Private Sequential Function Computation

Behrooz Tahmasebi and Mohammad Ali Maddah-Ali

Abstract

In this paper, we introduce the problem of private sequential function computation, where a user wishes to compute a composition of a sequence of $K$ linear functions, in a specific order, for an arbitrary input. The user does not run these computations locally, rather it exploits the existence of $N$ non-colluding servers, each can compute any of the $K$ functions on any given input. However, the user does not want to reveal any information about the desired order of computations to the servers. For this problem, we study the capacity $C$, defined as the supremum of the number of desired computations, normalized by the number of computations done at the servers, subject to the privacy constraint. In particular, we prove that $(1 - \frac{1}{N})/(1 - \frac{1}{\max(K,N)}) \leq C \leq 1$. For the achievability, we show that the user can retrieve the desired order of computations, by choosing a proper order of inquiries among different servers, while keeping the order of computations for each server fixed, irrespective of the desired order of computations. In the end, we develop an information-theoretic converse which results an upper bound on the capacity.

Keywords: Capacity, private information retrieval, private computation.
1 Introduction

Outsourcing storage and computation to external parties are the inevitable reaction to the growing size of data and increasing load of processing. One of the main challenges in those arrangements is to ensure the privacy of data and algorithms, with minimum overhead in terms of computation, storage, and communication.

Recently, in the context of private information retrieval (PIR), it is shown that redundancy in data storage over multiple servers can significantly reduce the communication overhead of preserving privacy. In PIR, a user wishes to retrieve a specific file from a database, duplicated across multiple non-colluding servers, while the file identity must be kept private from the servers. In [2], the basic PIR problem has been investigated from an information-theoretic viewpoint and its capacity has been characterized. The capacity there is defined as the maximum number of desired information bits per bit of download in privacy preserving algorithms. This work has been followed for different scenarios, including but not limited to, multiround PIR [3], PIR with colluding servers [4], PIR with coded storage [5], PIR with eavesdroppers [7], PIR with adversary [8], PIR from wiretap channel [9], cache-aided PIR [10,12], PIR with coded and colluding servers [13,15], connections between PIR and distributed storage systems [16], and many other problems [17–36]. The problem of anonymous communication, which is identical to the dual of PIR, is also investigated [37].

Private computation, also known as private function retrieval (PFR), is a problem, where the user wants to compute a linear combination of the files, stored in replicated servers without revealing any information about the coefficients [38,39]. In [38], it is shown that the capacity of the private computation is the same as the capacity of PIR. This means that private computation does not have any extra cost than private retrieval of pure files. This problem is also considered for the coded databases [10,12]. This is also extended to private computation of arbitrary polynomials on Lagrange coded data [43] (see also [44]).

Another popular formulation of private computation is the secure multi-party computation model [45–47]. In the secure multi-party computation, a group of parties are trying to perform a computation task on their private inputs without disclosing any information about them to each other [48]. This means that the objective here is to keep the inputs secure. In this context, an important question is to derive the minimum number of servers required to perform such task. Here, the servers may also collude, up to a given number, in order to gain information about the private inputs. A new formulation of this problem is also recently proposed, where the communication constraint for the computation of high dimensional inputs is also considered. For more explanations, see [49–52].

In this paper, we introduce a new and different formulation for the problem of private computation. We introduce the problem of private sequential function computation as follows. Assume that we have a number of basic functions \( \{F_1,F_2,\ldots,F_K\} \). Using the composition of these basic functions, we can construct a wide class of functions of interest. A user wishes to compute a particular composition of a number of those basic functions for an arbitrary input. The user also wants to offload the computation to \( N \) non-colluding servers which can compute the basic functions, while keeping the desired function secure. Due to the propriety of the basic functions, in this problem, to ensure the privacy, the only information that is needed to be kept secure from the servers is the order of the composition. In this paper, we study this problem for the cases that the basic functions are linear and so they can be represented by square matrices, and also we assume that in the desired composition of the user, each function \( F_k \) appears only once.\(^1\)

\(^1\)We note that there are applications where the computation of a complicated function is the main goal of the problem, and this function is identical to the composition of a number of basic functions, e.g., see [53].
To achieve the desired result, the user sends a sequence of inquiries to the servers in a recursive manner. Each time it sends an inquiry, including a vector and the index of a function, to one of the servers, and waits for that server to return the multiplication of the corresponding matrix to that vector. Then it forms another inquiry for one of the servers, as a function of the initial input vectors, the results of the previous inquiries, and possibly some extra random vectors. In the end, the user should be able to compute the final result, and the servers should not gather any information about the order of computations.

In the private sequential function computation problem, the capacity is defined as the supremum of the number of desired computations per query, subject to the privacy constraint. In this paper, we derive non-trivial lower and upper bounds on the capacity of the private sequential function computation as 

\[ \left(1 - \frac{1}{T}\right) / \left(1 - \frac{1}{\max(K,N)}\right) \leq C \leq 1. \]

For the achievability, we show that the user can compute the desired order of computations, by choosing a proper order of inquiries among different servers, while keeping the order of computations for each server fixed, irrespective of the desired order of computations. Therefore, each server observes a fixed order of computations, and thus gains no information about the desired order of computations.

We then provide an information theoretic converse, which results in an upper bound on the capacity. Moreover, we state an open problem regarding the one-shot capacity of the private sequential computation (i.e., the capacity in the non-asymptotic regime). While this problem remains open through the paper, we show that in a special case, the one-shot capacity of the problem is equal to the achieved lower bound on the (asymptotic) capacity of the problem. Our achievable scheme in this case is essentially different from the achievable scheme for the (asymptotic) capacity.

The rest of this paper is organized as follows. In Section 2, we introduce the mathematical model considered in the paper. Section 3 includes the main result of the paper. To prove the theorem, we first provide a number of illustrative examples in Section 4 and then present the achievable scheme in Section 5. The achievability proof is provided in Section 6, while the proof of the upper bound on the capacity is presented in Section 7. We then mention an open problem in Section 8 and finally, Section 9 concludes the paper.

**Notation:** For any positive integers \(a, b\), let \([a : b] := \{a, a+1, \ldots, b\}\). We use \(S_K\) to denote the set of all permutations of \([1 : K]\). Using \(\Sigma = (\Sigma_K \Sigma_{K-1} \ldots \Sigma_1) \in S_K\), we denote the permutation \(k \mapsto \Sigma_k\). Throughout the paper \(\mathbb{F}\) is an arbitrary finite field. The summation in \(\mathbb{F}\) is denoted by \(\oplus\). All the logarithms are in the base of \(|\mathbb{F}|\). For two random variables \(X, Y\) we write \(X \sim Y\) whenever \(X\) and \(Y\) are identically distributed. The notation \(W_{i:j}\) means \((W_i, W_{i+1}, \ldots, W_j)\). For two functions \(f(n), g(n)\), we write \(f(n) = o(g(n))\), when \(\frac{f(n)}{g(n)} \to 0\), as \(n \to \infty\).

2 Problem Statement

Consider a system, including a user, having access to \(T \in \mathbb{N}\) input vectors \(W_1, W_2, \ldots, W_T\), chosen independently and uniformly at random from \(\mathbb{F}^L\), for some finite field \(\mathbb{F}\) and some integer \(L\). Thus,

\[ H(W_1, W_2, \ldots, W_T) = \sum_{t=1}^{T} H(W_t) = TL. \] (1)

The user wishes to retrieve the result of the sequential (composition) computation of \(K\) linear functions on its input vectors in a specific order. The functions are represented by \(K\) square matrices \(F_1, F_2, \ldots, F_K \in \mathbb{F}^{L \times L}\), that are distributed independently and uniformly over the set of

---

2 In this paper, it is assumed that the servers are not synchronized.
invertible matrices\footnote{Note that the set of invertible matrices includes almost all of the matrices, specifically in high dimensions.} in $\mathbb{F}^{L \times L}$, thus,

$$H(F_1, F_2, \ldots, F_K) = \sum_{k=1}^{K} H(F_k).$$

(2)

The sequence of computations is represented by a permutation $\Sigma = (\Sigma_K \Sigma_{K-1} \cdots \Sigma_1)$ of $[1 : K]$, selected by the user uniformly at random from the set of all possible permutations. Thus, the user wants the result of $F_{\Sigma_K}(F_{\Sigma_{K-1}}(\cdots F_{\Sigma_2}(F_{\Sigma_1}W_t)\cdots))$ for all $t \in [1 : T]$. This means that the order of computations is the same for all $T$ input vectors. The user does not have any information about the matrices $F_k$, $k \in [1 : K]$. To obtain the desired result, it relies on $N \in \mathbb{N}$ non-colluding servers, each has access to the full knowledge of $F_k$, $k \in [1 : K]\footnote{Note that if we assume that the user knows $F_k$, $k \in [1 : K]$, then we cannot bound the possible computational limits at the user side with an information theoretic measure. By this assumption, the servers must compute the functions.}.\footnote{We also refer to the parameter $T$ as the number of requests in the paper.}

Each time that a server is called by the user, it receives some vector $V \in \mathbb{F}^L$ along with an index $k \in [1 : K]$, and returns $F_k V$ to the user. Assume that the servers are not synchronized. Also, assume that the user utilizes the servers $D$ times in total, in order to achieve its desired results. This means that the user generates a sequence of queries $Q_1, Q_2, \ldots, Q_D$, where query $Q_d$, $d \in [1 : D]$, is an ordered triple $Q_d = (Q_d^{(\text{server})}, Q_d^{(\text{input})}, Q_d^{(\text{function})})$, meaning that the user at the $d^{th}$ step, sends $Q_d^{(\text{input})} \in \mathbb{F}^L$ to the server $Q_d^{(\text{server})} \in [1 : N]$, and asks it to run the function $Q_d^{(\text{function})} \in [1 : K]$. The server then computes the desired result, denoted by $A_d$, $A_d = F_{Q_d^{(\text{function})}}Q_d^{(\text{input})} \in \mathbb{F}^L$, and sends it to the user.

A $(K, N, T, D, L)$ scheme of private sequential computation comprises of a sequence of (possibly randomized) encoders $\Phi_d$, $d \in [1 : D]$, such that $Q_d = \Phi_d(W_{1:T}, A_{1:d-1}, \Sigma)$. In addition, it includes a sequence of functions (decoders) $\Psi_t$, $t \in [1 : T]$, such that $\Psi_t(W_{1:T}, A_{1:D}, \Sigma)$ is an estimation of $F_{\Sigma_K} \cdots F_{\Sigma_1}W_t$.

**Definition 1.** The rate of a $(K, N, T, D, L)$ computation scheme is defined as $R = \frac{KT}{D}$.\footnote{We also refer to the parameter $T$ as the number of requests in the paper.}

**Remark 1.** The motivation of this definition for the rate is that in a $(K, N, T, D, L)$ computation scheme, the user wants to compute a composition of $K$ functions on $T$ input vectors (total of $KT$ computations) and it utilizes the servers for $D$ times.

To state the privacy constraint, we need to specify a notation for the sequence of queries received by each server.

**Definition 2.** For any $n \in [1 : N]$, the list of queries received by server $n$ is denoted by

$$\tilde{Q}_n := (Q_d : d \in [1 : D] \text{ and } Q_d^{(\text{server})} = n).$$

(3)

**Remark 2.** Note that the order of the received queries in $\tilde{Q}_n$ is known at server $n$. However, each server does not know the locations of its received queries in the query list of the user, i.e., the exact values of index $d$ for the queries are unknown at the server side, although their relative orders are known.

Now we are ready to present the definition of the achievable rates and the capacity of the private sequential computation problem.
Definition 3. For any positive integers $K, N, T, D$, a positive real $R$ is said to be $T$-achievable, if there is a sequence of $(K, N, T, D, L)$ computation scheme $\{\Phi^{(L)}_{1:D}, \Psi^{(L)}_{1:T}\}_{L=1}^{\infty}$ with the rate of at least $R$, such that

- [Correctness] For any $t \in [1 : T]$,
  \[
  \mathbb{P}\left( \Psi_t^{(L)}(W_{1:T}^{(L)}, A_{1:D}^{(L)}, \Sigma) \neq F^{(L)}_{\Sigma_K^{(L)} F^{(L)}_{\Sigma_{K-1}^{(L)} \ldots F^{(L)}_{\Sigma_1^{(L)}} W_t^{(L)}}} \right) = o(1).
  \]  

- [Privacy] For any $n \in [1 : N], L \in \mathbb{N}$, the list of queries $Q_n^{(L)}$ received by server $n$ must be independent from the order of computations, i.e., $I(Q_n^{(L)}, F^{(L)}_{1:K}; \Sigma) = 0$.

Definition 4. A positive rate $R$ is said to be achievable, if there is a sequence of $T$-achievable rates $\{R_T\}_{T \in \mathbb{N}}$, such that $\lim_{T \to \infty} R_T = R$.

Definition 5. The capacity of the private sequential computation, denoted by $C$, is defined as the supremum of all achievable rates.

3 Main Result

The main result of this paper is summarized in the following theorem.

Theorem 1. The capacity of the private sequential computation problem satisfies the following inequality:

\[
(1 - \frac{1}{N})/(1 - \frac{1}{\max(K, N)}) \leq C \leq 1,
\]  

where $K$ is the number of function to be computed in a sequential and private manner, and $N$ is the number of non-colluding servers.

Remark 3. When $K \leq N$, the capacity is equal to one. This means that if the number of functions does not exceed the number of servers, then one can achieve the private computation without any cost. This is not surprising, because a simple one-shot (i.e., $T = 1$) function computation scheme in which each function is asked to be computed by one specific server ensures privacy.

Remark 4. When $K > N$, the user wishes to compute a number of functions which is greater than the number of available servers. Intuitively speaking, in this case there is at least one server that must compute at least two functions. This means that to achieve privacy, the order of computations at that server should not leak information about the desired order of computations. To ensure privacy in this case, we propose a scheme which has a surprising feature: the order of computations at each server is fixed, irrespective of the desired order of computations. The user can retrieve any order of computations, by selecting an appropriate order for queries $Q_1, Q_2, \ldots, Q_D$, such that the order of computations at each server remains fixed. Some randomness is added to the inputs, such that the server cannot infer any order from some reverse computations. The proposed scheme satisfies both privacy and correctness with zero probability of error, and achieves the lower bound as $T \to \infty$.

Remark 5. The problem of private sequential computation is related to the problem of private computation. However, we have the following observations:
• In private sequential computation, the input vectors \( W_{1:T} \) are not delivered to the servers. This means that the user decides what information about the input files should be delivered to each server. This is due to the specific modeling of the problem where the servers are for function computation. They are not for storing the input files of the user. In addition, in private computation, the user holds the function, while in private sequential computation, the servers hold all of the functions.

• In the model of private sequential computation, the computational limits of servers are taken into account. This means that it is not efficient, due to the definition of the problem, to ask each server to compute all \( K! \) possible permutations of the functions on the input data, and then retrieve the desired result, using a PIR solution.

• In [38], it is shown that the capacity of private computation is \( \frac{1}{N} - \frac{1}{N^K} \), which is similar to the capacity of PIR. One may think that due to the problems definitions, the capacity of private computation is an upper bound on the capacity of private sequential computation problem. However, the capacity expression in Theorem [1] is strictly greater than the aforementioned capacity. It is not a contradiction. Note that the scalings in the definition of rates in the two problems are different. There is a multiplicative factor of \( K \) in this paper that makes the rates definitions different.

Remark 6. As the number of functions goes to infinity, i.e., \( K \to \infty \), the achievable lower bound approaches \( 1 - \frac{1}{N} \). Generally, we have the following inequality:

\[ 1 - \frac{1}{N} \leq C \leq 1. \]  

(6)

To prove of Theorem [1] we propose a private sequential computation scheme in Section 5. The scheme is different for the two cases \( K \leq N \) and \( K > N \). The achievability proof is provided in Section 6 while we propose the proof of the upper bound in Section 7.

4 Motivation and Examples

We consider special cases of the problem of private sequential computation in this section, in order to explain the main idea of the achievable scheme.

4.1 Warming Up

Example 1. This examples explain why for \( K \leq N \), Theorem [1] states that the capacity of private sequential computation is equal to one. Consider the problem of private sequential computation with \( N = 2 \) servers and \( K = 2 \) functions. Assume that the user has only \( T = 1 \) data vector \( W_1 \), and it wants to compute \( F_2(F_1W_1) \), i.e., \( \sigma = (2 \ 1) \). One simple achievable scheme is as follows: the user asks the first server to compute the function \( F_1 \) on \( W_1 \), i.e., \( F_1W_1 \). After receiving the result of the computation, the user asks the second server to compute the function \( F_2 \) on \( F_1W_1 \). Trivially, such scheme has the correctness property. Also, it is private [3]. Note that if desired order is \( \sigma = (1 \ 2) \), i.e., the user wants the results of \( F_1(F_2W_1) \), then the second server computes \( F_2 \) on \( W_1 \), i.e., it computes \( F_2W_1 \), then the first server run \( F_1 \) on \( F_2W_1 \), i.e., computes \( F_1(F_2W_1) \). In other words, for both \( \sigma = (2 \ 1) \) and \( \sigma = (1 \ 2) \), server one always runs \( F_1 \) and server two always runs \( F_2 \). Therefore, each server learns nothing about the desired order of computations.

Note that \( FW_i \sim W_i \) for any invertible matrix \( F \).
We use the following schematic to demonstrate this achievable scheme:

\[
\begin{array}{c|c}
\sigma = (2 \ 1) & \sigma = (1 \ 2) \\
\hline
\text{SERVER 1} & \text{SERVER 2} \\
F_1W_1 & F_2(F_1W_1) \\
\end{array}
\]

Generally, when \( K \leq N \), one can employ a similar approach to achieve the capacity in a one-shot zero-error setting as follows:

\[
\sigma = (\sigma_K \ \sigma_K^{-1} \ \cdots \ \sigma_1)
\]

Example 2. This example demonstrates how to achieve the lower bound on the capacity for cases \( K > N \). Consider the problem of private sequential computation with \( N = 2 \) servers and \( K = 3 \) functions. Assume that the user has \( T \) input vectors \( W_1:T \), for some integer \( T \), and wants to deliver \( F_{\sigma_3}(F_{\sigma_2}(F_{\sigma_1}W_1:T)) \), i.e., the desired order is \( \sigma = (\sigma_3 \ \sigma_2 \ \sigma_1) \). The scheme presented in Example II breaches information about \( \sigma \). It is because the computation of the third function at each server (without any coding) will disclose some information about the order of computations; see the following:

\[
\sigma = (3 \ 2 \ 1)
\]

In the above schemes, one server can apply reverse computation and gain information about the order of computations. One possible solution is to add randomness to the queries to ensure the privacy; see the following:

\[
\sigma = (3 \ 2 \ 1)
\]

Is the above scheme private? Indeed, due to the reverse computation, no information can be inferred by the servers. However, for the other orders of computations \( \sigma \), it is required to ask different orders of inquiries at the servers. For example, due to the above scheme, to compute the order \( \sigma = (2 \ 1 \ 3) \), it is required to ask the servers to compute \( F_3 \rightarrow F_2 \) and \( F_1 \rightarrow F_2 \) as orders of computations. It is impossible to ask a fixed order of inquiries that works for all permutations to achieve the privacy in the above one-shot setting. To solve this issue, we introduce the following computation scheme.

Assume that the user asks the servers to compute a specific sequence of functions, no matter what the desired order of computations is. In particular, assume that the user asks the first server to compute the sequence \( F_1 \rightarrow F_3 \rightarrow F_1 \rightarrow \cdots \rightarrow F_1 \) and the second server to compute the sequence \( F_2 \rightarrow F_3 \rightarrow F_2 \rightarrow \cdots \rightarrow F_2 \). Because the sequence of the computed functions by each server is predetermined, it cannot reveal any information about the desired order of computations.
The next step is to assign the input vectors to the servers at each step, such that the privacy and correctness constraints hold. To achieve privacy, we utilize a number of randomly generated vectors, which are independent from any other random variables in the problem. In below, we demonstrate how the user asks queries to attain this purpose for all possible options for the desired order of computations. Note that the arrows show the order of asking the queries and the random variables $Z_{1:T+2}$ are distributed independently and uniformly over $\mathbb{F}_L$.

\[
\sigma = (3 \ 2 \ 1) \quad \sigma = (3 \ 1 \ 2)
\]

| BLOCK | SERVER 1 | SERVER 2 | BLOCK | SERVER 1 | SERVER 2 |
|-------|----------|----------|-------|----------|----------|
| 1     | $F_1W_1$ | $F_2(F_1W_1)$ | 1     | $F_1(F_2W_1)$ | $F_2W_1$ |
|       | $F_3Z_1$ | $F_3(F_2(F_1W_1) \oplus Z_1)$ |       | $F_3(F_1(F_2W_1) \oplus Z_1)$ | $F_3Z_1$ |
| 2     | $F_1W_2$ | $F_2(F_1W_2)$ | 2     | $F_1(F_2W_2)$ | $F_2W_2$ |
|       | $F_3Z_2$ | $F_3(F_2(F_1W_2) \oplus Z_2)$ |       | $F_3(F_1(F_2W_2) \oplus Z_2)$ | $F_3Z_2$ |
| ...   | ...      | ...      | ...   | ...      | ...      |
| $T$   | $F_1W_T$ | $F_2(F_1W_T)$ | $T$   | $F_1(F_2W_T)$ | $F_2W_T$ |
|       | $F_3Z_T$ | $F_3(F_2(F_1W_T) \oplus Z_T)$ |       | $F_3(F_1(F_2W_T) \oplus Z_T)$ | $F_3Z_T$ |
| $T+1$ | $F_1Z_{T+1}$ | $F_2Z_{T+2}$ | $T+1$ | $F_1Z_{T+2}$ | $F_2Z_{T+1}$ |

\[
\sigma = (2 \ 3 \ 1) \quad \sigma = (2 \ 1 \ 3)
\]

| BLOCK | SERVER 1 | SERVER 2 | BLOCK | SERVER 1 | SERVER 2 |
|-------|----------|----------|-------|----------|----------|
| 1     | $F_1W_1$ | $F_2Z_1$ | 1     | $F_1Z_1$ | $F_2Z_2$ |
|       | $F_3Z_2$ | $F_3(F_1W_1 \oplus Z_2)$ |       | $F_3(W_1 \oplus Z_3)$ | $F_3Z_3$ |
| 2     | $F_1W_2$ | $F_2(F_3(F_1W_1))$ | 2     | $F_1(F_3W_1)$ | $F_2(F_1(F_3W_1))$ |
|       | $F_3Z_3$ | $F_3(F_1W_2 \oplus Z_3)$ |       | $F_3(W_2 \oplus Z_4)$ | $F_3Z_4$ |
| ...   | ...      | $F_2(F_3(F_1W_2))$ | ...   | $F_2(F_3(F_1W_2))$ | ...      |
| $T$   | $F_1W_T$ | $\ldots$ | $T$   | $\ldots$ | $\ldots$ |
|       | $F_3Z_{T+1}$ | $\ldots$ |       | $F_3(W_T \oplus Z_{T+2})$ | $F_3Z_{T+2}$ |
| $T+1$ | $F_1Z_{T+2}$ | $F_2(F_3(F_1W_T))$ | $T+1$ | $F_1(F_3W_T)$ | $F_2(F_1(F_3W_T))$ |
functions on $T$ about the desired permutation of the user. There are two reasons for that. As one can see above, the user has access to its desired content. In addition, the servers do not have any information on the output of the first server in the previous step of computation. After this step, the user has access to $F_2(F_1W_1)$. The user then generates a random vector $Z_1$ and asks the second server and first server to compute $F_3(F_2(W_1) \oplus Z_1)$ and $F_3Z_1$, respectively. After this step the user has access to its desired content. In addition, the servers do not have any information about the desired permutation of the user. There are two reasons for that. As one can see above, the order of computations is the same at server one, for any desired order of computations. In addition, additional randomness guarantees that each server receives $2T+1$ independent vectors, and thus subsequent computations at each server does not leak any information through reverse computations. Hence, the scheme is private.

After showing that the proposed scheme has the privacy and correctness properties, we compute its rate. In total, there are $4T+2$ usage of the servers, while the user wants to compute three functions on $T$ input vectors. Hence, the rate of computation is \( \frac{3T}{4T+2} \). As $T \to \infty$, this rate achieves the claimed lower bound on the capacity of the problem, which is equal to $\frac{3}{4}$.

### 4.2 A Systematic Approach to Construct Achievable Schemes

Example 2 shows that one can use a predetermined association for the sequence of functions to be computed by the servers, in addition to exploit random vectors in order to achieve a lower bound on the capacity in the asymptotic regime. However, we require a systematic approach to construct the achievable schemes for arbitrary $K, N$. We explain our construction in the following example. Note that the method is a bit different from the previous example. However, the ideas of both are similar.

**Example 3.** Consider the problem of private sequential computation with $N = 3$ servers and $K = 4$ functions. Assume that the user has $2T$ input vectors, denoted by $W_{i,j}$, $i \in [1 : T], j \in \{1, 2\}$, and wants to compute $F_{\sigma_4}(F_{\sigma_3}(F_{\sigma_2}(F_{\sigma_1}W_{1:T,1:2})))$, i.e., $\sigma = (\sigma_4 \sigma_3 \sigma_2 \sigma_1)$. We first assign a predetermined order of computations to each server.

---

In this example, we apply minor changes on the notations described in Section 2. These changes are explained through the example.
Assume that the user asks server \( n, n \in \{1, 2, 3\} \), to compute the sequence of functions as

\[
F_n \rightarrow F_n \rightarrow F_4 \rightarrow F_n \rightarrow F_n \rightarrow F_4 \rightarrow \cdots \rightarrow F_n \rightarrow F_n \rightarrow F_4,
\]

no matter what the desired order of computations is. This means that to compute \( 2T \) requests, the user asks \( 3(T + 3) \) queries from each server, and \( 9T + 27 \) queries in total. Let us illustrate the proposed function assignment as follows:

\[
\sigma = (\sigma_4 \sigma_3 \sigma_2 \sigma_1)
\]

| BLOCK | SERVER 1 | SERVER 2 | SERVER 3 |
|-------|----------|----------|----------|
| 1     | \( F_1 \) | \( F_2 \) | \( F_3 \) |
|       | \( F_1 \) | \( F_2 \) | \( F_3 \) |
|       | \( F_1 \) | \( F_4 \) | \( F_4 \) |
| 2     | \( F_1 \) | \( F_2 \) | \( F_3 \) |
|       | \( F_1 \) | \( F_2 \) | \( F_3 \) |
|       | \( F_4 \) | \( F_4 \) | \( F_4 \) |
| \vdots | \vdots   | \vdots   | \vdots   |
| \( T + 3 \) | \( F_1 \) | \( F_2 \) | \( F_3 \) |
|       | \( F_4 \) | \( F_4 \) | \( F_4 \) |

For the convenience in the description of the achievable scheme, we divide the computations of \( 3(T + 3) \) functions at each server to \( T + 3 \) blocks of computations, each comprised of three computations. It can be observed that the function assignment is the same for all the blocks. Assume that the blocks of computations are requested by the user sequentially. This means that the user first asks all of the queries in a block from the servers, then begins the next block. Also assume that the queries in each block are asked by the user in an arbitrary order. We claim that by exploiting this function assignment and order determination, one can design an achievable scheme for any desired permutation. Note that in this case, there are \( 4! = 24 \) distinct permutations. We focus on two specific permutations to illustrate the achievable scheme.

### 4.2.1 \( \sigma = (1 \, 3 \, 4 \, 2) \)

In the following figure, we propose the procedure of computations in the first four blocks. In this scheme, the variables\(^7\) \( Z_\star \) are drawn randomly and uniformly, and they are independent from all the other variables in the problem. In the first block, the requests corresponded to \( W_{1,\{1,2\}} \) are considered and the function \( F_2 \) is applied on them (at server 2). In the second block, the user has access to \( F_2 W_{1,\{1,2\}} \), and asks the servers one and two to apply the function \( F_4 \) on them. To run \( F_4 \), the user utilizes a randomly drawn vector to ensure the privacy of the computation. Also, the user in this block again asks from the second server to perform a similar task to the previous block on new vectors \( W_{2,\{1,2\}} \). The third block is also similar. The function \( F_3 \) is executed on the requests corresponded to \( W_{1,\{1,2\}} \), the function \( F_4 \) is computed for the requests corresponded to \( W_{2,\{1,2\}} \), and the function \( F_2 \) is computed for the requests corresponded to \( W_{3,\{1,2\}} \). In the fourth block, the user again asks three servers to compute specific functions similar to the third block. In

\(^7\) Note that the seven variables in the first block named as \( Z_\star \) are essentially different and independent from each other. However, because we do not exploit them any more in the scheme, they are not denoted differently.
addition, for the requests corresponded to $W_{1,\{1,2\}}$, the function $F_1$ is computed and the final result of commutation for them is available at the end of this block. The rest of the scheme is similar, where $2T$ requests are computed in the $T + 3$ blocks of commutations. Observe that the scheme is correct and private. The privacy is due to the fact that all the inputs given to each server are uniformly and independently drawn vectors, which are independent from the desired permutation of the user. Also the order of computations at each server is the same for all desired permutations.

$$\sigma = (1\ 3\ 4\ 2)$$

| BLOCK | SERVER 1 | SERVER 2 | SERVER 3 |
|-------|----------|----------|----------|
| 1     | $F_1Z_*$ | $F_2W_{1,1}$ | $F_3Z_*$ |
|       | $F_1Z_*$ | $F_2W_{1,1}$ | $F_3Z_*$ |
|       | $F_4Z_*$ | $F_4Z_*$ | $F_4Z_*$ |
| 2     | $F_1Z_*$ | $F_2W_{2,1}$ | $F_3Z_*$ |
|       | $F_1Z_*$ | $F_2W_{2,2} \oplus Z_2$ | $F_3Z_*$ |
|       | $F_4Z_*$ | $F_4Z_*$ | $F_4Z_*$ |
| 3     | $F_1Z_*$ | $F_2W_{3,1}$ | $F_3(F_4(F_2W_{1,1}))$ |
|       | $F_1Z_*$ | $F_2W_{3,2} \oplus Z_3$ | $F_3(F_4(F_2W_{1,2}))$ |
|       | $F_4Z_*$ | $F_4Z_*$ | $F_4Z_*$ |
| 4     | $F_1F_3(F_4(F_2W_{1,1}))$ | $F_2W_{4,1}$ | $F_3(F_4(F_2W_{2,1}))$ |
|       | $F_1F_3(F_4(F_2W_{1,2}))$ | $F_2W_{4,2}$ | $F_3(F_4(F_2W_{2,2}))$ |
|       | $F_4(F_3(F_2W_{3,1} \oplus Z_4))$ | $F_4F_3(F_2W_{3,2} \oplus Z_4)$ | $F_4Z_4$ |

Now we compute the rate of the proposed scheme. Note that there are $2T$ input vectors and we have 4 functions ($8T$ in total), while the user utilizes the servers for $9T + 27$ times. Therefore, the rate of the proposed achievable scheme is \(\frac{8T}{9T + 27}\), which goes to \(\frac{8}{9}\) as $T \rightarrow \infty$.

### 4.2.2 $\sigma = (4\ 3\ 2\ 1)$

To observe that the proposed approach works for any permutation, we consider another permutation here and construct a similar scheme for this case, which is as follows:

$$\sigma = (4\ 3\ 2\ 1)$$

| BLOCK | SERVER 1 | SERVER 2 | SERVER 3 |
|-------|----------|----------|----------|
| 1     | $F_1W_{1,1}$ | $F_2Z_*$ | $F_3Z_*$ |
|       | $F_1W_{2,1}$ | $F_2Z_*$ | $F_3Z_*$ |
|       | $F_4Z_*$ | $F_4Z_*$ | $F_4Z_*$ |
| 2     | $F_1W_{2,1}$ | $F_2(F_1W_{1,1})$ | $F_3Z_*$ |
|       | $F_1W_{2,2}$ | $F_2(F_1W_{1,2})$ | $F_3Z_*$ |
|       | $F_4Z_*$ | $F_4Z_*$ | $F_4Z_*$ |
| 3     | $F_1W_{3,1}$ | $F_2(F_1W_{2,1})$ | $F_3(F_2(F_1W_{1,1}))$ |
|       | $F_1W_{3,2}$ | $F_2(F_2W_{2,2})$ | $F_3(F_2(F_1W_{1,2}))$ |
|       | $F_4Z_*$ | $F_4Z_*$ | $F_4Z_*$ |
| 4     | $F_1W_{4,1}$ | $F_2(F_1W_{3,1})$ | $F_3(F_2(F_1W_{2,1}))$ |
|       | $F_1W_{4,2}$ | $F_2(F_2W_{3,2})$ | $F_3(F_2(F_1W_{2,2}))$ |
|       | $F_4(F_3(F_2(F_1W_{1,1})) \oplus Z_4)$ | $F_4(F_3(F_2(F_1W_{1,2})) \oplus Z_4)$ | $F_4Z_4$ |

\(^8\) We note that this rate is not necessarily optimum for the one-shot case and one may compute privately $2T$ requests with less than $9T + 27$ computations. However, in the asymptotic regime, the rate achieves the claimed lower bound on the capacity and the gap is vanishing.
We now exactly analyze the proposed scheme in Example 3 to generalize it in the next section. Let us formally denote the procedure of computation of the $k^{th}$ function in the sequential computation (which is identical to $F_{\sigma_k}$), for the requests corresponded to $W_{t,(1,2)}$, by $R^k_t$. In Example 3, the user utilized $T + 3$ blocks of computations, and performed all the tasks $R^k_t$, for $k \in \{1, 2, 3, 4\}$ and $t \in [1 : T]$. In particular, it is shown that the following task assignment is feasible (i.e., can be performed using specific codes):

| BLOCK | TASK | TASK | TASK | TASK |
|-------|------|------|------|------|
| 1     | $R_1^1$ |      |      |      |
| 2     | $R_2^1$ | $R_2^2$ |      |      |
| 3     | $R_3^1$ | $R_2^2$ | $R_3^3$ |      |
| 4     | $R_4^1$ | $R_3^2$ | $R_2^3$ | $R_4^4$ |
| 5     | $R_5^1$ | $R_4^2$ | $R_3^3$ | $R_2^4$ |
| ...   | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $T$   | $R_T^1$ | $R_{T-1}^2$ | $R_{T-2}^3$ | $R_{T-3}^4$ |
| $T + 1$ | $R_T^2$ | $R_{T-1}^3$ | $R_{T-2}^4$ |      |
| $T + 2$ | $R_T^3$ | $R_{T-1}^4$ |      |      |
| $T + 3$ | $R_T^4$ |      |      |      |

In the formal description in above, we assigned the task $R_1^1$ to the first block. This means that at the end of the first block, $F_{\sigma_1}W_{1,(1,2)}$ must be available at the user side. For the second block, we assigned two tasks $R_2^1$ and $R_2^2$. This means that at the end of this block, the user must be able to access $F_{\sigma_1}W_{2,(1,2)}$ and $F_{\sigma_2}(F_{\sigma_1}(W_{1,(1,2)}))$. Generally, before the $t^{th}$ block, the tasks $R_{t'}^{k'}$ such that $k' + t' \leq t$ must be already performed, and the tasks $R_t^{k_t}$ such that $k_t + t_t = t + 1$ must be performed at the $t^{th}$ block. By this explanation, one can see that if there is a computation scheme that can perform all of the above tasks, then it is $2T$-achievable. In Example 3, it is shown that such computation scheme exists. We will generalize this approach for arbitrary $K, N$.

5 The Achievable Scheme

5.1 Preliminaries

In this section, we propose an achievable scheme for arbitrary $K, N$. We note that as shown in Example 4, one can simply design an achievable scheme when $K \leq N$. Hence, throughout this section, we focus on cases $K > N$. Also, with a minor change in notations described in Section 2, we assume that there are $T(N - 1)$ requests (rather than $T$ requests) in the system.

We also denote the input vectors of user by $W_{i,j}$, for $i \in [1 : T]$ and $j \in [1 : N - 1]$.

To ensure the privacy and correctness constraints, we rely on the following considerations:

- The index of functions to be computed by each server is assigned in a deterministic manner, no matter what permutation is to be computed. We refer to this task as function assignment.

Note that the achievability proof requires a sequence of schemes for each $T$. We will discuss how this problem does not affect any thing in the asymptotic regime in next section.
in the paper. We use the notion of block of computations, similar to Section 4, to propose an appropriate function assignment.

- Whenever required, the user exploits randomly generated vectors to ensure privacy.

In the following subsections, we first propose a deterministic function assignment, and then we design an appropriate task assignment. After these steps, we propose a vector assignment, which is defined as the process of associating appropriate vectors to be transmitted from the user to the servers in queries.

5.2 Function Assignment

First we define the notion of block of computations as follows.

**Definition 6.** A block of the computations for a private sequential computation problem with $N$ servers and $K$ functions is defined as

| PHASE | NUMBER | SERVER 1 | SERVER 2 | ... | SERVER N |
|-------|--------|----------|----------|-----|----------|
| 1     | 1      | $F_1$    | $F_2$    | ... | $F_N$    |
| 2     | 2      | $F_1$    | $F_2$    | ... | $F_N$    |
|       | ...    | ...      | ...      | ... | ...      |
| $N-1$ | $F_1$  | $F_2$    | ...      | ... | $F_N$    |
| $N$   | $F_{N+1}$ | $F_{N+1}$ | ... | ... | $F_{N+1}$ |
| $N+1$ | $F_{N+2}$ | $F_{N+2}$ | ... | ... | $F_{N+2}$ |
|       | ...    | ...      | ...      | ... | ...      |
| $K-1$ | $F_K$  | $F_K$    | ...      | ... | $F_K$    |

As shown above, each block contains two phases. In the first phase, each server computes a specific function, for $N-1$ times, and in the second phase, all servers compute $K-N$ similar functions.

Now, we are ready to propose the function assignment of the achievable scheme. In the achievable scheme, we utilize a deterministic function assignment, which comprised of $T+K-1$ replicated blocks of computations described above. To determine the order of queries asked by the user, we follow up the following rules:

- The queries are asked block by block, i.e., the user first asks all of the queries of the first block, then asks all of the queries of the second block, and so on.
- At each block, the user asks the queries by an arbitrary order. As we will show, all the vectors sent by the user to the servers at each block are available at the user side before the block begins.

5.3 Task Assignment

Let us first introduce a few useful notations.

**Definition 7.** Formally, we denote the procedure of computation of the $k^{th}$ function in the sequential computation (which is identical to $F_{\sigma_k}$), for the requests corresponded to $W_{t,1:N-1}$, by $R^k_t$. 

13
We refer to the assignment of the tasks to the blocks of computations in order to perform them as task assignment in the paper. Note that there is not any necessity to define blocks, and then assign tasks to them. The only requirement in the paper is to find a correct and private computation scheme. But to the propose achievable scheme, we utilize these definitions and design a computation scheme that performs all tasks $R^t_k, t \in [1 : T], k \in [1 : K]$, in order to reach the desired result. In this way, an important step is to find an appropriate task assignment.

We propose the following task assignment for the achievable scheme:

| BLOCK | TASK | TASK | TASK | \cdots | TASK |
|-------|------|------|------|--------|------|
| 1     | $R^1_1$ |     |     | \cdots |     |
| 2     | $R^1_2$ | $R^2_1$ |     | \cdots |     |
| 3     | $R^1_3$ | $R^2_2$ | $R^3_1$ | \cdots |     |
| \vdots | \vdots | \vdots | \vdots | \ddots | \vdots |
| $K$   | $R^1_K$ | $R^2_{K-1}$ | $R^3_{K-2}$ | \cdots | $R^K_1$ |
| \vdots | \vdots | \vdots | \vdots | \ddots | \vdots |
| $T$   | $R^1_T$ | $R^2_{T-1}$ | $R^3_{T-2}$ | \cdots | $R^K_{T-K}$ |
| $T+1$ | $R^2_T$ | $R^3_{T-1}$ | $R^K_{T-K+1}$ | \cdots | \vdots |
| $T+2$ | $R^3_T$ | $R^K_{T-K+2}$ | \cdots | \vdots | \vdots |
| \vdots | \vdots | \vdots | \vdots | \ddots | \vdots |
| $T+K-1$ | \vdots | \vdots | \vdots | \ddots | $R^K_T$ |

Formally, we assign the task $R^t_k$ to the $t^{th}$ block of computations, if and only if $t' + k' = t + 1$. As a result, all tasks $R^t_k$, such that $t' + k' \leq t$, are performed before the block $t$ begins.

5.4 Vector Assignment

Let us provide a few definitions which are used later.

**Definition 8.** Define

$$
\text{In}_i(R^k_t) := F_{\sigma_{k-1}}(F_{\sigma_{k-2}}(\cdots (F_{\sigma_1} W_{t,i}) \cdots ))
$$

and

$$
\text{Out}_i(R^k_t) := F_{\sigma_k}(F_{\sigma_{k-1}}(\cdots (F_{\sigma_1} W_{t,i}) \cdots ))
$$

for any $t \in [1 : T], k \in [1 : K]$, and $i \in [1 : N - 1]$.

**Corollary 1.**

$$
\text{Out}_i(R^k_t) = F_{\sigma_k} \text{In}_i(R^k_t)
$$

$$
= \text{In}_i(R^{k+1}_t).
$$

**Corollary 2.** After performing the task $R^k_t$, the value of $\text{Out}_{1:N-1}(R^k_t)$ is available at the user side.

To introduce the achievable scheme, after the function assignment and task assignment, the last part is to define the vectors transmitted from the user to the servers in each step. Let us benefit $\pi$ to denote the unique permutation $\pi \in S_K$ such that $\sigma \pi = \pi \sigma$ is the identity permutation. In the
proposed achievable scheme, for each $t \in [1 : T + K - 1]$, at the $t^{th}$ block of computations, the user asks the $n^{th}$ server to perform the computation on the following vectors\[11\].

| BLOCK | PHASE | SERVER $1 \le n \le N - 1$ | SERVER $N$ |
|-------|-------|--------------------------|------------|
|       |       | $\vdots$                | $\vdots$   |
|       | 1     | $\vdots$               | $\vdots$   |
| $t$   |       | $I_n(R_{\pi_n}^{n+1})$ | $Z_{t,1}$  |
|       |       | $I_n(R_{\pi_n}^{n+1})$ | $Z_{t,2}$  |
|       |       | $\vdots$              | $\vdots$   |
|       | 2     | $I_n(R_{\pi_{K+1}}^{n+1}) \oplus Z_{t,K}$ | $Z_{t,K}$  |
|       |       | $\vdots$              | $\vdots$   |

In above, at the first phase, the server $n$, receives the vectors $I_n(R_{\pi_n}^{n+1})$, $i \in [1 : N - 1]$. In the second phase, the two cases $n < N$ and $n = N$ are different. If $n < N$, then the server receives $I_n(R_{\pi_n}^{n+1}) \oplus Z_{t,i}$, $i \in [1 : K - N]$, and if $n = N$, it receives $Z_{t,i}$, $i \in [1 : K - N]$. Note that the random vectors $Z_{t,i}$, $i \in [1 : K - N]$, are distributed uniformly over $\mathbb{F}^L$, and they are independent from each other and from all the other random variables in the problem.

We note that the vectors $I_n(R_{\pi_k}^{n+1})$ for each $i \in [1 : N - 1]$, $k \in [1 : K]$, and $t \in [1 : T + K - 1]$ are well defined, except the cases that $t - \pi_k + 1 \le 0$. For such cases, we assume that the user utilizes a uniformly drawn random vector, which is independent from all of the other variables in the problem, rather than $I_n(R_{\pi_k}^{n+1})$, in the vector assignment. See the vectors $Z_*$ in Example\[3\] for more details.

In Figure\[1\] we illustrate the proposed function assignment and vector assignment.

### 6 Achievability Proof of Theorem\[1\]

In this section, we prove the achievability part of Theorem\[1\]. In order to do so, we utilize the proposed scheme in the previous section. In particular, we first prove that the proposed scheme is feasible, i.e., the proposed vector assignment is feasible, meaning that the user has access to the contents required to be transmitted to the servers at the time of transmission (see Subsection\[5.4\]). Then, we prove the correctness and privacy of the proposed scheme. Finally, considering minor required modifications in the achievable scheme, we prove the desired result.

#### 6.1 Proof of Feasibility and Correctness

In this part, we first prove that the proposed vector assignment is feasible to be performed by the user, and then, we show the correctness of the proposed scheme.

Consider the following propositions:

- $(VA)_t$: The $t^{th}$ block’s vector assignment is feasible, meaning that the vectors sent by the user in that block are available at the user side before the block begins.

\[11\] Note that an important question regarding the proposed vector assignment is that whether it is feasible or not. More precisely, it is required to be proved that the inputs assigned to each block are available at the end of the previous block. We will prove this fact in the next section.
\[ \sigma = (\sigma_K \sigma_{K-1} \cdots \sigma_1), \quad \sigma^{-1} = (\pi_K \pi_{K-1} \cdots \pi_1) \]

| BLOCK | SERVER 1 | SERVER 2 | \ldots | SERVER \(N-1\) | SERVER \(N\) |
|-------|----------|----------|--------|----------------|----------------|
| \vdots | \vdots   | \vdots   | \vdots | \vdots        | \vdots        |
| \(F_1\text{In}_1(\mathcal{R}^\pi_{1-\pi_1+1})\) | \(F_2\text{In}_1(\mathcal{R}^\pi_{1-\pi_2+1})\) | \(F_{N-1}\text{In}_1(\mathcal{R}^\pi_{1-\pi_{N-1}+1})\) | \(F_N\text{In}_1(\mathcal{R}^\pi_{1-\pi_{N+1}+1})\) |
| \(F_1\text{In}_2(\mathcal{R}^\pi_{1-\pi_1+1})\) | \(F_2\text{In}_2(\mathcal{R}^\pi_{1-\pi_2+1})\) | \(F_{N-1}\text{In}_2(\mathcal{R}^\pi_{1-\pi_{N-1}+1})\) | \(F_N\text{In}_2(\mathcal{R}^\pi_{1-\pi_{N+1}+1})\) |
| \vdots | \vdots   | \vdots   | \vdots | \vdots        | \vdots        |
| \(F_1\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_{N-1}+1})\) | \(F_2\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_{N-2}+2})\) | \(F_{N-1}\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_{N-2}+2})\) | \(F_N\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_{N+1}+1})\) |
| \(F_{N+1}(\text{In}_1(\mathcal{R}^\pi_{1-\pi_{N+1}+1}) \oplus Z_{t,1})\) | \(F_{N+1}(\text{In}_2(\mathcal{R}^\pi_{1-\pi_{N+1}+1}) \oplus Z_{t,2})\) | \(F_{N+1}(\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_{N-1}+1}) \oplus Z_{t,1})\) | \(F_{N+1}Z_{t,1}\) |
| \(F_{N+2}(\text{In}_1(\mathcal{R}^\pi_{1-\pi_{N+2}+1}) \oplus Z_{t,1})\) | \(F_{N+2}(\text{In}_2(\mathcal{R}^\pi_{1-\pi_{N+2}+1}) \oplus Z_{t,2})\) | \(F_{N+2}(\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_{N-1}+1}) \oplus Z_{t,2})\) | \(F_{N+2}Z_{t,2}\) |
| \vdots | \vdots   | \vdots   | \vdots | \vdots        | \vdots        |
| \(F_K(\text{In}_1(\mathcal{R}^\pi_{1-\pi_K+1}) \oplus Z_{t,K-N})\) | \(F_K(\text{In}_2(\mathcal{R}^\pi_{1-\pi_K+1}) \oplus Z_{t,K-N})\) | \(F_K(\text{In}_{N-1}(\mathcal{R}^\pi_{1-\pi_K+1}) \oplus Z_{t,K-N})\) | \(F_KZ_{t,K-N}\) |
| \vdots | \vdots   | \vdots   | \vdots | \vdots        | \vdots        |

Figure 1: The proposed function assignment and vector assignment.
• \((TA)_t\): The \(t^{th}\) block’s task assignment is feasible, meaning that after that block, the assigned tasks are perfectly performed.

In order to prove the feasibility of the proposed function and task assignments, we need to show that all the propositions \((VA)_t\), \((TA)_t\), \(t \in [1 : T + K - 1]\), hold. Note that \((VA)_1\) is correct, trivially (see Subsection 5.4). Let us state the following lemma.

**Lemma 1.** For any \(t \in [1 : T + K - 1]\), \((VA)_t \implies (TA)_t\).

**Proof.** Using the assumption in lemma, \(I_{1:N-1}(R^{\pi_k}_{t-\pi_k+1}), k \in [1 : K]\), are available before the \(t^{th}\) block of computations begins. Hence, the user can ask the queries based on the structure described in vector assignment step. Now note that after the first phase of the block, the values of \(F_{n'} I_i(R^{\pi'_{n'}}_{t-\pi'_n+1}), i \in [1 : N - 1], n' \in [1 : N]\) are available at the user side.

In addition, at the second phase of the \(t^{th}\) block, if we cancel the random vectors \(Z_{t,i}, i \in [1 : K - N]\), we conclude that all of the values of \(F_{n'} I_i(R^{\pi'_{n'}}_{t-\pi'_n+1}), i \in [1 : K - 1], n' \in [N + 1 : K]\). All in all, all the values of \(F_{n'} I_i(R^{\pi'_{n'}}_{t-\pi'_n+1}), i \in [1 : N - 1], n' \in [1 : K]\), are available for the user after the completion of the \(t^{th}\) block.

Now observe that by changing the variables as \(n' = \sigma_k\),

\[
F_{n'} I_i(R^{\pi'_{n'}}_{t-\pi'_n+1}) = F_{\sigma_k} I_i(R^{\pi'_{\sigma_k}}_{t-\pi_{\sigma_k}+1})
\]

(11)

\[
= F_{\sigma_k} I_i(R_{t-k+1}^K)
\]

(12)

\[
= \text{Out}_i(R_{t-k+1}^K),
\]

(13)

which means that after \(t^{th}\) block, all tasks \(R_{t-k+1}^K, k \in [1 : K]\) are performed. Therefore, the lemma is proved.

\[\square\]

**Lemma 2.** For any \(t \in [1 : T + K - 2]\), \((TA)_t \implies (VA)_{t+1}\).

**Proof.** We need to show that all of the values of \(I_i(R^{\pi_k}_{(t+1)-\pi_k+1}), i \in [1 : N - 1], k \in [1 : K]\), are available for the user before the \((t + 1)^{th}\) block begins. Note that \(I_i(R^{\pi_k}_{(t+1)-\pi_k+1}) = \text{Out}_i(R^{\pi_k-1}_{(t+1)-\pi_k+1})\). Also, the task \(R^{\pi_k-1}_{(t+1)-\pi_k+1}\) is performed before the \((t + 1)^{th}\) block (using the assumption in lemma), since \(((t + 1) - \pi_k + 1) + (\pi_k - 1) = t + 1\). Hence, the proof is complete.

\[\square\]

Considering Lemma 1 and Lemma 2 and using induction, one can see that all the propositions \((IA)_{1:T+K-1}\) and \((TA)_{1:T+K-1}\) hold. This means that at the end of the last block, all the tasks \(R_t^K, t \in [1 : T]\) are performed, and hence the values of \(\text{Out}_{1:N-1}(R_t^K)\) are available at the user side. Therefore, the correctness proof of the proposed scheme is complete.

### 6.2 Proof of Privacy

To prove the privacy constraint, we require to show that for the proposed achievable scheme, \(I(Q_n, F_{1:K}; \Sigma) = 0\), for each \(n \in [1 : N]\). Due to the deterministic function assignment in the proposed scheme, the only requirement is to show that the inputs given to the servers do not leak any information about the desired permutation.

\[\text{If } (t + 1) - \pi_k + 1 \leq 0, \text{ then the claim is trivial, because the required vectors are randomly drawn.}\]
Let us denote the inputs given to the $n^{th}$ server at the $t^{th}$ block of computations by $X_{n,t} := \text{In}_{1:N}(\mathcal{R}_{t-\pi_n}^{\pi_n+1}),$ for the first phase, and $Y_{n,t,[1:K-N]} \oplus Z_{t,[1:K-N]},$ for the second phase. Due to the proposed vector assignment, we have

$$Y_{n,t} := Y_{n,t,[1:K-N]} = \begin{cases} \text{In}_{n}(\mathcal{R}_{t-\pi_n+1}^{\pi_n+1}) & : n \in [1 : N - 1] \\ O_{[N+1:K]} & : n = N, \end{cases}$$

where $O_t \in F^L, i \in [K + 1 : N],$ are all zero vectors. In addition, we define $Z_t := Z_{t,[1:K-N]}$ and we briefly write $Y_{n,t} \oplus Z_t$ to denote $Y_{n,t,[1:K-N]} \oplus Z_{t,[1:K-N]}.$

We need to show that

$$\mathbb{P}(\Sigma = \sigma'|\tilde{Q}_n, F_{1:K}) = \mathbb{P}(\Sigma = \sigma''|\tilde{Q}_n, F_{1:K}),$$

for each $\sigma', \sigma'' \in S_K.$ Note that

$$\mathbb{P}(\Sigma = \sigma|\tilde{Q}_n, F_{1:K}) = \frac{\mathbb{P}(\tilde{Q}_n, F_{1:K} | \Sigma = \sigma) \mathbb{P}(\Sigma = \sigma)}{\mathbb{P}(\tilde{Q}_n, F_{1:K})} = \frac{\mathbb{P}(F_{1:K} | \Sigma = \sigma) \mathbb{P}(\tilde{Q}_n | \Sigma = \sigma, F_{1:K}) \mathbb{P}(\Sigma = \sigma)}{\mathbb{P}(\tilde{Q}_n, F_{1:K})} = \mu(\tilde{Q}_n, F_{1:K}) \mathbb{P}(\tilde{Q}_n | \Sigma = \sigma, F_{1:K}),$$

where $\mu(\tilde{Q}_n, F_{1:K})$ is a constant that does not depend on $\sigma.$ Also,

$$\mathbb{P}(\tilde{Q}_n | \Sigma = \sigma, F_{1:K}) \overset{(a)}{=} \mathbb{P}\left( (X_{n,t}, Y_{n,t} \oplus Z_t)_{t \in [1:T+K-1]} | \Sigma = \sigma, F_{1:K} \right) = \overset{(b)}{=} \prod_{t \in [1:T+K-1]} \mathbb{P}(X_{n,t}, Y_{n,t} \oplus Z_t | \Sigma = \sigma, F_{1:K}) = \overset{(c)}{=} \prod_{t \in [1:T+K-1]} \mathbb{P}(X_{n,t} | \Sigma = \sigma, F_{1:K}) \prod_{t \in [1:T+K-1]} \mathbb{P}(Y_{n,t} \oplus Z_t | \Sigma = \sigma, F_{1:K})$$

$$= \frac{1}{|\mathbb{P}[L(N-1)(T+K-1)]|} \times \frac{1}{|\mathbb{P}[L(K-N)(T+K-1)]|} = \frac{1}{|\mathbb{P}[L(1)(T+K-1)]|},$$

which completes the proof. Note that (a) holds because the utilized function assignment is deterministic, (b) holds because conditioning on $\Sigma = \sigma$ and $F_{1:K},$ we have the following Markov chain:

$$(X_{n,t_1}, Y_{n,t_1} \oplus Z_{t_1}) \rightarrow W_{t_1-\pi_n+1:1:N-1} \rightarrow W_{t_2-\pi_n+1:1:N-1} \rightarrow (X_{n,t_2}, Y_{n,t_2} \oplus Z_{t_2}),$$

for each $t_1 \neq t_2.$ Also, (c) holds because

$$I(Y_{n,t} \oplus Z_t; X_{n,t} | \Sigma = \sigma, F_{1:K}) = H(Y_{n,t} \oplus Z_t | \Sigma = \sigma, F_{1:K}) - H(Y_{n,t} \oplus Z_t | X_{n,t}, \Sigma = \sigma, F_{1:K}) \leq (K - N)L - H(Y_{n,t} \oplus Z_t | X_{n,t}, \Sigma = \sigma, F_{1:K})$$

$$\leq (K - N)L - H(Y_{n,t} \oplus Z_t | X_{n,t}, \Sigma = \sigma, F_{1:K}, Y_{n,t})$$

$$= (K - N)L - H(Z_t)$$

$$= (K - N)L - (K - N)L$$

$$= 0.$$

\footnote{Note that if $t_i - \pi_n + 1 \leq 0,$ then the claimed Independence hold trivially.}
6.3 Proof of Achievability

In the previous section, we provided an achievable scheme for the cases where the number of requests is divided by $N - 1$. However, to prove the achievability, we need to propose a sequence of achievable scheme for any number of requests (see Section 2).

Consider a general case of the private sequential computation problem in which there are arbitrary number of requests $T(N - 1) + r$. Here $0 \leq r \leq N - 2$ is an arbitrary integer. We propose an achievable scheme for the problem with these parameters as follows. For the first $T(N - 1)$ requests, assume that the user utilizes the proposed scheme of the previous section. For any $r$ remaining requests, the user asks an arbitrary server to compute all of the possible permutations for that request, i.e., the user asks the server for $K \times K!$ times for each request. Due to the previous discussions, it is obvious that this scheme is both private and correct. Hence, it gives a $(T(N - 1) + r)$-achievable rate.

Let us compute the rate for the proposed scheme for arbitrary number of requests $T(N - 1) + r$. For the first $T(N - 1)$ requests, the scheme includes $T + K - 1$ blocks, each require $N(K - 1)$ function computations. For the remaining $r$ requests, the user asks the servers for $r \times K \times K!$ times. All in all, there are $(T + K - 1) \times N(K - 1) + r \times K \times K!$ number of queries. Also, there are $T(N - 1) + r$ requests for the sequential computation of $K$ function at the user side. Hence, the rate of the proposed scheme is

$$R = \frac{K \times (T(N - 1) + r)}{(T + K - 1) \times N(K - 1) + r \times K \times K!}.$$  \hspace{1cm} (30)

One can see that as the number of requests tends to infinity (i.e., $T \to \infty$), this rate achieves

$$\frac{K(N-1)}{N(K-1)} = \frac{1 - \frac{1}{N}}{1 - \frac{1}{K}},$$

which matches the capacity of the problem. Therefore, the achievability proof is complete.

7 Converse Proof of Theorem I

We prove the converse of Theorem I in the following.

7.1 Preliminaries

Lemma 3. Consider a matrix $F \in \mathbb{F}^{L \times L} : \det(F') \neq 0$, which is chosen randomly and uniformly, and $T$ randomly and uniformly generated vectors, $W_t \in \mathbb{F}$, $t \in [1 : T]$, each independent from the other vectors and from $F$. Let $W := (W_1, W_2, \ldots, W_T) \in \mathbb{F}^{L \times T}$ be a randomly generated matrix. We claim that the following propositions hold:

(a) $H(FW_{1:T}) = H(W_{1:T}) = TL$

(b) $H(FW_{1:T}|W_{1:T}) = L \times E\left[ \text{rank}(W) \right]$

(c) $P(\text{rank}(W) < T) \xrightarrow{L \to \infty} 0$

(d) $H(FW_{1:T}|W_{1:T}) = T \times (L - o(L))$

Proof of (a). $TL = H(W_{1:T}) = H(FW_{1:T}|F) \leq H(FW_{1:T}) \leq TL$. \hfill $\Box$

14 Actually, this is not efficient to ask this numerous number of requests and the user can ask fewer questions. However, this effect vanishes in the asymptotic regime.
Proof of (b). Denote an arbitrary realization of random vectors $W_{1:T}$ by $w_{1:T}$. Let $\operatorname{rank}(w) = \dim(\text{span}(w_S))$, where $S = \{s_1, s_2, \ldots, s_{\operatorname{rank}(w)}\} \subseteq [1 : T]$ includes $\operatorname{rank}(w)$ distinct elements. Add specific vectors to the columns of matrix $(w_{s_1}, w_{s_2}, \ldots, w_{s_{\operatorname{rank}(w)}})$, and construct a matrix

$$U = \left( w_{s_1}, w_{s_2}, \ldots, w_{s_{\operatorname{rank}(w)}}, U_1, U_2, \ldots, U_{L - \operatorname{rank}(w)} \right) \in \mathbb{F}^{L \times L},$$

such that $U$ is an invertible matrix. Note that $F' := FU \sim F$. Let $e_t \in \mathbb{F}^L$ denotes the vector with all zero elements, except the $t^{th}$ element, which is equal to one. Note that $U^{-1}w_s = e_t$, for all $t \in [1 : \operatorname{rank}(w)]$. Now we write

$$H(FW_{1:T} | W_{1:T} = w_{1:T}) = \sum_{t=1}^{\operatorname{rank}(w)} H(Fw_{s_t} | Fw_{s_{1:t-1}}) + H(Fw_{1:T}\setminus S | Fw_S) \tag{32}$$

$$(a) \quad \sum_{t=1}^{\operatorname{rank}(w)} H(Fw_{s_t} | Fw_{s_{1:t-1}}) \tag{33}$$

$$= \sum_{t=1}^{\operatorname{rank}(w)} H((FU)U^{-1}w_{s_t} | (FU)U^{-1}w_{s_{1:t-1}}) \tag{34}$$

$$= \sum_{t=1}^{\operatorname{rank}(w)} H(F'U^{-1}w_{s_t} | F'U^{-1}w_{s_{1:t-1}}) \tag{35}$$

$$= \sum_{t=1}^{\operatorname{rank}(w)} H(F'e_t | F'e_{1:t-1}) \tag{36}$$

$$= \sum_{t=1}^{\operatorname{rank}(w)} L \tag{37}$$

$$= L \times \operatorname{rank}(w), \tag{38}$$

where $(a)$ follows since $w_t \in \text{span}(w_S)$, for each $t \in [1 : T] \setminus S$. Now taking the expectation from the two sides of the above inequality yields the desired result. \hfill \Box

Proof of (c). Observe that

$$\mathbb{P}(\operatorname{rank}(W) < T) = \mathbb{P}\left( \bigcup_{t=1}^{T} \{W_t \in \text{span}(W_{1:t-1})\} \right). \tag{39}$$

Note that $\dim(\text{span}(W_{1:t-1})) \leq t - 1$, which means that each vector in $\text{span}(W_{1:t-1})$ can be written as the weighted summation of $t - 1$ specific vectors. This means that $\text{span}(W_{1:t-1})$ contains at most $|\mathbb{F}|^{t-1}$ distinct vectors. However, the vector $W_t$ is chosen uniformly from the set $\mathbb{F}^L$. Therefore, the probability that $W_t$ lie in $\text{span}(W_{1:t-1})$ can be upper bounded by $|\mathbb{F}|^{t-1-L}$. Now we write

$$\mathbb{P}\left( \bigcup_{t=1}^{T} \{W_t \in \text{span}(W_{1:t-1})\} \right) \leq \sum_{t=1}^{T} \mathbb{P}\left( W_t \in \text{span}(W_{1:t-1}) \right) \tag{41}$$

$$\leq \sum_{t=1}^{T} |\mathbb{F}|^{t-1-L} \tag{42}$$

20
\[ F_{T-1}(F) - 1 \] \[ = \frac{|F|^T - 1}{|F|^L (|F| - 1)}, \] \[ (43) \]

which goes to zero as \( L \to \infty \). This completes the proof.

Proof of (d). Using (b), (c), we obtain

\[ H(FW_{1:T}|W_{1:T}) = L \times E \left[ \text{rank}(W) \right] \] \[ = L \times \left( \sum_{t=0}^{T-1} t \times P(\text{rank}(W) = t) + T \times P(\text{rank}(W) = T) \right) \] \[ = L \times \left( \sum_{t=0}^{T-1} t \times o(1) + T \times (1 - o(1)) \right) \] \[ = T \times (L - o(L)). \] \[ (48) \]

Lemma 4. For any random variables \( X, Y, Z \), such that \( Z \) takes values from \( \mathcal{Z} \),

\[ I(X;Y|Z) - \log(|\mathcal{Z}|) \leq I(X,Y) \leq \log(|\mathcal{Z}|) + I(X;Y|Z). \] \[ (49) \]

Proof.

\[ I(X,Y) = H(X) - H(X|Y) \] \[ \leq H(X) - H(X|Y, Z) \] \[ \leq H(X, Z) - H(X|Y, Z) \] \[ = H(Z) + H(X|Z) - H(X|Y, Z) \] \[ = H(Z) + I(X,Y|Z) \] \[ \leq \log(|\mathcal{Z}|) + I(X,Y|Z). \] \[ (55) \]

In addition, we write

\[ I(X,Y) = H(X) - H(X|Y) \] \[ \geq H(X|Z) - H(X|Y) \] \[ = H(X|Z) - H(X|Y, Z) + H(X|Y, Z) - H(X|Y) \] \[ = I(X,Y|Z) + H(X|Y, Z) - H(X|Y) \] \[ = I(X,Y|Z) - I(X;Z|Y) \] \[ \geq I(X,Y|Z) - H(Z|Y) \] \[ \geq I(X,Y|Z) - H(Z) \] \[ \geq I(X,Y|Z) - \log(|\mathcal{Z}|). \] \[ (63) \]
7.2 Proof of the Converse

In order to prove the converse, we show that for each \( T \)-achievable rate \( R_T \), we have \( R_T \leq 1 \), meaning that \( D \geq KT \). Let us define \( D_k^{(L)} \) for a given computation scheme. Note that \( D = \sum_{k=1}^{K} \mathbb{E}[D_k^{(L)}] \). Observe that to prove the converse, it is sufficient to show that \( \mathbb{E}[D_k^{(L)}] \geq T - o(1) \) for each \( k \). Consider a sequence computations schemes, for \( L \in \mathbb{N} \), for the \( T \)-achievable rate \( R_T \) (see Definition 3). To review, note that from Fano’s inequality, we obtain

\[
H(F_{S_1}^{(L)} F_{S_{K-1}}^{(L)} \cdots F_{S_1}^{(L)} W_t^{(L)} | W_{1:T}^{(L)}, Q_{1:D}, A_{1:D}, \Sigma) = o(L),
\]

for any \( t \in [1 : T] \). For the sake of brevity, we do not write the superscripts \( (L) \) throughout this section any more. Also, let \( F_S := F_{S_{K-1}} \cdots F_{S_1} \) and \( F_{\sim k} := (F_1, F_2, \ldots, F_{k-1}, F_{k+1}, \ldots, F_K) \).

Fix an integer \( k \in [1 : K] \). We write

\[
TL = H(F_{S_1} W_{1:T} | F_{\sim k}, \Sigma) \\
= H(F_{S_1} W_{1:T} | W_{1:T}, Q_{1:D}, A_{1:D}, F_{\sim k}, \Sigma) + I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:D}, A_{1:D} | F_{\sim k}, \Sigma)
\]

\[
\leq \sum_{t=1}^{T} H(F_{S_1} W_t | W_{1:T}, Q_{1:D}, A_{1:D}, F_{\sim k}, \Sigma) + I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:D}, A_{1:D} | F_{\sim k}, \Sigma)
\]

\[
\leq \sum_{t=1}^{T} H(F_{S_1} W_t | W_{1:T}, Q_{1:D}, A_{1:D}) + I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:D}, A_{1:D} | F_{\sim k}, \Sigma)
\]

\[
\leq T \times o(L) + I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:D}, A_{1:D} | F_{\sim k}, \Sigma)
\]

\[
\leq o(L) + D \log(K) + I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:D}, A_{1:D} | F_{\sim k}, \Sigma, Q_{1:D}^{(function)}),
\]

where (a) follows from Lemma 3, (b) follows from the correctness property, and (c) follows from Lemma 4 and the fact that \( Q_{1:D}^{(function)} \in [1 : K]^D \).

Let \( \delta_{1:D} \in [1 : K]^D \) denotes an arbitrary realization of \( Q_{1:D}^{(function)} \). Let \( S = \{ s \in [1 : D] : \delta_s = k \} \). Assume \( S = \{ s_1, s_2, \ldots, s_{d_k} \} \) denote the elements of \( S \), which are ordered increasingly.

**Lemma 5.** For any integer \( d \in [1 : D] \),

\[
I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:d}, A_{S_\cup[1:d]} | F_{\sim k}, \Sigma, Q_{1:D}^{(function)} = \delta_{1:D}) = \delta_{1:D}
\]

\[
I(F_{S_1} W_{1:T} ; W_{1:T}, Q_{1:d-1}, A_{S_{\cup[1:d-1]}} | F_{\sim k}, \Sigma, Q_{1:D}^{(function)} = \delta_{1:D}).
\]

**Proof.** Conditioning on \( F_{\sim k}, \Sigma \) and \( Q_{1:D}^{(function)} = \delta_{1:D} \), we have the following Markov chain:

\[
F_{S_1} W_{1:T} \rightarrow (W_{1:T}, Q_{1:d-1}, A_{S_{\cup[1:d-1]}}) \rightarrow Q_d \rightarrow A_d \times 1\{d \in S\},
\]

which concludes the desired result. \( \square \)

\(^{13}\) Th numbers \( D_k^{(L)} \), \( k \in [1 : K] \), are possibly random, due to the random function assignment. However, the summation of them is deterministic, which is equal to \( D \).

\(^{14}\) The realization of the random variable \( D_k \) is denoted by \( d_k \).
Using Lemma 3, we obtain

\[
I(F_2 W_1; W_1, Q_1, A_1 | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D})
= I(F_2 W_1; W_1, Q_1, A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D})
= I(F_2 W_1; W_1, Q_{1:D-1}, A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D})
= I(F_2 W_1; W_1, Q_{1:D-2}, A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D})
= \ldots
= I(F_2 W_1; W_1, Q_1, A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D})
= I(F_2 W_1; W_1 | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D}) + I(F_2 W_1; A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D}, W_1)
\]

(a) \leq I(F_2 W_1; W_1 | F_{\sim k}, \Sigma) + D \log(K) + I(F_2 W_1; A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D}, W_1)

\leq I(F_2 W_1; W_1 | F_{\sim k}, \Sigma) + D \log(K) + H(A_S | F_{\sim k}, \Sigma, Q_{1:D}^{(\text{function})} = \delta_{1:D}, W_1)

\leq I(F_2 W_1; W_1 | F_{\sim k}, \Sigma) + D \log(K) + H(A_S)

\leq I(F_2 W_1; W_1 | F_{\sim k}, \Sigma) + D \log(K) + Ld_k

\leq H(W_1 | F_{\sim k}, \Sigma) - H(F_2 W_1 | F_{\sim k}, \Sigma, W_1) + D \log(K) + Ld_k

\equiv TL - T(L - o(L)) + D \log(K) + Ld_k

where (a) follows from Lemma 3, (b) follows from the fact that |S| = d_k, and (c) follows from Lemma 3. Taking the expectation from (87) and combining with (70) results

\[
TL \leq o(L) + D \log(K) + T(L - o(L)) + D \log(K) + L \times E\left[D_k\right]
= o(L) + 2D \log(K) + L \times E\left[D_k\right].
\]

Therefore,

\[
T \leq o(1) + \frac{2D \log(K)}{L} + E\left[D_k\right]
\leq o(1) + E\left[D_k\right].
\]

Hence, we conclude that \(E[D_k] \geq T - o(1)\), which completes the proof.

8 An Open Problem: One-shot Capacity?

As long as \(K \leq N\), it is shown in the paper that the one-shot capacity of the private sequential computation is the same as the asymptotic capacity, which is equal to one. A remaining open problem is the characterization of the one-shot capacity of private sequential computation for cases \(K > N\). In particular, for a given \(T\), if we define \(T\)-capacity as the supremum of all \(T\)-achievable rates, how can we characterize the exact value of it. While this problem remains open throughout the paper, we provide a minor result on the one-shot capacity for the special case of \(K = 3\) functions and \(N = 2\) servers. Note that due to Theorem 1, for the (asymptotic) capacity of the problem in this case, the rate of \(\frac{3}{4}\) is achievable.
Theorem 2. The one-shot capacity of the private sequential computation for the case of $K = 3$ functions and $N = 2$ servers is the same as the claimed lower bound for the (asymptotic) capacity of the problem, i.e., when $T = 1$, the user can ask 4 queries from 2 servers in order to sequentially private compute three functions.

Proof. To prove the theorem, it is sufficient to propose an achievable scheme for the one-shot case. The proposed one-shot achievable scheme is definitely different from the achievable scheme of Theorem 1. The key idea to construct a one-shot achievable scheme is to exploit a random function assignment strategy.

8.1 One-shot Achievable Scheme

Let us label the servers from $\{1, 2\}$. Assume the servers know their label. Let $(U_3 \ U_2 \ U_1) \in \mathcal{S}_3$ be a uniform random permutation, independent from the desired permutation $\Sigma = (\Sigma_3 \ \Sigma_2 \ \Sigma_1)$. We utilize a random function assignment to propose an achievable scheme as follows:

$$U_1 = \Sigma_3 \quad \text{and} \quad U_1 \neq \Sigma_3$$

In the above random function assignment, for cases where $U_1 = \Sigma_3$, we utilize the left assignment, and for cases where $U_1 \neq \Sigma_3$, we utilize the right assignment.

Let $Z \in \mathbb{F}^L$ be a randomly and uniformly drawn vector, independent from all of the other random variables in the problem. For vector assignment, we use the following rules:

Now we prove the desired properties for the proposed one-shot scheme. The correctness proof is trivial, due to the above explanations in the figures. For privacy, note that the inputs given to each
server by the user are independent from each other, and independent from the order of computations, and both are distributed uniformly over $\mathbb{F}^L$. Hence, they do not reveal any information about the order of computations to the servers. To complete the privacy proof, it just remains to show that the proposed function assignment is also the same.

The first server cannot attain any information about the order of computations from the function assignment, because $(U_1, U_2)$ is independent from $\Sigma$. For the second server, let $(V_1, V_2)$ denote the index of functions asked from it to compute, respectively. In other words, $(V_1, V_2) = (U_3, U_1)$, if $U_1 = \Sigma_3$, and $(V_1, V_2) = (U_1, U_2)$, if $U_1 \neq \Sigma_3$. We require to show that $P(\Sigma|V_1 = v_1, V_2 = v_2)$ does not depend on $\Sigma$, for each distinct $v_1, v_2 \in \{1, 2, 3\}$.

Let $\mathcal{A} := \{U_1 = \Sigma_3\}$ be a probability event, which happens with probability $\frac{1}{3}$. We write

$$P(\Sigma|V_1 = v_1, V_2 = v_2) = P(\Sigma|V_1 = v_1, V_2 = v_2, \mathcal{A})P(\mathcal{A}|V_1 = v_1, V_2 = v_2)$$

$$+ P(\Sigma|V_1 = v_1, V_2 = v_2, \mathcal{A}^c)P(\mathcal{A}^c|V_1 = v_1, V_2 = v_2)$$

$$= \frac{1}{3} \times P(\Sigma|V_1 = v_1, V_2 = v_2, \mathcal{A}) + \frac{2}{3} \times P(\Sigma|V_1 = v_1, V_2 = v_2, \mathcal{A}^c)$$

$$= \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} \times (0, 0, 0, 0) + \frac{2}{3} \times (0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$$

$$= \frac{1}{6} \times (1, 1, 1, 1, 1, 1, 1),$$

where $(a)$ is due to the structure of the proposed random function assignment. \hfill $\square$

## 9 Conclusion and Discussion

Information and coding theory open new directions toward optimal resource utilization for the distributed storages and distributed computation systems. Codes have demonstrated their ability to solve challenges in such systems. This line of research revolves around finding the potentials of coding theory to be used in the computation/storage systems.

In this paper, we progress toward the obtaining fundamental limits of private computation from information theoretic viewpoint. A system model for specific type of sequential computation was proposed, in addition to deriving non-trivial lower/upper bounds for the capacity of the problem. An open problem regarding the capacity of private sequential computation for the finite number of requests is also proposed. The future direction for this work is the consideration of the non-linear functions, colluding databases, and the computation when the user is allowed to request the computation of a function in the sequence redundantly. Another question which can be considered for the future works is the investigation of the potentials of random function assignment to achieve improved one-shot or asymptotic rates for the private sequential computation problem.

### References

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

[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, July 2017.

[3] ——, “Multiround private information retrieval: Capacity and storage overhead,” *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5743–5754, Aug. 2018.
[4] ——, “The capacity of robust private information retrieval with colluding databases,” *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2361–2370, April 2018.

[5] ——, “Private information retrieval from MDS coded data with colluding servers: Settling a conjecture by freij-hollanti et al.” *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1000–1022, Feb. 2018.

[6] K. Banawan and S. Ulukus, “The capacity of private information retrieval from coded databases,” *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1945–1956, March 2018.

[7] Q. Wang and M. Skoglund, “Secure private information retrieval from colluding databases with eavesdroppers,” in 2018 *IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2456–2460.

[8] ——, “Secure symmetric private information retrieval from colluding databases with adversaries,” in 2017 55th *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct. 2017, pp. 1083–1090.

[9] K. Banawan and S. Ulukus, “Private information retrieval through wiretap channel II,” in 2018 *IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 971–975.

[10] R. Tandon, “The capacity of cache aided private information retrieval,” in 2017 55th *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct. 2017, pp. 1078–1082.

[11] Y. Wei, K. Banawan, and S. Ulukus, “Cache-aided private information retrieval with partially known uncoded prefetching,” in 2018 *IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[12] ——, “Cache-aided private information retrieval with unknown and uncoded prefetching,” in 2018 *IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1949–1953.

[13] R. Freij-Hollanti, O. Gnilke, C. Hollanti, and D. Karpuk, “Private information retrieval from coded databases with colluding servers,” *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 647–664, 2017.

[14] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, and C. Hollanti, “Robust private information retrieval from coded systems with byzantine and colluding servers,” in 2018 *IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2451–2455.

[15] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, and C. Hollanti, “Private information retrieval from coded storage systems with colluding, byzantine, and unresponsive servers,” *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3898–3906, June 2019.

[16] H. Yang, W. Shin, and J. Lee, “Private information retrieval for secure distributed storage systems,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 12, pp. 2953–2964, Dec. 2018.

[17] Z. Zhang and J. Xu, “The optimal sub-packetization of linear capacity-achieving PIR schemes with colluding servers,” *IEEE Transactions on Information Theory*, pp. 1–1, 2018.
[18] M. A. Attia, D. Kumar, and R. Tandon, “The capacity of uncoded storage constrained PIR,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1959–1963.

[19] K. Banawan and S. Ulukus, “Private information retrieval under asymmetric traffic constraints,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2446–2450.

[20] S. Li and M. Gastpar, “Single-server multi-user private information retrieval with side information,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1954–1958.

[21] H. Lin, S. Kumar, E. Rosnes, and A. G. i Amat, “An MDS-PIR capacity-achieving protocol for distributed storage using non-mds linear codes,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 966–970.

[22] C. Tian, H. Sun, and J. Chen, “A shannon-theoretic approach to the storage-retrieval tradeoff in PIR systems,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1904–1908.

[23] N. Raviv and I. Tamot, “Private information retrieval is graph based replication systems,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1739–1743.

[24] Z. Chen, Z. Wang, and S. Jafar, “The asymptotic capacity of private search,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2122–2126.

[25] H. Sun, “Anonymous information delivery,” *CoRR*, vol. abs/1806.05601, 2018. [Online]. Available: [http://arxiv.org/abs/1806.05601](http://arxiv.org/abs/1806.05601)

[26] H. Sun and S. A. Jafar, “The capacity of symmetric private information retrieval,” *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 322–329, Jan 2019.

[27] 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)*, Nov 2018, pp. 1–5.

[28] Z. Jia, H. Sun, and S. A. Jafar, “Cross subspace alignment and the asymptotic capacity of x–secure t–private information retrieval,” *IEEE Transactions on Information Theory*, pp. 1–1, 2019.

[29] K. Banawan and S. Ulukus, “Noisy private information retrieval,” in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, Oct 2018, pp. 1694–1698.

[30] ———, “Private information retrieval from multiple access channels,” in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[31] Z. Jia and S. A. Jafar, “On the asymptotic capacity of x-secure t-private information retrieval with graph based replicated storage,” *CoRR*, vol. abs/1904.05906, 2019. [Online]. Available: [http://arxiv.org/abs/1904.05906](http://arxiv.org/abs/1904.05906)

[32] H. Sun and S. A. Jafar, “On the capacity of computation broadcast,” *CoRR*, vol. abs/1903.07597, 2019. [Online]. Available: [http://arxiv.org/abs/1903.07597](http://arxiv.org/abs/1903.07597)
[33] W. Chang and R. Tandon, “On the upload versus download cost for secure and private matrix multiplication,” CoRR, vol. abs/1906.10684, 2019. [Online]. Available: http://arxiv.org/abs/1906.10684

[34] Y. Zhang, E. Yaakobi, and T. Etzion, “Private proximity retrieval codes,” CoRR, vol. abs/1907.10724, 2019. [Online]. Available: http://arxiv.org/abs/1907.10724

[35] M. H. Mousavi, M. A. Maddah-Ali, and M. Mirmohseni, “Private inner product retrieval for distributed machine learning,” CoRR, vol. abs/1902.06319, 2019. [Online]. Available: http://arxiv.org/abs/1902.06319

[36] F. Kazemi, E. Karimi, A. Heidarzadeh, and A. Sprintson, “Private information retrieval with private coded side information: The multi-server case,” CoRR, vol. abs/1906.11278, 2019. [Online]. Available: http://arxiv.org/abs/1906.11278

[37] H. Sun, “The capacity of anonymous communications,” IEEE Transactions on Information Theory, vol. 65, no. 6, pp. 3871–3879, June 2019.

[38] H. Sun and S. A. Jafar, “The capacity of private computation,” IEEE Transactions on Information Theory, 2018. [Online]. Available: http://arxiv.org/abs/1710.11098

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

[40] S. A. Obead, H. Lin, E. Rosnes, and J. Kliewer, “Capacity of private linear computation for coded databases,” CoRR, vol. abs/1810.04230, 2018. [Online]. Available: http://arxiv.org/abs/1810.04230

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

[42] S. A. Obead and J. Kliewer, “Achievable rate of private function retrieval from MDS coded databases,” in 2018 IEEE International Symposium on Information Theory (ISIT), June 2018, pp. 2117–2121.

[43] N. Raviv and D. A. Karpuk, “Private polynomial computation from lagrange encoding,” CoRR, vol. abs/1812.04142, 2018. [Online]. Available: https://arxiv.org/abs/1812.04142

[44] S. A. Obead, H. Lin, E. Rosnes, and J. Kliewer, “Private polynomial computation for noncolluding coded databases,” CoRR, vol. abs/1901.10286, 2019. [Online]. Available: http://arxiv.org/abs/1901.10286

[45] A. C. Yao, “Protocols for secure computations,” in 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), Nov. 1982, pp. 160–164.

[46] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[47] M. Ben-Or, S. Goldwasser, and A. Wigderson, “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” in Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 1988, pp. 1–10.

[48] D. Evans, V. Kolesnikov, and M. Rosulek, “A pragmatic introduction to secure multi-party computation,” Foundations and Trends in Privacy and Security, vol. 2, no. 2-3, pp. 70–246, 2018.
[49] H. A. Nodehi and M. A. Maddah-Ali, “Limited-sharing multi-party computation for massive matrix operations,” in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1231–1235.

[50] H. A. Nodehi, S. R. H. Najarkolaei, and M. A. Maddah-Ali, “Entangled polynomial coding in limited-sharing multi-party computation,” in *2018 IEEE Information Theory Workshop (ITW)*, Nov. 2018.

[51] Q. Yu, N. Raviv, and A. S. Avestimehr, “Coding for private and secure multiparty computing,” in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[52] J. Kakar, S. Ebadifar, and A. Sezgin, “Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation,” *CoRR*, vol. abs/1810.13006, 2018. [Online]. Available: [http://arxiv.org/abs/1810.13006](http://arxiv.org/abs/1810.13006)

[53] B. Tahmasebi, M. A. Maddah-Ali, S. Parsaeefard, and B. H. Khalaj, “Optimum transmission delay for function computation in NFV-based networks: The role of network coding and redundant computing,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2233–2245, Oct. 2018.