The Capacity of Private Computation

Hua Sun
Department of Electrical Engineering
University of North Texas, Denton, TX 76203
hua.sun@unt.edu

Syed A. Jafar
Center for Pervasive Communications and Computing (CPCC)
University of California Irvine, Irvine, CA 92697
syed@uci.edu

***Abstract*—We introduce the problem of private computation, comprised of N distributed and non-colluding servers, K datasets, and a user who wants to compute a function of the datasets privately, i.e., without revealing which function he wants to compute to any individual server. This private computation problem is a strict generalization of the private information retrieval (PIR) problem, by expanding the PIR message set (which consists of only independent messages) to also include functions of those messages. The capacity of private computation, C, is defined as the maximum number of bits of the desired function that can be retrieved per bit of total download from all servers. We characterize the capacity of an elemental private computation setting, with $N = 2$ servers and $K = 2$ datasets that are replicated at each server, for linear computations. Surprisingly, the capacity, $C = 2/3$, matches the capacity of PIR with $N = 2$ servers and $K = 2$ messages. Thus, allowing arbitrary linear computations does not reduce the communication rate compared to pure dataset retrieval. The same insight is shown to hold at the opposite extreme where the number of datasets $K \to \infty$, the number of servers N can be arbitrary, and arbitrary (including non-linear) computations are allowed.**

I. Introduction

Motivated by privacy concerns in distributed computing applications, we introduce the private computation (PC) problem, where a user wishes to privately compute a function of datasets that are stored at distributed servers. Specifically, K datasets are stored at N non-colluding servers, and a user wishes to compute a function on these datasets. A private computation scheme allows the user to compute his desired function, while revealing no information to any individual server about the identity of the desired function. The achievable rate of a private computation scheme is the ratio of the number of bits of the desired function that the user is able to retrieve, to the total number of bits downloaded from all servers. The capacity of private computation is the supremum of achievable rates.

The private computation problem is a strict generalization of the private information retrieval (PIR) problem, where one of the K datasets is desired by the user, i.e., the function to be computed simply returns the desired dataset. The capacity was characterized recently for PIR in [1] and for several of its variants in [2]–[7]. In the PIR setting, the datasets are called messages and all messages are independent. Private computation may also be viewed as PIR with *dependent* messages, where each possible function that may be desired by a user is interpreted as a dependent message, i.e., a message whose value depends on other messages.

Our main result is the capacity characterization of an elemental instance of the problem of private computation, where a user wishes to compute arbitrary linear combinations of $K = 2$ independent datasets (messages), replicated at $N = 2$ servers. Note that if the user can only choose one of $M = 2$ linear combinations that are linearly independent, then the setting is equivalent to the PIR problem with $K = 2$ messages and $N = 2$ servers. From [1], we know that the capacity of PIR in this setting is equal to $2/3$. Surprisingly, we show that even if the user wishes to compute arbitrary linear combinations of the two datasets, the capacity of private computation remains $2/3$, i.e., in terms of capacity, arbitrary linear computation incurs no additional penalty. The same insight is shown to hold at the opposite extreme where the number of datasets $K \to \infty$, the number of servers N can be arbitrary, and arbitrary (including, possibly non-linear) computations are allowed. While the asymptotic result is relatively straightforward, the elemental setting requires sophisticated structure in the queries in order to optimally exploit message dependencies. Specifically, the private computation scheme utilizes an optimized symbol index structure, and a sophisticated assignment of signs ('+' or '−') to each symbol in order to optimally exploit the linear dependencies.

Notation: For integers $Z_1, Z_2, Z_1 \leq Z_2$, we use the compact notation $[Z_1 : Z_2] = \{Z_1, Z_1+1, \cdots, Z_2\}$. The notation $X \sim Y$ is used to indicate that X and Y are identically distributed. For sets S_1, S_2, we define S_1/S_2 as the set of elements that are in S_1 and not in S_2. For $n \in \{1, 2\}$ we define $\bar{n}$ as the complement of n, i.e., $\bar{n} = 1$ if $n = 2$ and $\bar{n} = 2$ if $n = 1$.

II. Problem Statement and Definitions

Consider the private computation problem with $N = 2$ servers and $K = 2$ datasets. We will assume that the datasets are replicated at both servers, that the servers do not collude, and that the functions to be computed are linear combinations of the messages. We will focus primarily on this basic setting which opens the door to numerous other open problems through various generalizations, e.g., $K > 2$ datasets, coded storage instead of replication, $N > 2$ servers, some of which may collude, symmetric privacy requirements, non-linear functions, etc.

The two datasets, denoted by $W_{d_1}, W_{d_2} \in \mathbb{F}_p^{L\times 1}$, are each comprised of L i.i.d. uniform symbols from a finite field $\mathbb{F}_p$. In p-ary units,

$$H(W_{d_1}) \quad = \quad H(W_{d_2}) = L, \tag{1}$$

$$H(W_{d_1}, W_{d_2}) \quad = \quad H(W_{d_1}) + H(W_{d_2}). \tag{2}$$

A linear combination of these datasets is represented as a dependent message,

$$W_m = \alpha_m W_{d_1} + \beta_m W_{d_2}, m \in [1:M] \tag{3}$$

where α_m, β_m are two constants from $\mathbb{F}_p$, and '+' represents element-wise addition over $\mathbb{F}_p$. The case $M = 1$ is trivial, so we consider $M \geq 2$. Without loss of generality, the vectors (α_m, β_m) are assumed to be pairwise linearly independent.

There are $N = 2$ servers and each server stores both datasets W_{d_1}, W_{d_2}. A user privately generates $\theta \in [1:M]$ and wishes to compute (retrieve) W_θ while keeping θ a secret from each server. Depending on θ, there are M strategies that the user could employ to privately compute his desired function. For example, if $\theta = m$, then in order to compute W_m, the user employs $N = 2$ queries, $Q_1^{[m]}$ and $Q_2^{[m]}$. Since the queries are determined by the user with no knowledge of the realizations of the messages, the queries must be independent of the messages,[1]

$$\forall m \in [1:M], \quad I(W_1, \cdots, W_M; Q_1^{[m]}, Q_2^{[m]}) = 0. \tag{4}$$

The user sends $Q_1^{[m]}$ to the first server and $Q_2^{[m]}$ to the second server. Upon receiving $Q_n^{[m]}, n \in [1:2]$, the n-th server generates an answering string $A_n^{[m]}$, which is a function of $Q_n^{[m]}$ and the data stored (i.e., all the messages),

$$\forall m \in [1:M], n \in [1:2], H(A_n^{[m]} | Q_n^{[m]}, W_1, \cdots, W_M) = 0.$$

Each server returns to the user its answer $A_n^{[m]}$. From all the information that is now available to the user $(A_1^{[m]}, A_2^{[m]}, Q_1^{[m]}, Q_2^{[m]})$, the user decodes the desired message W_m according to a decoding rule that is specified by the private computation scheme. Let P_e denote the probability of error achieved with the specified decoding rule.

To protect the user's privacy, the M strategies must be indistinguishable (identically distributed) from the perspective of each server, i.e., the following privacy constraint must be satisfied $\forall n \in [1:2], \forall m \in [1:M]$,

$$(Q_n^{[1]}, A_n^{[1]}, W_1, \cdots, W_M) \sim (Q_n^{[m]}, A_n^{[m]}, W_1, \cdots, W_M). \tag{5}$$

The PC *rate* characterizes how many bits of desired information are computed per downloaded bit, and is defined as follows.

$$R \triangleq \frac{L}{D} \tag{6}$$

where D is the expected value (over random queries) of the total number of bits downloaded by the user from both servers.

A rate R is said to be ϵ-error achievable if there exists a sequence of private computation schemes, indexed by L, each of rate greater than or equal to R, for which $P_e \to 0$ as $L \to \infty$. Note that for such a sequence of private computation schemes, from Fano's inequality, we have

$$\text{[Correctness]} \quad H(W_m | A_1^{[m]}, A_2^{[m]}, Q_1^{[m]}, Q_2^{[m]}) = o(L) \tag{7}$$

where any function of L, say $f(L)$, is said to be $o(L)$ if $\lim_{L\to\infty} f(L)/L = 0$. The supremum of ϵ-error achievable rates is called the capacity C.

III. Capacity of Private Computation

Theorem 1 states our main result.

Theorem 1: For the private computation problem where a user wishes to privately retrieve one of M arbitrary linear combinations of $K = 2$ independent datasets from $N = 2$ servers, the capacity is $C = 2/3$.

When $M = 2$, the problem reduces to the PIR problem with $N = 2$ servers and $K = 2$ messages, for which the capacity is $2/3$ [1]. Adding more computation requirements $M > 2$ can not help (surprisingly it does not hurt either), so the converse of Theorem 1 is implied. We only need to prove the achievability, which is presented in Section IV.

It is quite surprising that increasing the number of messages by including arbitrary linear combinations of K datasets does not reduce capacity in the elemental setting of $K = 2$ and $N = 2$ servers. A natural question then is whether this insight holds more broadly. Remarkably, the insight is also true at the other extreme, where the number of datasets is large ($K \to \infty$) and the number of servers is arbitrary. It turns out that in this case, again the capacity of private computation is equal to the capacity of PIR. This supplemental result is rather straightforward and is stated in the following theorem.

Theorem 2: For the private computation problem with K independent datasets, W_k, $k \in [1:K]$, $H(W_k) = L$, arbitrary N servers and $M - K$ arbitrary (possibly non-linear) dependent messages, W_m, $m \in [K+1:M]$, $H(W_m \mid W_k, k \in [1:K]) = 0$, $H(W_m) \leq L$, if $K \to \infty$, then the capacity of private computation $C \to 1 - 1/N$, which is the capacity of PIR with $K \to \infty$ messages and N servers.

Proof: For Theorem 2, the achievability is identical to the symmetric PIR[2] scheme of Theorem 1 in [3], where the M functions are viewed as the messages in the symmetric PIR problem and common randomness is not used. The dependence of the messages has no impact on privacy or correctness of that scheme. The converse follows from the converse of regular PIR [1] because restricting the message set to $W_k, k \in [1:K]$ cannot reduce capacity. ■

IV. The Achievable Scheme

The private computation scheme needed for Theorem 1 builds upon and significantly generalizes the capacity achieving PIR scheme presented in [1], [8]. If we ignore the dependence of the messages in the private computation problem and directly use the PIR scheme (capacity achieving for independent messages) in [1], the rate achieved

[1] Since $M \geq 2$ and the functions are linearly independent, the message sets (W_{d_1}, W_{d_2}) and $(W_1, W_2, \cdots, W_M)$ are invertible functions of each other, so, e.g., conditioning on one is the same as conditioning on the other.

[2] In fact, Theorem 2 extends immediately to the symmetric private computation problem, where the user is prohibited from learning anything beyond the desired function.

is $\left(1+1/2+1/4+\cdots+1/2^{M-1}\right)^{-1} = \frac{2^{M-1}}{2^M-1}$, which is strictly less than $2/3$ (independent of M), the capacity of private computation. To optimally exploit the dependence of the messages, we start with the original PIR scheme of [1] and incorporate two new ideas. For ease of reference, let us denote the original PIR scheme of [1] as $PIR1$.

(1) **Index assignment:** Additional structure is required from symbol indices within the queries because dependence only exists across message symbols associated with the same index. This requirement yields a new PIR scheme, that we will denote as $PIR2$. If the messages are independent, then in terms of downloads $PIR2$ is as efficient as $PIR1$, i.e., they are both capacity achieving schemes. However, $PIR2$ is more efficient in terms of uploads than $PIR1$.[3]

(2) **Sign assignment:** The index structure of $PIR2$ seems essential to accommodate dependent messages. By itself, however, it is not sufficient. For example, the queries in both $PIR1$ and $PIR2$ are comprised of *sums* of symbols. Depending on the form of message dependencies, more sophisticated forms of combining symbols within queries may be needed. For our present purpose, with linear message dependencies, we will need both sums and differences. To this end, we need to carefully assign a 'sign' ('+' or '−') to each symbol. The sign assignment produces the optimal private computation scheme, denoted PC, for Theorem 1.

To present these schemes, we need to introduce the following notation. Let π represent a permutation over $[1:L]$. For all $m \in [1:M], i \in [1:L]$ let

$$u_m(i) \quad = \quad \sigma_i W_m(\pi(i)) \tag{8}$$

Thus, $W_m(\pi(i))$ are the symbols from message W_m, permuted by π, and $u_m(i)$ are the corresponding signed versions obtained by scaling with $\sigma_i \in \{+1,-1\}$. Since both m and i are indices in $u_m(i)$, if there is a potential for confusion, we will refer to m as the 'message index' and i as the 'symbol index'. Note that the same permutation is applied to all messages, and the same sign variable σ_i is applied to symbols from different messages that have the same symbol index. Both π and σ_i are generated privately, independently and uniformly by the user such that they are not known to the servers.

We will refer to the message W_m equivalently as the message u_m. To illustrate the key ideas we will use the special $M=4$ setting as our running example in this work.

Example A: Suppose the $M=4$ functions that we wish to compute are the following.

$$W_1 = W_{d_1}$$
$$W_2 = W_{d_2}$$
$$W_3 = \alpha_3 W_{d_1} + \beta_3 W_{d_2}$$
$$W_4 = \alpha_4 W_{d_1} + \beta_4 W_{d_2} \tag{9}$$

Each message consists of $L=16$ symbols from $\mathbb{F}_p$. The specialized setting allows us to use a simpler notation as follows.

$$(a_i, b_i, c_i, d_i) = (u_1(i), u_2(i), u_3(i), u_4(i))$$

The notation is simpler because we only have symbol indices. Message indices are not necessary in this toy setting because a different letter is used for each message.

[3]Based on our ongoing work we believe that the index structure of $PIR2$ (with an additional step required to minimize redundancy in the permutation π, see [1]) requires the smallest upload of any capacity achieving scheme for PIR.

We will start with the query structure of the PIR scheme, which we will modify using the two principles outlined earlier, to obtain the private computation scheme. First we explain the index assignment step.

A. Index Assignment: PIR2

In this section, we introduce the $PIR2$ scheme, built upon $PIR1$ by an index assignment process. The index assignments are necessary because unlike $PIR1$ where independent permutations are applied to symbols from each message, in $PIR2$ the same permutation is applied to symbols from every message. For ease of exposition, we will first illustrate the index assignment process through the $M=4$ example, and then present the general algorithm for arbitrary M. Since we do not use sign assignments in $PIR2$, the σ_i are redundant for this scheme. Without loss of generality, the reader may assume $\sigma_i = 1$ for all i for $PIR2$.

1) $M=4$ **Example***:* Suppose the desired message is W_1, i.e., $\theta=1$. Recall the query structure of $PIR1$, where we have left some of the indices of undesired symbols undetermined.

$\theta = 1$

Server 1	Server 2
a_1, b_1, c_1, d_1	a_2, b_2, c_2, d_2
$a_3 + b_2$	$a_6 + b_1$
$a_4 + c_2$	$a_7 + c_1$
$a_5 + d_2$	$a_8 + d_1$
$b_* + c_*$	$b_* + c_*$
$b_* + d_*$	$b_* + d_*$
$c_* + d_*$	$c_* + d_*$
$a_9 + b_* + c_*$	$a_{12} + b_* + c_*$
$a_{10} + b_* + d_*$	$a_{13} + b_* + d_*$
$a_{11} + c_* + d_*$	$a_{14} + c_* + d_*$
$b_* + c_* + d_*$	$b_* + c_* + d_*$
$a_{15} + b_* + c_* + d_*$	$a_{16} + b_* + c_* + d_*$

Note that the first row of the query to Server n, $n \in \{1,2\}$, is a_n, b_n, c_n, d_n, just as in the original PIR scheme. In the original PIR scheme, the permutations are chosen independently for each message, so that c_n, d_n are not necessarily functions of a_n, b_n. However, here, because we apply the same permutation to every message, and because the same sign σ_n is applied to a_n, b_n, c_n, d_n, the dependence of messages is

preserved in these symbols. In particular, $c_n = \alpha_3 a_n + \beta_3 b_n$, $d_n = \alpha_4 a_n + \beta_4 b_n$, and $H(a_n, b_n, c_n, d_n) = 2$ p-ary units.

The next three rows of the queries to each server are 2-sums (i.e., sums of two symbols) that are also identical to the original PIR scheme, because these queries exploit the side-information from the other server to retrieve new desired symbols. However, notice that because permutations of message symbols are identical, there is a special property that holds here that is evident to each server. For example, Server 1 notes that the 2-sums that contain a_i symbols, i.e., $a_3 + b_2, a_4 + c_2, a_5 + d_2$ have the same index for the other symbol, in this case the index 2. Since we do not wish to expose the identity of the desired message, the same property must hold for all messages. This observation forces the index assignments of all remaining 2-sums.

For example, let us consider the next query term, $b_* + c_*$, from, say, Server 1. Since b_2 was mixed with a_3 in the query $a_3 + b_2$, all 2-sums that include some b_i must have index 3 for the other symbol. Similarly, since c_2 was mixed with a_4, all 2-sums that include some c_j must have index 4 for the other symbol. Thus, for Server 1, the only index assignment possible for query $b_* + c_*$ is $b_4 + c_3$. Similarly, the $b_* + d_*$ must be $b_5 + d_3$ and $c_* + d_*$ must be $c_5 + d_4$. All indices for 2-sums are similarly assigned for Server 2 as well. Thus all indices for 2-sums are settled.

Now let us consider 3-sums. The index assignments for the first three rows for the 3-sums are again straightforward, because as in [1], these are side-information exploitation terms, i.e., new desired message symbols must be mixed with the side-information symbols (2-sums) downloaded from the other server that do not contain desired message symbols. This gives us the following query structure.

$\theta = 1$

Server 1	Server 2
a_1, b_1, c_1, d_1	a_2, b_2, c_2, d_2
$a_3 + b_2$	$a_6 + b_1$
$a_4 + c_2$	$a_7 + c_1$
$a_5 + d_2$	$a_8 + d_1$
$b_4 + c_3$	$b_7 + c_6$
$b_5 + d_3$	$b_8 + d_6$
$c_5 + d_4$	$c_8 + d_7$
$a_9 + b_7 + c_6$	$a_{12} + b_4 + c_3$
$a_{10} + b_8 + d_6$	$a_{13} + b_5 + d_3$
$a_{11} + c_8 + d_7$	$a_{14} + c_5 + d_4$
$b_* + c_* + d_*$	$b_* + c_* + d_*$
$a_{15} + b_* + c_* + d_*$	$a_{16} + b_* + c_* + d_*$

Now, again there is a special property that is evident to each server based on the 3-sums that contain symbols from message a. Suppose we choose any two messages, one of which is a. For example, suppose we choose a, b and consider Server 1. Then there are 2 instances of 3-sums that contain a, b, namely, $a_9 + b_7 + c_6$ and $a_{10} + b_8 + d_6$. Note that the third symbol in each case has the same index (6 in this case). The same is true if for example, we choose a, c or a, d instead. The two 3-sums that contain a, c are $a_9 + b_7 + c_6$ and $a_{11} + c_8 + d_7$, and in each case the third symbol has the *same* index (7 in this case). The two 3-sums that contain a, d are $a_{10} + b_8 + d_6$ and $a_{11} + c_8 + d_7$, and in each case the third symbol has the *same* index (8 in this case). Again, because we do not wish to expose a as the desired message, the same property must be true for all messages. This observation fixes the indices of the remaining 3-sum, $b_* + c_* + d_*$ as follows. The index of d in this term must be 9 because the two 3-sums that contain b, c must have the same index for the third symbol, and according to $a_9 + b_7 + c_6$ this index must be 9. Similarly, the index of c in $b_* + c_* + d_*$ must be 10 because the two 3-sums that contain b, d must have the same index for the third term, and according to $a_{10} + b_8 + d_6$ it has to be 10. The index of b in $b_* + c_* + d_*$ is similarly determined by the term $a_{11} + c_8 + d_7$ to be 11. Thus, the query $b_* + c_* + d_*$ from Server 1 must be $b_{11} + c_{10} + d_9$. Similarly, the query $b_* + c_* + d_*$ from Server 2 must be $b_{14} + c_{13} + d_{12}$.

The last step is again a side-information exploitation step, for which index assignment is trivial (new desired symbol must be combined with the 3-sums queried from the other server that do not contain the desired symbol). Thus, the index assignment is complete, giving us the queries for *PIR2*.

$\theta = 1$

Server 1	Server 2
a_1, b_1, c_1, d_1	a_2, b_2, c_2, d_2
$a_3 + b_2$	$a_6 + b_1$
$a_4 + c_2$	$a_7 + c_1$
$a_5 + d_2$	$a_8 + d_1$
$b_4 + c_3$	$b_7 + c_6$
$b_5 + d_3$	$b_8 + d_6$
$c_5 + d_4$	$c_8 + d_7$
$a_9 + b_7 + c_6$	$a_{12} + b_4 + c_3$
$a_{10} + b_8 + d_6$	$a_{13} + b_5 + d_3$
$a_{11} + c_8 + d_7$	$a_{14} + c_5 + d_4$
$b_{11} + c_{10} + d_9$	$b_{14} + c_{13} + d_{12}$
$a_{15} + b_{14} + c_{13} + d_{12}$	$a_{16} + b_{11} + c_{10} + d_9$

For the sake of comparison, here are the queries generated with *PIR2* when $\theta = 3$, i.e., when message W_3 (symbols c) is desired.

$\theta = 3$

Server 1	Server 2
a_1, b_1, c_1, d_1	a_2, b_2, c_2, d_2
$c_3 + a_2$	$c_6 + a_1$
$c_4 + b_2$	$c_7 + b_1$
$c_5 + d_2$	$c_8 + d_1$
$a_4 + b_3$	$a_7 + b_6$
$a_5 + d_3$	$a_8 + d_6$
$b_5 + d_4$	$b_8 + d_7$
$c_9 + a_7 + b_6$	$c_{12} + a_4 + b_3$
$c_{10} + a_8 + d_6$	$c_{13} + a_5 + d_3$
$c_{11} + b_8 + d_7$	$c_{14} + b_5 + d_4$
$a_{11} + b_{10} + d_9$	$a_{14} + b_{13} + d_{12}$
$c_{15} + a_{14} + b_{13} + d_{12}$	$c_{16} + a_{11} + b_{10} + d_9$

To see why the queries for $\theta = 1$ are indistinguishable from the queries for $\theta = 3$ under $PIR2$, say from the perspective of Server 1, note that the former is mapped to latter under the permutation on $[1 : L]$ that maps

$$\begin{aligned} & (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16) \\ \longrightarrow \quad & (1,3,4,2,5,9,6,10,7,11,8,12,15,13,14,16) \end{aligned}$$

The permutation π is chosen privately and uniformly by the user independent of θ, so both queries are equally likely whether $\theta = 1$ or $\theta = 3$.

2) **Arbitrary** M**:** Recall that the PIR scheme of [1], [8] generates the queries block by block (the k^{th} block is comprised of only k-sums, i.e., sums of k symbols) based on repeated application of two algorithms: **Exploit-SI** and **M-Sym**. **Exploit-SI** takes all the queries from a server (Server n) in block k that do not contain a desired message symbol and adds a new desired symbol to each of them to create new queries for block $k+1$ for the other server (Server $\bar{n}$). This algorithm remains unchanged in our new PIR scheme. The other algorithm, **M-Sym**, is called after **Exploit-SI** for each block and adds queries that do not include desired symbols to force symmetry in queries with respect to messages. The new index assignment process changes only this algorithm. In particular, the change is as follows. Suppose the **M-Sym** algorithm needs to add a query $u_{i_1}(*) + u_{i_2}(*) + \cdots + u_{i_k}(*)$ to Server n in the k^{th} block where $*$ symbols represent indices that need to be assigned. In order to find the index $*$ for $u_{i_l}(*)$, $l \in [k]$, it finds the k-sum query that contains symbols from $u_{i_1}, u_{i_2}, \cdots, u_{i_{l-1}}, u_\theta, u_{i_{l+1}}, \cdots u_{i_k}$ that was generated already for Server n by **Exploit-SI** algorithm. If this query contains $u_\theta(j_l)$ then the index j_l is assigned to u_{i_l}. In this way, the **Exploit-SI** algorithm assigns all indices to generate the query $u_{i_1}(j_1) + u_{i_2}(j_2) + \cdots + u_{i_k}(j_k)$. No other changes are made in the PIR scheme of [1] by the index assignment process.

The privacy of $PIR2$ follows from the following observations. Consider Block k, $k > 1$, of the query to Server n. It contains only k-sums. In fact it contains a k-sum of every type for a total of $\binom{M}{k}$ terms. Now choose any $(k-1)$ message indices $\vec{i} = (i_1, i_2, \cdots, i_{k-1})$, and consider all the queries in Block k that reduce to a $(k-1)$-sum of type $(i_1, i_2, \cdots, i_{k-1})$ after one of their symbols is removed. Then, all the symbols that are removed have the same symbol index j. Furthermore, these are the only symbols with symbol index j among all queries to Server n. For each block k this establishes a one-to-one mapping between each type set $(i_1, i_2, \cdots, i_{k-1})$ and the corresponding symbol index j. The mapping depends on θ. Let us denote this mapping as $j(\theta, \vec{i})$. Then the query for θ is mapped into the query for θ' under a permutation that maps indices $j(\theta, \vec{i}) \longrightarrow j(\theta', \vec{i})$. However, since the permutation of indices is chosen privately and uniformly by the user independent of θ, both queries are equally likely, regardless of whether the desired message is θ or θ'. Thus, the queries do not reveal the identity of the desired message.

The correctness of $PIR2$ follows directly from the correctness of $PIR1$. By the same token, if the messages are independent then $PIR1$ and $PIR2$ have the same rate. Thus, the index assignment process produces a new PIR scheme, $PIR2$, that for independent messages, is equally efficient as $PIR1$ in terms of download, i.e., $PIR2$ is capacity achieving for independent messages. However, depending upon the form of the message dependencies, it turns out that the 'sums' may not be sufficient and more sophisticated mixing of message symbols may be required. For $N = K = 2$ and the dependencies that we consider in this paper, we will need sign assignments, that are explained next.

B. Sign Assignment: PC

In this section, we present the sign assignment procedure that produces the private computation scheme PC from $PIR2$ for arbitrary M. We will use the $M = 4$ example to illustrate its steps. The sign assignment procedure depends on θ. Let us choose $\theta = 3$ to illustrate the process. Note that σ_i are now generated uniformly and independently from $\{+1, -1\}$.

To explain the sign assignment, it is convenient to express each query in a lexicographic order. For example, the query $u_{i_1}(j_1) + u_{i_2}(j_2) + \cdots + u_{i_k}(j_k)$ is in lexicographic order if $i_1 < i_2 < \cdots < i_k$ regardless of the values of the indices j. For our $M = 4$ example, the query $c_9 + a_7 + b_6$ is expressed as $a_7 + b_6 + c_9$ under lexicographic ordering. Note that the lexicographic order for the $M = 4$ example is simply the ordering $a < b < c < d$ and the indices do not matter. The position of the c_* symbol within this lexicographic ordering of query q will be denoted as $\Delta_c(q)$, i.e., for the query $q = a_7 + b_6 + c_9$, we have $\Delta_a(q) = 1, \Delta_b(q) = 2, \Delta_c(q) = 3$ and $\Delta_d(q) = 0$ where the 0 value indicates that a symbol from that message is not present in the query.

Next, the queries are sorted in increasing order of blocks, B, so that the k^{th} block $B = k$, contains only k-sums. Each block is partitioned into sub-blocks, S, such that all the queries q in the same sub-block have the same value of $\Delta_{W_\theta}(q)$. The sub-blocks are sorted within a block in *descending* order of $\Delta_{W_\theta}(q)$ and numbered $S = 1, 2, \cdots$. With this sorting, the query structure is represented as follows.

$\theta = 3$

B	$S(\Delta_c)$	Server 1	Server 2
1	$\cdots$	c_1, a_1, b_1, d_1	c_2, a_2, b_2, d_2
2	1(2)	$a_2 + c_3$	$a_1 + c_6$
	1(2)	$b_2 + c_4$	$b_1 + c_7$
	2(1)	$c_5 + d_2$	$c_8 + d_1$
	3(0)	$a_4 + b_3$	$a_7 + b_6$
	3(0)	$a_5 + d_3$	$a_8 + d_6$
	3(0)	$b_5 + d_4$	$b_8 + d_7$
3	1(3)	$a_7 + b_6 + c_9$	$a_4 + b_3 + c_{12}$
	2(2)	$a_8 + c_{10} + d_6$	$a_5 + c_{13} + d_3$
	2(2)	$b_8 + c_{11} + d_7$	$b_5 + c_{14} + d_4$
	3(0)	$a_{11} + b_{10} + d_9$	$a_{14} + b_{13} + d_{12}$
4	1(3)	$a_{14} + b_{13} + c_{15} + d_{12}$	$a_{11} + b_{10} + c_{16} + d_9$

The sign assignment algorithm for arbitrary M is comprised of 4 steps.

Algorithm: SignAssign

(**Step 1**) Consider queries for which $\Delta_{W_\theta}(q) = 0$, i.e., queries that do not contain desired message symbols. The terms in these queries that occupy even positions (in lexicographic order within each query) are assigned the '$-$' sign. Thus, for example the query $q = a_{11}+b_{10}+d_9$ changes to $q \to q' = a_{11} - b_{10} + d_9$ after the sign assignment. Notice that the signs are alternating in the lexicographic ordering of symbols within the query. The sign assignments for the queries with $\Delta_{W_\theta}(q) = 0$ are now settled.

(**Step 2**) If a symbol is assigned a negative sign in Step 1 then in Step 2 it is assigned a negative sign everywhere it appears. Note that any undesired symbol that appears in the query from one server, appears exactly once within the query to each server.

For our $M = 4$ example, at this point we have,

$\theta = 3$

B	$S(\Delta_c)$	Server 1	Server 2
1	$\cdots$	c_1, a_1, b_1, d_1	c_2, a_2, b_2, d_2
2	1(2)	$a_2 + c_3$	$a_1 + c_6$
	1(2)	$b_2 + c_4$	$b_1 + c_7$
	2(1)	$c_5 + d_2$	$c_8 + d_1$
	3(0)	$a_4 - b_3$	$a_7 - b_6$
	3(0)	$a_5 - d_3$	$a_8 - d_6$
	3(0)	$b_5 - d_4$	$b_8 - d_7$
3	1(3)	$a_7 - b_6 + c_9$	$a_4 - b_3 + c_{12}$
	2(2)	$a_8 + c_{10} - d_6$	$a_5 + c_{13} - d_3$
	2(2)	$b_8 + c_{11} - d_7$	$b_5 + c_{14} - d_4$
	3(0)	$a_{11} - b_{10} + d_9$	$a_{14} - b_{13} + d_{12}$
4	1(3)	$a_{14} - b_{13} + c_{15} + d_{12}$	$a_{11} - b_{10} + c_{16} + d_9$

(**Step 3**) Every query such that $\Delta_{W_\theta}(q) > 0$, i.e., every query that contains a desired message symbol is multiplied by $(-1)^{S+1(\theta\neq 1)}$, where S is the sub-block index and $1(\theta \neq 1)$ is the indicator function that takes the value 1 if $\theta \neq 1$ and 0 if $\theta = 1$.

(**Step 4**) Finally, in Step 4, for each query q that contains a desired symbol, i.e., $\Delta_{W_\theta}(q) > 0$, the desired symbol is assigned the negative sign if it occupies an even numbered position, i.e., if $\Delta_{W_\theta}(q)$ is an even number, and a positive sign if it occupies an odd numbered position, i.e., if $\Delta_{W_\theta}(q)$ is an odd number.

Following this procedure for our running example, we have the final form of the queries as follows.

$\theta = 3$

B	$S(\Delta_c)$	Server 1	Server 2
1	$\cdots$	c_1, a_1, b_1, d_1	c_2, a_2, b_2, d_2
2	1(2)	$a_2 - c_3$	$a_1 - c_6$
	1(2)	$b_2 - c_4$	$b_1 - c_7$
	2(1)	$c_5 - d_2$	$c_8 - d_1$
	3(0)	$a_4 - b_3$	$a_7 - b_6$
	3(0)	$a_5 - d_3$	$a_8 - d_6$
	3(0)	$b_5 - d_4$	$b_8 - d_7$
3	1(3)	$a_7 - b_6 + c_9$	$a_4 - b_3 + c_{12}$
	2(2)	$-a_8 - c_{10} + d_6$	$-a_5 - c_{13} + d_3$
	2(2)	$-b_8 - c_{11} + d_7$	$-b_5 - c_{14} + d_4$
	3(0)	$a_{11} - b_{10} + d_9$	$a_{14} - b_{13} + d_{12}$
4	1(3)	$a_{14} - b_{13} + c_{15} + d_{12}$	$a_{11} - b_{10} + c_{16} + d_9$

To complete the illustration for our $M = 4$ example, let us also present the final queries for $\theta = 1, 2, 4$.

$\theta = 1$

B	$S(\Delta_c)$	Server 1	Server 2
1	$\cdots$	a_1, b_1, c_1, d_1	a_2, b_2, c_2, d_2
2	1(1)	$a_3 - b_2$	$a_6 - b_1$
	1(1)	$a_4 - c_2$	$a_7 - c_1$
	1(1)	$a_5 - d_2$	$a_8 - d_1$
	2(0)	$b_4 - c_3$	$b_7 - c_6$
	2(0)	$b_5 - d_3$	$b_8 - d_6$
	2(0)	$c_5 - d_4$	$c_8 - d_7$
3	1(1)	$a_9 - b_7 + c_6$	$a_{12} - b_4 + c_3$
	1(1)	$a_{10} - b_8 + d_6$	$a_{13} - b_5 + d_3$
	1(1)	$a_{11} - c_8 + d_7$	$a_{14} - c_5 + d_4$
	2(0)	$b_{11} - c_{10} + d_9$	$b_{14} - c_{13} + d_{12}$
4	1(1)	$a_{15} - b_{14} + c_{13} - d_{12}$	$a_{16} - b_{11} + c_{10} - d_9$

$\theta = 2$

B	$S(\Delta_b)$	Server 1	Server 2
1	$\cdots$	b_1, a_1, c_1, d_1	b_2, a_2, c_2, d_2
2	1(2)	$a_2 - b_3$	$a_1 - b_6$
	2(1)	$b_4 - c_2$	$b_7 - c_1$
	2(1)	$b_5 - d_2$	$b_8 - d_1$
	3(0)	$a_4 - c_3$	$a_7 - c_6$
	3(0)	$a_5 - d_3$	$a_8 - d_6$
	3(0)	$c_5 - d_4$	$c_8 - d_7$
3	1(2)	$a_7 - b_9 - c_6$	$a_4 - b_{12} - c_3$
	1(2)	$a_8 - b_{10} - d_6$	$a_5 - b_{13} - d_3$
	2(1)	$b_{11} - c_8 + d_7$	$b_{14} - c_5 + d_4$
	3(0)	$a_{11} - c_{10} + d_9$	$a_{14} - c_{13} + d_{12}$
4	1(2)	$a_{14} - b_{15} - c_{13} + d_{12}$	$a_{11} - b_{16} - c_{10} + d_9$

$\theta = 4$

B	$S(\Delta_d)$	Server 1	Server 2
1	$\cdots$	d_1, a_1, b_1, c_1	d_2, a_2, b_2, c_2
2	1(2)	$a_2 - d_3$	$a_1 - d_6$
	1(2)	$b_2 - d_4$	$b_1 - d_7$
	1(2)	$c_2 - d_5$	$c_1 - d_8$
	2(0)	$a_4 - b_3$	$a_7 - b_6$
	2(0)	$a_5 - c_3$	$a_8 - c_6$
	2(0)	$b_5 - c_4$	$b_8 - c_7$
3	1(3)	$a_7 - b_6 + d_9$	$a_4 - b_3 + d_{12}$
	1(3)	$a_8 - c_6 + d_{10}$	$a_5 - c_3 + d_{13}$
	1(3)	$b_8 - c_7 + d_{11}$	$b_5 - c_4 + d_{14}$
	2(0)	$a_{11} - b_{10} + c_9$	$a_{14} - b_{13} + c_{12}$
4	1(4)	$a_{14} - b_{13} + c_{12} - d_{15}$	$a_{11} - b_{10} + c_9 - d_{16}$

We include the full algorithm here for completeness. $Q(n, \theta)$ denotes the queries for Server $n \in \{1, 2\}$ when W_θ is desired. For any ordered tuple u, let $\texttt{new}(u)$ be a function that, starting with $u(1)$, returns the "next" element in u each time it is called with the same tuple u as its argument.

Algorithm (1) Q-Gen Algorithm.

Input: θ

Output: $Q(1, \theta), Q(2, \theta)$

Initialize: All query sets are initialized as null sets. Also initialize Block $= 1$;

$$Q(1, \theta, \text{Block}, \mathcal{M}) \leftarrow \{u_\theta(1)\}$$
$$Q(1, \theta, \text{Block}, \mathcal{I}) \leftarrow \{u_1(1), \cdots, u_M(1)\}/\{u_\theta(1)\}$$
$$Q(2, \theta, \text{Block}, \mathcal{M}) \leftarrow \{u_\theta(2)\}$$
$$Q(2, \theta, \text{Block}, \mathcal{I}) \leftarrow \{u_1(2), \cdots, u_M(2)\}/\{u_\theta(2)\}$$

for Block $= 2 : M$ **do**

$$Q(1, \theta, \text{Block}, \mathcal{M}) \leftarrow \textbf{Exploit-SI}(Q(2, \theta, \text{Block} - 1, \mathcal{I}))$$
$$Q(2, \theta, \text{Block}, \mathcal{M}) \leftarrow \textbf{Exploit-SI}(Q(1, \theta, \text{Block} - 1, \mathcal{I}))$$

$$Q(1, \theta, \text{Block}, \mathcal{I}) \leftarrow \textbf{M-Sym}(Q(1, \theta, \text{Block}, \mathcal{M}))$$
$$Q(2, \theta, \text{Block}, \mathcal{I}) \leftarrow \textbf{M-Sym}(Q(2, \theta, \text{Block}, \mathcal{M}))$$

end for (Block)

$$Q(1, \theta) \leftarrow \bigcup_{\text{Block} \in [1:M]} \left(Q(1, \theta, \text{Block}, \mathcal{I}) \cup Q(1, \theta, \text{Block}, \mathcal{M})\right)$$
$$Q(2, \theta) \leftarrow \bigcup_{\text{Block} \in [1:M]} \left(Q(2, \theta, \text{Block}, \mathcal{I}) \cup Q(2, \theta, \text{Block}, \mathcal{M})\right)$$

SignAssign$(Q(1, \theta), Q(2, \theta))$

The sub-routines are as follows. θ, Block are assumed to be available to the sub-routines as global variables. $\mathcal{T}_k$ represents the set of all possible choices of k distinct indices in $[1 : M]$. $\overrightarrow{\mathcal{T}}$ indicates that the elements of $\mathcal{T}$ are to be accessed in the natural lexicographic increasing order.

Algorithm (2) M-Sym Algorithm.

Input: $Q = Q(n, \theta, \text{Block}, \mathcal{M})$

Output: $Q^* = Q(n, \theta, \text{Block}, \mathcal{I})$

Initialize: $Q^* \leftarrow \emptyset$.

for each $i_{[1:\text{Block}]} \in \overrightarrow{\mathcal{T}_{\text{Block}}}$, $\theta \notin i_{[1:\text{Block}]}$ **do**

$$Q^* \leftarrow Q^* \cup \{u_{i_1}(j_1) + u_{i_2}(j_2) + \cdots + u_{i_{\text{Block}}}(j_{\text{Block}})\}$$
such that $\forall l \in [1 : \text{Block}]$
$$\exists\ u_\theta(j_l) + \sum_{r \in [1:\text{Block}], r \neq l} u_{i_r}(*) \in Q$$

end for $(i_{[1:\text{Block}]})$

Algorithm (3) Exploit-SI Algorithm.

Input: $Q = Q(n, \theta, \text{Block} - 1, \mathcal{I})$

Output: $Q' = Q(\bar{n}, \theta, \text{Block}, \mathcal{M})$

Initialize: $Q' \leftarrow \emptyset$.

for each $q \in \overrightarrow{Q}$ **do**

$$Q' \leftarrow Q' \cup \{\texttt{new}(u_\theta) + q\}$$

end for (q)

This completes the description of the scheme PC. The correctness of PC follows from that of $PIR2$. Remarkably, if the messages are independent, then PC may be seen as another PIR scheme that achieves the same rate as $PIR1$, $PIR2$, i.e., all three are capacity achieving schemes. The proof of privacy of PC is deferred to Section VI-A for the $M = 4$ setting and to Section VI-B for arbitrary M.

The main advantage of PC is that for the dependent message setting of Theorem 1, it is the optimal private computation scheme. Its proof of optimality is presented next.

V. Proof of Optimality of PC

In this section, we show how PC achieves the capacity of private computation when the messages are dependent. The key idea is that the message dependencies combined with the special index and sign structure of PC create redundant queries, and by downloading generic combinations of the queries[4] instead of downloading each query separately, the download requirement is reduced without compromising on privacy.

[4]Alternatively, random binning (Slepian-Wolf coding) may be used.

A. Proof of Optimality for $M = 4$

To prove optimality, we need to show that the scheme achieves a rate that matches the capacity of private computation according to Theorem 1. Specifically, let us prove that the rate achieved is $8/12 = 2/3$. For this, we will show that the user downloads only 12 symbols from each server. Note that ostensibly there are 15 symbols that are queried from each server. However, it turns out that based on the information available from the other server, 3 of these symbols are redundant. Thus, 12 generic combinations of these 15 symbols are sufficient.

Let us see why this is the case for the queries from Server 1. c_1, d_1 are clearly redundant symbols because according to (9) they are functions of a_1, b_1. So we need one more redundant symbol. Suppose a is desired ($\theta = 1$). Then, consider the 2-sum queries that do not involve the desired message, a. There are 3 such queries. However, the key is that from any 2 we can construct the 3^{rd}. In this case from Server 1 we have: $b_4 - c_3, b_5 - d_3, c_5 - d_4$. But note that

$$\beta_3(b_5 - d_3) - \beta_4(b_4 - c_3) - (\alpha_3\beta_4 - \alpha_4\beta_3)a_3 - \alpha_4 a_4 + \alpha_3 a_5 = (c_5 - d_4)$$

Verify:

$$\begin{aligned}\text{LHS} &= \beta_3(b_5 - d_3) - \beta_4(b_4 - c_3) - (\alpha_3\beta_4 - \alpha_4\beta_3)a_3 - \alpha_4 a_4 + \alpha_3 a_5 \\ &\overset{(9)}{=} \beta_3(b_5 - \alpha_4 a_3 - \beta_4 b_3) - \beta_4(b_4 - \alpha_3 a_3 - \beta_3 b_3) - (\alpha_3\beta_4 - \alpha_4\beta_3)a_3 - \alpha_4 a_4 + \alpha_3 a_5 \\ &= \alpha_3 a_5 + \beta_3 b_5 - \alpha_4 a_4 - \beta_4 b_4 \overset{(9)}{=} (c_5 - d_4) = \text{RHS}\end{aligned}$$

Since the user knows a_3, a_4, a_5 due to the side information available from the other server, out of these 3 equations, 1 is redundant. Thus, one more symbol is saved, giving us 12 effective downloaded symbols, and the rate $8/12$ is achieved. Since this is also the outer bound, this scheme achieves capacity. It can similarly be verified for this $M = 4$ example that the redundancy exists no matter which message is desired.

As another example, suppose c is desired ($\theta = 3$). Referring to the scheme, from Server 1, the three queries (that are 2-sums) not involving c are $a_4 - b_3$, $a_5 - d_3$, $b_5 - d_4$. But note that

$$(\alpha_3\beta_4 - \alpha_4\beta_3)(a_4 - b_3) - \alpha_3(a_5 - d_3) - \alpha_4 c_3 - \beta_4 c_4 + c_5 = \beta_3(b_5 - d_4)$$

Verify

$$\begin{aligned}&\text{LHS} \\ &= (\alpha_3\beta_4 - \alpha_4\beta_3)(a_4 - b_3) - \alpha_3(a_5 - d_3) - \alpha_4 c_3 - \beta_4 c_4 + c_5 \\ &\overset{(9)}{=} (\alpha_3\beta_4 - \alpha_4\beta_3)(a_4 - b_3) - \alpha_3(a_5 - \alpha_4 a_3 - \beta_4 b_3) \\ &\quad - \alpha_4(\alpha_3 a_3 + \beta_3 b_3) - \beta_4(\alpha_3 a_4 + \beta_3 b_4) + (\alpha_3 a_5 + \beta_3 b_5) \\ &= \beta_3(b_5 - \alpha_4 a_4 - \beta_4 b_4) \\ &\overset{(9)}{=} \beta_3(b_5 - d_4) = \text{RHS}\end{aligned}$$

Note that the scheme is designed to satisfy server symmetry, so redundancy exists for Server 2 as well. Note also that the redundant symbols are created in the message symmetry step so that regardless of the value of θ, the sign structure (alternating) is maintained and the symbol index structure is guaranteed to be symmetric. So for all $\theta \in [1 : 4]$, we always have 3 redundant symbols from each server, and downloading 12 symbols per server suffices. The rate achieved is $L/D = 16/24 = 2/3 = C$.

B. Proof of Optimality for Arbitrary M

To prove optimality, we need to show that the scheme achieves a rate of $2/3$. For this, we will show that the user downloads only $\sum_{k=1}^{M}\left(\binom{M}{k} - \binom{M-2}{k}\right) = 2^M - 2^{M-2}$ symbols from each server. Note that the message size is $L = 2^M$, then the rate achieved is $2^M/(2(2^M - 2^{M-2})) = 2/3$, as desired. Note that there are $\binom{M}{k}$ symbols queried in Block $k, k \in [1 : M]$ from each server. However, it turns out that based on information available from the other sever, $\binom{M-2}{k}$ of these symbols are redundant. Thus, $\binom{M}{k} - \binom{M-2}{k}$ generic combinations of these $\binom{M}{k}$ symbols are sufficient.

Next we prove why this is the case in the following lemma.

Lemma 1: For all $\theta \in [1 : M]$, for each server, in Block $k \in [1 : M-2]$, $\binom{M-2}{k}$ of the $\binom{M}{k}$ symbols are redundant, based on the information available from the other server.

Proof: Let us start with the case where $\theta = 1$. Consider the k-sum queries that do not involve the desired message u_1. There are $\binom{M-1}{k}$ such queries, divided into two groups:

1) $\binom{M-2}{k-1}$ queries that involve u_2,
2) $\binom{M-2}{k}$ queries that do not involve u_2.

The key is that the symbols in Group 2 are redundant. Specifically, we show that they are functions of the symbols in Group 1 when u_1 is known.[5]

Example 1: We accompany the general proof with a concrete example to explain the idea. For this example, assume $M = 6$ messages, and denote symbols $u_1, u_2, \cdots, u_6$ by distinct letters $a, b, \cdots, f$, respectively, for simplicity. Consider Block $k = 3$. The queries that do not involve the desired message u_1 are shown below. For this example, we will see that the 4 symbols in Group 2 are functions of the 6 symbols in Group 1.

Group 1	$b_{j_5} - c_{j_2} + d_{j_1}$
	$b_{j_6} - c_{j_3} + e_{j_1}$
	$b_{j_7} - c_{j_4} + f_{j_1}$
	$b_{j_8} - d_{j_3} + e_{j_2}$
	$b_{j_9} - d_{j_4} + f_{j_2}$
	$b_{j_{10}} - e_{j_4} + f_{j_3}$
Group 2	$c_{j_8} - d_{j_6} + e_{j_5}$
	$c_{j_9} - d_{j_7} + f_{j_5}$
	$c_{j_{10}} - e_{j_7} + f_{j_6}$
	$d_{j_{10}} - e_{j_9} + f_{j_8}$

[5]This is guaranteed because the desired variable u_1 in Block k is mixed with side information in Block $k-1$ available from the other server.

Specifically, let us prove that any query in Group 2 is a function of k queries from Group 1. Consider an arbitrary query in Group 2:

$$q_0 = u_{i_1}(j_1) - u_{i_2}(j_2) + u_{i_3}(j_3) - \cdots + (-1)^{k-1} u_{i_k}(j_k)$$

where $3 \le i_1 < i_2 \cdots < i_k$. We show that when u_1 is known, the query q_0 is a function of the following k queries in Group 1:

$$\begin{aligned}
q_1 &= u_2(j_1) - u_{i_2}(p_1) + u_{i_3}(p_2) - \cdots + (-1)^{k-1} u_{i_k}(p_{k-1}) \\
q_2 &= u_2(j_2) - u_{i_1}(p_1) + u_{i_3}(*) - \cdots + (-1)^{k-1} u_{i_k}(*) \\
q_3 &= u_2(j_3) - u_{i_1}(p_2) + u_{i_2}(*) - \cdots + (-1)^{k-1} u_{i_k}(*) \\
&\vdots \\
q_k &= u_2(j_k) - u_{i_1}(p_{k-1}) + u_{i_2}(*) - \cdots \\
&\qquad + (-1)^{k-1} u_{i_{k-1}}(p_{\frac{k(k-1)}{2}})
\end{aligned}$$

where in q_l, we place u_{i_l} in q_0 with u_2. The indices are assigned by the index assignment process.

We assume without loss of generality that $[\alpha_1, \beta_1] = [1, 0], [\alpha_2, \beta_2] = [0, 1]$ (by an invertible change of basis operation). Then we have

$$u_{i_l}(*) = \alpha_{i_l} u_1(*) + \beta_{i_l} u_2(*), l \in [1:k] \tag{10}$$

Note that $u_1(*)$ are assumed known, so we may set $u_1(j_l)$ to zero. Now we show that q_0 is a linear function of $q_{[1:k]}$.

$$q_0 = \beta_{i_1} q_1 - \beta_{i_2} q_2 + \cdots + (-1)^{k-1} \beta_{i_k} q_k, \text{ when } u_1(*) = 0$$

In other words,

$$\begin{aligned}
&\beta_{i_1} u_2(j_1) - \beta_{i_2} u_2(j_2) + \beta_{i_3} u_2(j_3) - \cdots + (-1)^{k-1} \beta_{i_k} u_2(j_k) = \\
&\beta_{i_1} \Big(u_2(j_1) - \beta_{i_2} u_2(p_1) + \beta_{i_3} u_2(p_2) - \cdots + (-1)^{k-1} \beta_{i_k} u_2(p_{k-1}) \Big) \\
&- \beta_{i_2} \Big(u_2(j_2) - \beta_{i_1} u_2(p_1) + \beta_{i_3} u_2(*) - \cdots + (-1)^{k-1} \beta_{i_k} u_2(*) \Big) \\
&+ \beta_{i_3} \Big(u_2(j_3) - \beta_{i_1} u_2(p_2) + \beta_{i_2} u_2(*) - \cdots + (-1)^{k-1} \beta_{i_k} u_2(*) \Big) \\
&\cdots \\
&+ (-1)^{k-1} \beta_{i_k} \Big(u_2(j_k) - \beta_{i_1} u_2(p_{k-1}) + \beta_{i_2} u_2(*) - \cdots \\
&\qquad + (-1)^{k-1} \beta_{i_{k-1}} u_{i_2}(p_{\frac{k(k-1)}{2}}) \Big)
\end{aligned} \tag{11}$$

We verify the above equality holds. First, note that the LHS only contains variables $u_2(j_1), u_2(j_2), u_2(j_3), \cdots, u_2(j_k)$. For these variables, the RHS matches the LHS. Second, we consider the remaining variables. To this end, pick an arbitrary index $p_q, q \in [1 : \frac{k(k-1)}{2}]$ and we show that $u_2(p_q)$ is cancelled in the RHS. Suppose p_q is assigned when the symbol with index p_q is removed, we have the following $k-1$ variables in the k-sum.

$$\begin{aligned}
&u_2, u_{i_1}, u_{i_2}, \cdots, u_{i_{t_1-1}}, u_{i_{t_1+1}}, \cdots, u_{i_{t_2-1}}, u_{i_{t_2+1}}, \\
&\cdots, u_{i_k}, \qquad t_1, t_2 \in [1:k], t_1 < t_2.
\end{aligned} \tag{12}$$

In other words, only $u_{i_{t_1}}$ and $u_{i_{t_2}}$ are not present. There are two such k-sums and then two $u_2(p_q)$ terms. The first $u_2(p_q)$ comes from $u_{i_{t_2}}(p_q)$ and appears in q_{t_1} as $(-1)^{t_2-1} u_{i_{t_2}}(p_q)$ (note that $u_{i_{t_1}}$ is missing in q_{t_1}). The second $u_2(p_q)$ comes from $u_{i_{t_1}}(p_q)$ and appears in q_{t_2} as $(-1)^{t_1} u_{i_{t_1}}(p_q)$ (note that $u_{i_{t_2}}$ is missing in q_{t_2}). Setting $u_1(*) = 0$ and replacing $u_{i_{t_1}}(p_q), u_{i_{t_2}}(p_q)$ with $u_2(p_q)$ in (11), we have

$$\begin{aligned}
&(-1)^{t_1-1} \beta_{i_{t_1}} \Big((-1)^{t_2-1} \beta_{i_{t_2}} u_2(p_q) \Big) + \\
&(-1)^{t_2-1} \beta_{i_{t_2}} \Big((-1)^{t_1} \beta_{i_{t_1}} u_2(p_q) \Big) = 0
\end{aligned} \tag{13}$$

So all terms $u_2(p_q)$ are cancelled in the RHS and (11) holds.

Example 1 (Continued): Pick any query in Group 2, say

$$c_{j_9} - d_{j_7} + f_{j_5} \tag{14}$$

We show that this query is a function of the following 3 queries in Group 1,

$$b_{j_9} - d_{j_4} + f_{j_2}, b_{j_7} - c_{j_4} + f_{j_1}, b_{j_5} - c_{j_2} + d_{j_1}. \tag{15}$$

Next we set the desired variables to zero and we have

$$\begin{aligned}
&\beta_3 b_{j_9} - \beta_4 b_{j_7} + \beta_6 b_{j_5} = \beta_3 \Big(b_{j_9} - \beta_4 b_{j_4} + \beta_6 b_{j_2} \Big) \\
&- \beta_4 \Big(b_{j_7} - \beta_3 b_{j_4} + \beta_6 b_{j_1} \Big) + \beta_6 \Big(b_{j_5} - \beta_3 b_{j_2} + \beta_4 b_{j_1} \Big)
\end{aligned}$$

Note that $b_{j_1}, b_{j_2}, b_{j_4}$ are cancelled. So (14) is redundant (when the desired message symbols are known) .

The proof for arbitrary $\theta \neq 1$ follows from similar procedures. In presenting the proof, we highlight the differences while similar steps are described succinctly.

Note that $k \le M - 2$, so for any k-sum, except u_θ, there exists at least one missing message variable. Pick an arbitrary missing one and suppose it is u_γ. Similarly, consider the k-sum queries that do not involve the desired message u_θ, which are further divided into two groups, depending on whether u_γ is involved (Group 1, $\binom{M-2}{k-1}$ queries) or not (Group 2, $\binom{M-2}{k}$ queries). We show that any query in Group 2 is a function of the queries in Group 1. Define

$$\text{Sum}(\{u_{i_1}, u_{i_2}, \cdots, u_{i_k}\}) \triangleq \sum_{l=1}^{k} (-1)^{l-1} u_{i_l}(j_l)$$

where $i_1 < i_2 \cdots < i_k$. Consider an arbitrary query in Group 2:

$$q_0 = \text{Sum}(\{u_{i_1}, u_{i_2}, \cdots, u_{i_k}\})$$

Note that $u_{i_l}, l \in [1:k]$ is not equal to θ or γ. The symbol indices in q_0 are assigned by the index assignment process. By a change of basis, we express each variable as a linear combination of u_θ and u_γ. Assume without loss of generality,

$$u_{i_l}(*) = \alpha_{i_l} u_\theta(*) + \beta_{i_l} u_\gamma(*), l \in [1:k] \tag{16}$$

Suppose $i_1 < \cdots < i_t < \gamma < i_{t+1} < \cdots < i_k$. Then we show the following equality holds, when $u_\theta(*) = 0$.

$$q_0 = (-1)^{t-1} \left(\sum_{l=1}^{t} (-1)^{l-1} \beta_{i_l} q_l - \sum_{l=t+1}^{k} (-1)^{l-1} \beta_{i_l} q_l \right) \tag{17}$$

where $\forall l \in [1:k], q_l = \text{Sum}(\{u_\gamma, u_{i_{[1:k]/l}}\})$. The rest of the proof, where we verify (17) by showing that the LHS is equal to the RHS, is identical to the case of $\theta = 1$. ■

VI. Proof of Privacy of PC

A. Proof of Privacy for $M = 4$

To see why this scheme is private, we show that the queries are identically distributed, regardless of the value of θ. To this end, we show that the query for $\theta = 2, 3, 4$ has a one-to-one mapping to the query for $\theta = 1$, respectively, through a choice of permutation π and signs σ_i which is made privately and uniformly by the user.

For example, for Server 1 and Server 2, the query for $\theta = 2$ can be converted into the query for $\theta = 1$ by the following mapping:

$$\begin{aligned}
\text{Server 1:}\quad & (3, 2, 7, 9, 10, 8, 15, 14, -\sigma_6, -\sigma_{12}, -\sigma_{13}) \\
\longrightarrow\quad & (2, 3, 9, 7, 8, 10, 14, 15, \sigma_6, \sigma_{12}, \sigma_{13}) \\
\text{Server 2:}\quad & (6, 1, 12, 4, 13, 5, 16, 11, -\sigma_3, -\sigma_9, -\sigma_{10}) \\
\longrightarrow\quad & (1, 6, 4, 12, 5, 13, 11, 16, \sigma_3, \sigma_9, \sigma_{10})
\end{aligned}$$

However, these mappings are privately generated by the user and both alternatives are equally likely regardless of desired message. Hence, these queries are indistinguishable.

We can similarly verify that the other remaining queries for $\theta = 3, 4$, are indistinguishable as well. For Server 1 and Server 2, the query for $\theta = 3$ can be converted into the query for $\theta = 1$ by the following mapping:

$$\begin{aligned}
\text{Server 1:}\quad & (3, 4, 2, 7, 6, 9, 10, 11, 8, -\sigma_8, 14, 13, 15, -\sigma_{12}) \\
\longrightarrow\quad & (2, 3, 4, 9, 7, 6, 8, 10, 11, \sigma_{11}, 15, 14, 13, \sigma_{12}) \\
\text{Server 2:}\quad & (7, 6, 1, 4, 3, 12, 14, 13, 5, -\sigma_5, 11, 10, 16, -\sigma_9) \\
\longrightarrow\quad & (6, 1, 7, 12, 4, 3, 13, 5, 14, \sigma_{14}, 16, 11, 10, \sigma_9)
\end{aligned}$$

The last case is when $\theta = 4$. The mapping from that to $\theta = 1$ is as follows.

$$\begin{aligned}
\text{Server 1:}\quad & (3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 14, 13, 12, 15) \\
\longrightarrow\quad & (2, 3, 4, 5, 8, 10, 11, 6, 7, 9, 15, 14, 13, 12) \\
\text{Server 2:}\quad & (6, 7, 8, 1, 3, 4, 5, 12, 13, 14, 11, 10, 9, 16) \\
\longrightarrow\quad & (1, 6, 7, 8, 5, 13, 14, 3, 4, 12, 16, 11, 10, 9)
\end{aligned}$$

Again, since these mappings are privately generated by the user and both alternatives are equally likely regardless of desired message, these queries are indistinguishable. Thus all queries are indistinguishable and the scheme is private.

B. Proof of Privacy for Arbitrary M

We prove that PC is private. We know that $PIR2$ is private and PC is obtained from $PIR2$ by the sign assignment. Therefore it suffices to show that the sign assignment does not destroy privacy, i.e., $Q(n, \theta)$ still has a one-to-one mapping to $Q(n, 1)$ by a choice of permutation π and signs σ_i which is made by the user privately and uniformly.

The one-to-one mapping is quite simple. Note that each query in $Q(n, 1)$ has alternating signs. Consider $Q(n, \theta)$. We only need to consider the non-desired symbols in queries introduced by **Exploit-SI** (so u_θ is involved). The reason is that the signs of the desired symbols introduced by **Exploit-SI** and the other queries introduced by **M-Sym** are the same as the signs of the queries in $Q(n, 1)$.[6] These queries all satisfy that $\Delta_{W_\theta} > 0$. Now to map $Q(n, \theta)$ to $Q(n, 1)$, for each block, we flip the signs (i.e., replace σ_i with $-\sigma_i$) of variables to the right of u_θ in queries from sub-blocks S if S is odd, and the signs of variables to the left of u_θ in queries from sub-blocks S if S is even.

Example 2: We accompany the general proof with a concrete example to explain the idea. Consider $M = 6$ (messages), block $k = 4$, desired message index $\theta = 4$. For simplicity, we denote $u_1, u_2, \cdots, u_6$ by $a, b, \cdots, f$. In Block $B = k = 4$, we have $\binom{6-1}{4-1} = 10$ queries introduced by **Exploit-SI** *(contains d) as follows. The signs that need to be flipped are colored in red.*

$\theta = 4$

B	$S(\Delta_d)$	*Server n*
4	1(4)	$a_{j_5} - b_{j_2} + c_{j_1} - d_*$
	2(3)	$-a_{j_6} + b_{j_3} + d_* - e_{j_1}$
	2(3)	$-a_{j_7} + b_{j_4} + d_* - f_{j_1}$
	2(3)	$-a_{j_8} + c_{j_3} + d_* - e_{j_2}$
	2(3)	$-a_{j_9} + c_{j_4} + d_* - f_{j_2}$
	2(3)	$-b_{j_8} + c_{j_6} + d_* - e_{j_5}$
	2(3)	$-b_{j_9} + c_{j_7} + d_* - f_{j_5}$
	3(2)	$a_{j_{10}} - d_* - e_{j_4} + f_{j_3}$
	3(2)	$b_{j_{10}} - d_* - e_{j_7} + f_{j_6}$
	3(2)	$c_{j_{10}} - d_* - e_{j_9} + f_{j_8}$

Note that σ_i appears in all message variables with symbol index i, so σ_i might be flipped multiple times and we need to make sure that σ_i is flipped consistently, i.e., the sign flipping rule either changes or does not change the signs of all variables with the same index. This is indeed true, proved as follows. Note that we flip the signs depending on whether the sub-block index is even or odd and if the variables are to the left or right of u_θ. This means, for variables in two consecutive sub-blocks, the variables to the left of u_θ in one sub-block and the variables to the right of u_θ in the other sub-block are simultaneously flipped or unflipped. So it suffices to show that all variables with the same index are

- either in the same sub-block, and all are on the same side of u_θ,
- or in two consecutive sub-blocks, but are on different sides of u_θ.

Example 2 (continued): Referring to the table above, consider all variables with index j_1, i.e., c_{j_1}, e_{j_1}, f_1. c_1 is in sub-block 1 and is to the left of d. e_{j_1}, f_{j_1} are in sub-block 2 and are to the right of d. Further, the signs of $c_{j_1}, e_{j_1}, f_{j_1}$ are all unflipped. As another example, consider all variables with index j_{10}, i.e., $a_{j_{10}}, b_{j_{10}}, c_{j_{10}}$. They are all in sub-block 3 and their signs are all unflipped. One more example: all variables

[6] Note that the indices of the non-desired symbols introduced by **Exploit-SI** do not appear in the queries introduced by **M-Sym**. The reason is seen as follows. Consider a symbol $u_i, i \neq \theta$ that appears in a query introduced by **Exploit-SI** (denote the query by q, so u_θ appears in q) and suppose the index of u_i is j (i.e., we have $u_i(j)$). Now from index assignment, symbols with index j all appear in terms that contain u_θ (thus these terms are all generated by **Exploit-SI**).

with index j_6, $a_{j_6}, c_{j_6}, f_{j_6}$. a_{j_6}, c_{j_6} are in sub-block 2 and are to the left of d. f_{j_6} is in sub-block 3 and is to the right of d. The signs of $a_{j_6}, c_{j_6}, f_{j_6}$ all need to be flipped.

We now find variables with the same symbol index, say $\#$. From index assignment, we know that all occurrences of symbol index $\#$ are in queries that contain the same $k-1$ (distinct) variables (u_θ included). Suppose the message indices of these $k-1$ variables are $i_1, \cdots, i_j, \theta, i_{j+1}, \cdots, i_{k-2}$, where

$$i_1 < \cdots < i_j < \theta < i_{j+1} < \cdots < i_{k-2}. \tag{18}$$

and let the remaining $M-(k-1)$ message indices be denoted by $r_1, r_2, \cdots, r_{M-(k-1)}$. Then the symbol index $\#$ appears in queries

$$\begin{aligned} &u_{r_1}(\#) \pm u_{i_1}() \pm \cdots \pm u_{i_j}() \pm u_\theta() \pm u_{i_{j+1}}() \pm \cdots \pm u_{i_{k-2}}() \\ &\cdots \\ &u_{i_1}() \pm \cdots \pm u_{i_j}() \pm u_\theta() \pm u_{i_{j+1}}() \pm \cdots \pm u_{i_{k-2}}() \pm u_{r_{M-(k-1)}}(\#) \end{aligned} \tag{19}$$

where $\pm$ represents either '+' or '−', determined by sign assignment. These $M-(k-1)$ variables $u_{r_l}, l \in [1 : M-(k-1)]$ can be divided into two sets (one set could be empty), where

- the first set are those u_{r_l} where $r_l < \theta$
- and the second set are those u_{r_l} where $r_l > \theta$

So the variables in the first set are to the left of u_θ and the variables in the second set are to the right of u_θ. Further, the two sets are in consecutive sub-blocks because Δ_{u_θ} only differs by 1. Therefore the sign flipping rule is consistent and the privacy proof is complete.

Example 2 (continued): Suppose we want to find all variables with index $\# = j_1$. They appear in queries that contain a, b, d. The queries in (19) are

$$\begin{aligned} &a_{j_5} - b_{j_2} + c_{j_1} - d_* \\ -&a_{j_6} + b_{j_3} + d_* - e_{j_1} \\ -&a_{j_7} + b_{j_4} + d_* - f_{j_1} \end{aligned}$$

The 3 variables with index $\# = j_1$ are $c_{j_1}, e_{j_1}, f_{j_1}$ (colored in blue). The first set contains $c_{j_1} (< d)$ (in sub-block 1) and the second set contains $e_{j_1}, f_{j_1} (> d)$ (in sub-block 2). As another example, suppose we want to find all variables with index $\# = j_{10}$. The queries in (19) are

$$\begin{aligned} &a_{j_{10}} - d_* - e_{j_4} + f_{j_3} \\ &b_{j_{10}} - d_* - e_{j_7} + f_{j_6} \\ &c_{j_{10}} - d_* - e_{j_9} + f_{j_8} \end{aligned}$$

The 3 variables with index $\# = j_{10}$ are $a_{j_{10}}, b_{j_{10}}, c_{j_{10}} (< d)$. They all belong to the first set (sub-block 3). One more example: find all variables with index $\# = j_6$. The queries in (19) are

$$\begin{aligned} -&a_{j_6} + b_{j_3} + d_* - e_{j_1} \\ -&b_{j_8} + c_{j_6} + d_* - e_{j_5} \\ &b_{j_{10}} - d_* - e_{j_7} + f_{j_6} \end{aligned}$$

The 3 variables with index $\# = j_6$ are $a_{j_6}, c_{j_6}, f_{j_6}$. The first set contains $a_{j_6}, c_{j_6} (< d)$ (in sub-block 2) and the second set contains $f_{j_6} (> d)$ (in sub-block 3). ■

VII. Conclusion

Motivated by privacy concerns in distributed computing, we introduce the private computation problem where a user wishes to compute a desired function of datasets stored at distributed servers without disclosing any information about the function that he wishes to compute to any individual server. The private computation problem may be seen as a generalization of the PIR problem by allowing dependencies among messages. We characterize the capacity of private computation for the elemental setting of $N = 2$ servers, $K = 2$ independent datasets, and arbitrary M linear combinations of the two independent datasets as the possible functions. Surprisingly, this capacity turns out to be identical to the capacity of PIR with $N = 2$ servers and $K = 2$ independent messages. Thus, there is no loss in capacity from the expansion of possible messages to include arbitrary linear combinations in this elemental setting. Furthermore, this insight is also shown to hold at the other extreme, where the number of independent datasets $K \to \infty$ and the user may be interested in arbitrary functions (including possibly non-linear functions). While the asymptotic case is relatively straightforward, the elemental case requires sophisticated structure in the queries in order to optimally exploit message dependencies.

References

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

[2] ——, "The Capacity of Robust Private Information Retrieval with Colluding Databases," *arXiv preprint arXiv:1605.00635*, 2016.

[3] ——, "The Capacity of Symmetric Private Information Retrieval," *arXiv preprint arXiv:1606.08828*, 2016.

[4] K. Banawan and S. Ulukus, "The Capacity of Private Information Retrieval from Coded Databases," *arXiv preprint arXiv:1609.08138*, 2016.

[5] Q. Wang and M. Skoglund, "Symmetric private information retrieval for mds coded distributed storage," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[6] K. Banawan and S. Ulukus, "The capacity of private information retrieval from byzantine and colluding databases," *arXiv preprint arXiv:1706.01442*, 2017.

[7] R. Tandon, "The capacity of cache aided private information retrieval," *arXiv preprint arXiv:1706.07035*, 2017.

[8] H. Sun and S. A. Jafar, "Optimal Download Cost of Private Information Retrieval for Arbitrary Message Length," *arXiv preprint arXiv:1610.03048*, 2016.