the structure of the queries in each stage, up until the end of Step 4 (index assignment), is exactly like the structure of the multicast messages in the delivery phase of the MAN coded caching scheme with M files and M users in which every user demands a different file; thus all the files are requested. To restate the index structure in Lemma 1, take a stage in round i and choose any $i-1$ messages. The set of queries with these messages have the same index for the other symbol involved in the query. This is the exact same structure as in the delivery phase of the MAN scheme when $t = i-1$, where each multicast message includes $t+1$ users.

In [33], the authors show that when some of the demanded files are linear combinations of the others, by carefully designing the signs of each symbol in the delivery phase, some of the multicast messages are linear combinations of the other ones, and thus redundant. In their paper, the users requesting independent messages are called *leaders*, and the other ones *non-leaders*. Therefore in our scheme, the independent messages correspond to the leaders, and the dependent ones to the non-leaders. In [33, Appendix B] they show using the structure plus in sign assignment, the multicast messages which do not include any leaders, are redundant and can be derived by other multicast messages. This is the first part of the proof.

On the other hand, in a stage, we can take a slight modification on the composition of the multicast messages in [33, Eq. 54], where the sign between the required blocks by the leaders and the non-leaders is changed from $+1$ (structure plus) to -1 (structure minus) such that the new composition of $X_{\mathcal{S}}$ becomes

$$\begin{aligned} X_{\mathcal{S}} = &\sum_{i\in[|\mathcal{L}_{\mathcal{S}}|]} (-1)^{i-1} B_{\mathcal{L}_{\mathcal{S}}(i),\mathcal{S}\setminus\{\mathcal{L}_{\mathcal{S}}(i)\}} \\ &- \sum_{j\in[|\mathcal{N}_{\mathcal{S}}|]} (-1)^{j-1} B_{\mathcal{N}_{\mathcal{S}}(j),\mathcal{S}\setminus\{\mathcal{N}_{\mathcal{S}}(j)\}}. \end{aligned} \tag{33}$$

By the new multicast message composition in (33), we can still prove the [33, Eq. (57a)] holds, which refers to the redundancy of some multicast messages, but with sightly modified decoding coefficients

$$\beta_{\mathcal{A},\mathcal{S}} = (-1)^{1+\mathrm{Tot}(\overline{\mathrm{Ind}}_{\mathcal{S}})+|\mathcal{S}\setminus\mathcal{A}|}\det(\mathbb{D}'_{\mathcal{A}\setminus\mathcal{S},\mathcal{L}_{\mathcal{S}}}). \tag{34}$$

The proof of [33, Eq. (57a)] with new multicast message composition in (33) and decoding coefficients in (34) directly follows the same steps as in [33, Appendix B], and thus we do not repeat it. This proves the same redundancy exists with the structure minus of sign assignment. Notice that the sign switching variables in Step 5.2 clearly does not affect the redundancy. This completes the proof of the theorem.

References

[1] A. Gholami, K. Wan, T. Jahani-Nezhad, H. Sun, M. Ji, and G. Caire, "On multi-message private computation," in *2024 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024, pp. 945–950.

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

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

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

[5] 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.

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] Y. Yakimenka, H.-Y. Lin, and E. Rosnes, "On the capacity of private monomial computation," *arXiv preprint arXiv:2001.06320*, 2020.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] N. Esmati, A. Heidarzadeh, and A. Sprintson, "Private linear transformation: The joint privacy case," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2125–2130.

[20] ——, "Private linear transformation: The individual privacy case," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2131–2136.

[21] F. Kazemi and A. Sprintson, "Multi-server private linear transformation with joint privacy," in *2021 XVII International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*, 2021, pp. 182–187.

[22] N. Esmati, A. Heidarzadeh, and A. Sprintson, "Multi-server private linear computation with joint and individual privacy guarantees," in *2021 XVII International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*, 2021, pp. 1–6.

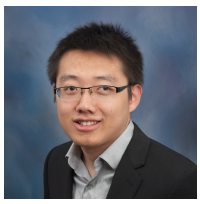
[23] W.-T. Chang and R. Tandon, "On the upload versus download cost for secure and private matrix multiplication," in *2019 IEEE Information Theory Workshop (ITW)*, 2019, pp. 1–5.

[24] M. Kim, H. Yang, and J. Lee, "Private coded matrix multiplication," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1434–1443, 2020.
[25] H. Akbari-Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *IEEE Transactions on Information Theory*, vol. 67, no. 4, pp. 2379–2398, 2021.
[26] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.
[27] J. Li and C. Hollanti, "Private and secure distributed matrix multiplication schemes for replicated or mds-coded servers," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 659–669, 2022.
[28] Q. Yu and A. S. Avestimehr, "Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 59–72, 2021.
[29] J. Zhu and S. Li, "A systematic approach towards efficient private matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 257–274, 2022.
[30] H. Yang, S. Hong, and J. Lee, "Private and secure coded computation in straggler-exploiting distributed matrix multiplication," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2137–2142.
[31] J. Zhu, L. Li, X. Tang, and P. Deng, "Private multiple linear computation: A flexible communication-computation tradeoff," *arXiv:2404.09165*, Apr. 2024.
[32] N. B. Shah, K. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 856–860.
[33] 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.
[34] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on information theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

Ali Gholami (Student Member, IEEE) received the B.Sc. degree from University of Tehran, Iran, in 2016, and the M.Sc. degree from Sharif University of Technology, Iran, in 2018, both in electrical engineering. He is currently pursuing the Ph.D. degree with the Communications and Information Theory (CommIT) group, Technical University of Berlin, Germany. His research interests include information theory, coding theory, and privacy.

Kai Wan (S '15 – M '18) received the B.E. degree in Optoelectronics from Huazhong University of Science and Technology, China, in 2012, the M.Sc. and Ph.D. degrees in Communications from Université Paris-Saclay, France, in 2014 and 2018. From 2018 to 2022, he was a post-doctoral researcher with the Communications and Information Theory Chair (CommIT) at Technische Universität Berlin, Berlin, Germany. He is now a Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology. His research interests include information theory, coding techniques, and their applications on coded caching, index coding, distributed storage, distributed computing, wireless communications, privacy and security. He received the Best Young Scientist Award in the 8th International Conference on Computer and Communication Systems, 2023. He has served as an Associate Editor of IEEE Transactions on Communications from Mar. 2024, and of IEEE Communications Letters from Aug. 2021.

Tayyebeh Jahani-Nezhad received her B.Sc. degree in Electrical Engineering and M.Sc. degree in Communication Systems from Isfahan University of Technology, Iran, in 2015 and 2017, respectively. She earned her Ph.D. in Communication Systems from Sharif University of Technology, Iran, in 2022. She is currently a Postdoctoral Researcher at the Communications and Information Theory Chair (CommIT), Technische Universität Berlin, Germany. Her research focuses on developing efficient and secure distributed learning frameworks, particularly in coded distributed computing and federated learning.



Hua Sun received the B.E. degree in Communications Engineering from Beijing University of Posts and Telecommunications, China, in 2011, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from University of California Irvine, USA, in 2013 and 2017, respectively. He is an Associate Professor in the Department of Electrical Engineering at the University of North Texas, USA. His research interests include information theory and its applications to communications, privacy, security, and storage.

Dr. Sun is a recipient of the NSF CAREER award in 2021, the UNT College of Engineering Junior Faculty Research Award in 2021, and the UNT College of Engineering Distinguished Faculty Fellowship in 2023. His co-authored papers received the IEEE Jack Keil Wolf ISIT Student Paper Award in 2016, an IEEE GLOBECOM Best Paper Award in 2016, and the 2020-2021 IEEE Data Storage Best Student Paper Award. He serves as Associate Editor for IEEE Transactions on Information Theory since 2025.

Mingyue Ji (Member, IEEE) received the Ph.D. degree from the Ming Hsieh Department of Electrical and Computer Engineering at the University of Southern California in 2015, where he received the USC Annenberg Fellowship from 2010 to 2014. He subsequently was a Staff II System Design Scientist with Broadcom Inc. from 2015 to 2016. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at the University of Florida. His research interests span a broad spectrum, including cloud and edge computing, distributed machine learning, and 5G and beyond wireless communications, networking, and sensing. Mingyue Ji's research activities cover fundamental theory study, algorithm design and analysis, and practical system implementation and experimentation. He received the NSF CAREER Award in 2022, the IEEE Communications Society Leonard G. Abraham Prize for the Best IEEE Journal on Selected Areas in Communications (JSAC) Paper in 2019, the Best Paper Awards at 2021 IEEE GLOBECOM Conference, 2015 IEEE ICC Conference and 2024 IEEE ISICN Conference, the Best Student Paper Award at 2010 IEEE European Wireless Conference, the 2022 Outstanding ECE Teaching Award and the 2023 Outstanding ECE Research Award at the University of Utah. He has been serving as Associate Editors for IEEE Transactions on Information Theory since 2022 and IEEE Transactions on Communications since 2020.

Giuseppe Caire (S '92 – M '94 – SM '03 – F '05) was born in Torino in 1965. He received a B.Sc. in Electrical Engineering from Politecnico di Torino in 1990, an M.Sc. in Electrical Engineering from Princeton University in 1992, and a Ph.D. from Politecnico di Torino in 1994. He has been a post-doctoral research fellow with the European Space Agency (ESTEC, Noordwijk, The Netherlands) in 1994-1995, Assistant Professor in Telecommunications at the Politecnico di Torino, Associate Professor at the University of Parma, Italy, Professor with the Department of Mobile Communications at the Eurecom Institute, Sophia-Antipolis, France, a Professor of Electrical Engineering with the Viterbi School of Engineering, University of Southern California, Los Angeles, and he is currently an Alexander von Humboldt Professor with the Faculty of Electrical Engineering and Computer Science at the Technical University of Berlin, Germany.

He received the Jack Neubauer Best System Paper Award from the IEEE Vehicular Technology Society in 2003, the IEEE Communications Society and Information Theory Society Joint Paper Award in 2004 and in 2011, the Okawa Research Award in 2006, the Alexander von Humboldt Professorship in 2014, the Vodafone Innovation Prize in 2015, an ERC Advanced Grant in 2018, the Leonard G. Abraham Prize for best IEEE JSAC paper in 2019, the IEEE Communications Society Edwin Howard Armstrong Achievement Award in 2020, the 2021 Leibniz Prize of the German National Science Foundation (DFG), and the CTTC Technical Achievement Award of the IEEE Communications Society in 2023. Giuseppe Caire is a Fellow of IEEE since 2005. He has served in the Board of Governors of the IEEE Information Theory Society from 2004 to 2007, and as officer from 2008 to 2013. He was President of the IEEE Information Theory Society in 2011. His main research interests are in the field of communications theory, information theory, channel and source coding with particular focus on wireless communications.