

# Private Information Retrieval and Its Applications: An Introduction, Open Problems, Future Directions

Sajani Vithana      Zhusheng Wang      Sennur Ulukus

Department of Electrical and Computer Engineering

University of Maryland, College Park, MD 20742

*spallego@umd.edu*      *zhusheng@umd.edu*      *ulukus@umd.edu*

arXiv:2304.14397v1 [cs.IT] 27 Apr 2023

**Abstract**—Private information retrieval (PIR) is a privacy setting that allows a user to download a required message from a set of messages stored in a system of databases without revealing the index of the required message to the databases. PIR was introduced under computational privacy guarantees, and is recently re-formulated to provide information-theoretic guarantees, resulting in *information theoretic privacy*. Subsequently, many important variants of the basic PIR problem have been studied focusing on fundamental performance limits as well as achievable schemes. More recently, a variety of conceptual extensions of PIR have been introduced, such as, private set intersection (PSI), private set union (PSU), and private read-update-write (PRUW). Some of these extensions are mainly intended to solve the privacy issues that arise in distributed learning applications due to the extensive dependency of machine learning on users' private data. In this article, we first provide an introduction to basic PIR with examples, followed by a brief description of its immediate variants. We then provide a detailed discussion on the conceptual extensions of PIR, along with potential research directions.

## I. INTRODUCTION

Private information retrieval (PIR) describes an elemental privacy setting where a user downloads a single message out of a set of messages stored in multiple non-colluding and replicated databases, without revealing the identity of the downloaded message. PIR finds applications in a multitude of fields, such as, medicine, finance and national defense, to access useful data without leaking any information about the retriever's needs, intents or interests. For example, an investor may wish to download certain relevant stock market records without revealing their identities, from which information about potential investments can be leaked. Similarly, an inventor may wish to search for inventions in a patent database without revealing what is being searched for, to avoid leaking any information on their own invention prior to publication.

PIR problem was first introduced in the seminal paper [1] which provided PIR schemes and computational guarantees. While being an active area of research in computer science for many years, PIR recently has attracted significant interest in information theory with the leading paper [2] which characterizes the capacity of PIR. The subsequent papers have characterized the capacities of various PIR settings in different scenarios. These capacity results provide fundamental limits on the performance of PIR, analogous to Shannon's capacity theorem for communication channels.

The basic PIR setting considers a system of  $N$  non-colluding databases, each storing  $K$  independent messages. A user sends queries to the system of databases, with the goal

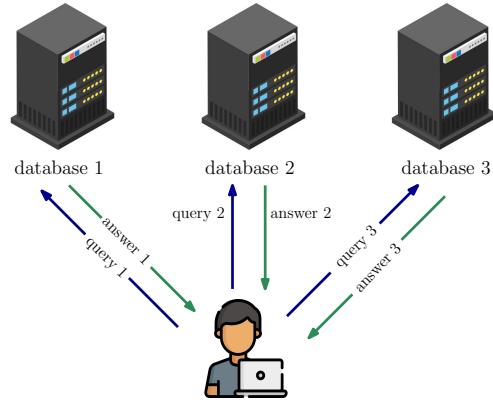


Fig. 1: Communications in the PIR process.

of privately requesting a desired message. Each database then sends the corresponding answer back to the user, as shown in Fig. 1. The queries sent by the user must not reveal any information about the required message index to any of the databases. Formally, from the perspective of each individual database, the posterior probability that the user-required message index is  $\theta$ , conditioned on the query transmitted by the user, must be equal to the corresponding posterior probability with the user-required message index being  $\theta'$ , for all  $\theta' \neq \theta$ . This is known as the user privacy constraint. The user should be able to correctly retrieve the required message using the collection of answers from all the databases. This is known as the correctness constraint (also known as the decodability or reliability constraint), which formally states that there should be no uncertainty in the message retrieved.

The goal of the PIR problem is to design schemes that satisfy the privacy and correctness constraints while achieving the minimum possible download cost, equivalently, the largest possible PIR rate. The download cost of a PIR scheme is defined as the total number of bits downloaded by the user from all the databases, normalized by the message size. The PIR rate is defined as the reciprocal of the PIR download cost. The system model for PIR is shown in Fig 2, where a user wants to download the message  $W_\theta$ , without revealing the message index  $\theta$  to any of the databases.

## II. PIR SCHEMES

The first known PIR scheme that achieves information theoretic privacy is presented in [1]. This scheme is based on the concept of using a pair of databases to retrieve a single symbol of the required message as illustrated in Fig. 3.

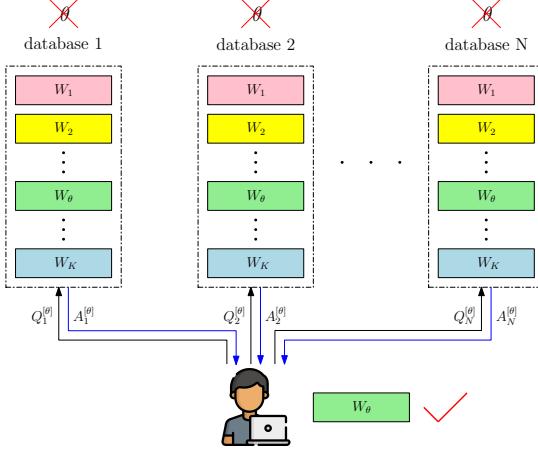
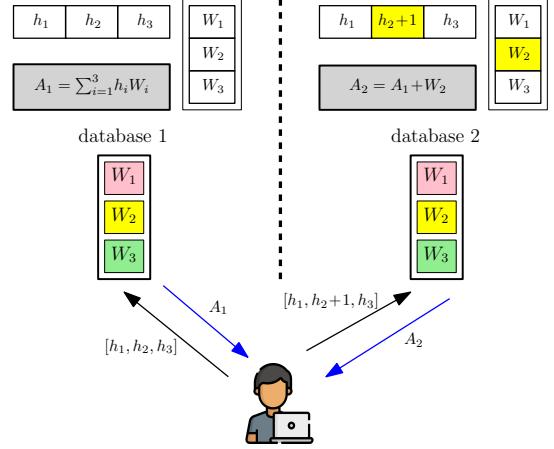


Fig. 2: The system model of PIR.

In the example in Fig. 3, the three single-symbol messages denoted by  $W_1, W_2, W_3$  are stored across two databases. These message symbols take values from a finite field  $\mathbb{F}_q$ . Assume that the user wants to download the second message  $W_2$ . As queries, the user sends  $K = 3$  randomly chosen symbols from  $\mathbb{F}_q$  to database 1, denoted by  $[h_1, h_2, h_3]$ . At the same time, the user sends  $[h_1, h_2 + 1, h_3]$  to database 2, where “+” is addition within the finite field  $\mathbb{F}_q$ ; for instance, within the binary field, it is the XOR operation. Upon receiving the query, each database simply computes the dot product of the query and the three message symbols, and sends the result back to the user as a single symbol, as shown in Fig. 3. Then, the user obtains the required message as  $W_2 = A_2 - A_1$ . User privacy is guaranteed since the databases are non-colluding, and each database simply receives a set of random symbols from  $\mathbb{F}_q$ . The rate of this scheme is  $R = \frac{1}{2}$ , as the user downloads two symbols for one privately received symbol.

Based on the concept introduced in [1], a more efficient PIR scheme that is compatible with arbitrary number of databases and arbitrary message lengths was proposed in [3]. This scheme basically improves the scheme in [1] by utilizing the same piece of side information (i.e.,  $A_1$  in Fig. 3) multiple times throughout the process, as opposed to only using it once in [1]. As an illustration, consider the following example with  $N = 3$  databases storing  $K = 2$  messages. The scheme is explained on a 2-symbol segment of each message, which is called a *subpacket*, and is applied on all such subpackets repeatedly in an identical manner. Let  $W_1 = (W_{1,1}, W_{1,2})$  and  $W_2 = (W_{2,1}, W_{2,2})$  be individual subpackets of the first and second messages, respectively, each consisting of two symbols from  $\mathbb{F}_q$ . Let  $h_{1,1}, h_{1,2}, h_{2,1}, h_{2,2}$  be four randomly and independently selected symbols from  $\mathbb{F}_q$ . Assume that the user wants to download the first message  $W_1$ .

The PIR scheme for this example is shown in Fig 4. The user sends the queries  $Q_1 = [h_{1,1}, h_{1,2}, h_{2,1}, h_{2,2}]$ ,  $Q_2 = [h_{1,1} + 1, h_{1,2}, h_{2,1}, h_{2,2}]$  and  $Q_3 = [h_{1,1}, h_{1,2} + 1, h_{2,1}, h_{2,2}]$  to databases 1, 2 and 3, respectively. Since the three databases are non-colluding, the user privacy follows from [1]. Each database computes the dot product of the received query and the four stored message symbols ( $W_{1,1}, W_{1,2}, W_{2,1}, W_{2,2}$ ), and sends it back to the user as the answer. The explicit

Fig. 3: PIR scheme in [1] for  $N = 2$  and  $K = 3$ .

expressions for the answers are shown in Fig. 4. The user obtains the two symbols of the user-required message via,

$$W_{1,1} = A_2 - A_1, \quad W_{1,2} = A_3 - A_1. \quad (1)$$

The rate achieved for this example is  $R = \frac{2}{3}$ , as two symbols are privately obtained by three downloads.

For general  $N$  and  $K$ , with this scheme, we can privately obtain  $N - 1$  symbols by a total of  $N$  downloads. Thus, the rate of this scheme is,

$$R = \frac{N - 1}{N} = 1 - \frac{1}{N}, \quad (2)$$

which leads to the following question: Is this the best achievable rate or can we do better than  $1 - \frac{1}{N}$ ?

This question is answered in [2], which first shows that the rate any valid PIR scheme for the general setting of  $N$  databases storing  $K$  messages is upper bounded by,

$$R \leq \left( 1 + \frac{1}{N} + \dots + \frac{1}{N^{K-1}} \right)^{-1}, \quad (3)$$

which is derived using fundamental bounds in information theory. Meanwhile, there does exist optimal PIR schemes that achieve the upper bound in (3) for any  $N$  and  $K$ , characterizing the capacity of PIR as,

$$C_{PIR} = \left( 1 + \frac{1}{N} + \dots + \frac{1}{N^{K-1}} \right)^{-1}. \quad (4)$$

Note that the PIR capacity  $C_{PIR}$  in (4) is strictly greater than the achievable rate  $R$  in (2).

To date, there are two primary information theoretic approaches towards achieving the capacity of PIR. The first is a *deterministic* approach, inspired by the idea of blind interference alignment [2]. The second is a *probabilistic* approach [4], [5], based on the idea that from the viewpoint of each database, each potential query is designed such that it could be used to retrieve any message in the message set with equal probability. These two types of schemes are described next.

The basic idea of the deterministic scheme in [2] is to enforce message symmetry within the queries, to prevent the databases from identifying the index of the user-desired message. At the same time, the queries should be carefully designed in such a way that the user is able to decode the

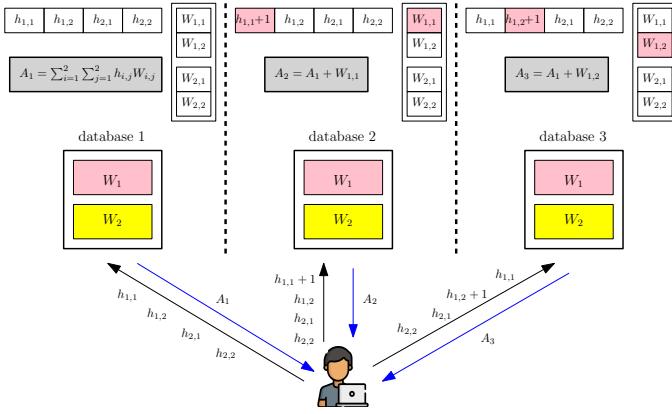


Fig. 4: PIR scheme in [3] for  $N = 3$  and  $K = 2$ .

desired message by exploiting the unwanted message bits (side information) downloaded from different databases.

This concept is illustrated in the following example with  $N = 2$  databases storing  $K = 2$  messages. This scheme requires the messages to be divided into *subpackets* of size  $N^K = 4$ . The subpackets corresponding to the first and second messages are denoted by  $(a_1, a_2, a_3, a_4)$  and  $(b_1, b_2, b_3, b_4)$ , respectively. In order to retrieve the desired message  $W_1$  or  $W_2$ , the user sends the corresponding queries to both databases with the aim of downloading the symbols as shown in Table I.

Retrieve $W_1$		Retrieve $W_2$	
DB 1	DB 2	DB 1	DB 2
$a_1$	$a_2$	$a_1$	$a_2$
$b_1$	$b_2$	$b_1$	$b_2$
$a_3 + b_2$	$a_4 + b_1$	$a_2 + b_3$	$a_1 + b_4$

TABLE I: Deterministic scheme in [2] for  $N = 2$ ,  $K = 2$ .

In Table I, to retrieve any message, the user first downloads a single symbol of each of the two messages from both databases. Then, the user downloads sums in the form of  $a+b$  from both databases to satisfy the message symmetry. If the desired message is  $W_1$ ,  $a$  is a new symbol of  $W_1$  and  $b$  is an already downloaded symbol from the other database and vice versa. Each unwanted download is used as side information in another database to increase the efficiency of the PIR process. User privacy is guaranteed by maintaining message symmetry among all types of queries, i.e., each individual database always receives queries requesting a single bit of each of the two messages, and a sum of two new bits of the two messages, irrespective of the user's message requirement.<sup>1</sup> The rate achieved in this example for any given message requirement is  $R = \frac{2}{3}$ , since 6 bits are downloaded in total to obtain 4 bits of the required message. This rate equals the capacity in (4) when  $N = 2$  and  $K = 2$ .

Another example with  $N = 2$  and  $K = 3$  is given in Fig. 5. The size of a subpacket in this example is  $N^K = 8$ , and the single subpackets of the three messages  $W_1$ ,  $W_2$  and  $W_3$  are denoted by  $(a_1, \dots, a_8)$ ,  $(b_1, \dots, b_8)$  and  $(c_1, \dots, c_8)$ , respectively. The rate achieved in this example is  $R = \frac{4}{7}$  for

<sup>1</sup>The subscripts of  $a$  and  $b$  are permuted by the user, prior to sending the queries to ensure that no information is leaked by them.

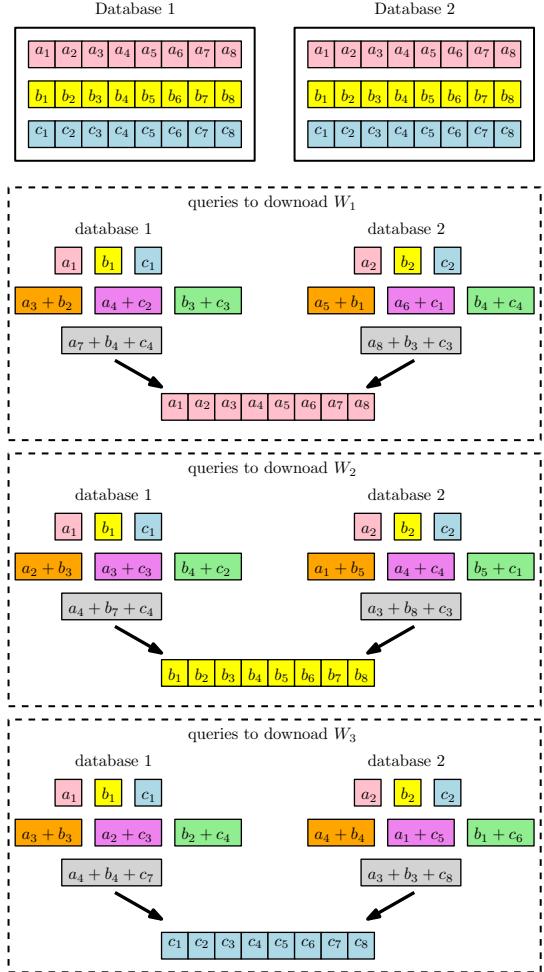


Fig. 5: Deterministic scheme [2] for  $N = 2$ ,  $K = 2$ .

any message requirement, which equals the capacity in (4) when  $N = 2$  and  $K = 3$ .

The probabilistic approach is based on a set of universal queries that are used for all message requirements with equal probability, in the perspective of an individual database. There are two main probabilistic PIR schemes in the literature, proposed in [4] and [5], which are described next. For the example of  $N = 2$  and  $K = 2$ , the set of possible queries sent to the two databases in the probabilistic approach from [4] is given in Table II, where  $W_1$  and  $W_2$  represent the first and second messages, respectively. To retrieve any message, the user selects one query option from the query set, i.e., one row of Table II, with equal probability, and transmits the corresponding queries to the two databases.

Prob.	Retrieve $W_1$		Retrieve $W_2$	
	DB 1	DB 2	DB 1	DB 2
$\frac{1}{4}$	$W_1$	—	$W_2$	—
$\frac{1}{4}$	—	$W_1$	—	$W_2$
$\frac{1}{4}$	$W_2$	$W_1 + W_2$	$W_1$	$W_1 + W_2$
$\frac{1}{4}$	$W_1 + W_2$	$W_2$	$W_1 + W_2$	$W_1$

TABLE II: Probabilistic scheme in [4] for  $N = 2$ ,  $K = 2$ .

User privacy is guaranteed since each individual database will always receive one query from 4 available query options

$\{W_1, W_2, W_1 + W_2, -\}$  with equal probability, irrespective of the user's message requirement (here, “-” corresponds to the “empty” query, i.e., “no” query). The rate achieved in this example for any message is,

$$R = \left( \frac{\frac{1}{4} \times L + \frac{1}{4} \times L + \frac{1}{4} \times 2L + \frac{1}{4} \times 2L}{L} \right)^{-1} = \frac{2}{3}, \quad (5)$$

where  $L$  is the message length. The expression in (5) is obtained by considering the fact that the first two query options in Tables II require only  $L$  symbols to be downloaded while the last two options require  $2L$  symbols. The inverse of the expected download cost is calculated as the rate in (5), which is the same as the capacity in (4) for  $N = 2$  and  $K = 2$ .

The probabilistic scheme proposed in [5] is based on the same principles as [4], except that it does not have the message symmetry present in [4]. As an illustration of the scheme in [5], consider the case of  $N = 2$  and  $K = 2$  again. As a reduced version of Table II, the explicit forms of the query options are provided in Table III.

Prob.	Retrieve $W_1$		Retrieve $W_2$	
	DB 1	DB 2	DB 1	DB 2
$\frac{1}{2}$	—	$W_1$	—	$W_2$
$\frac{1}{2}$	$W_1 + W_2$	$W_2$	$W_1 + W_2$	$W_1$

TABLE III: Probabilistic scheme in [5] for  $N = 2$ ,  $K = 2$ .

To illustrate the general mechanism in [5], consider an example with  $N = 3$  and  $K = 3$ . Each subpacket in this scheme consists of  $N - 1 = 2$  symbols of each message. Let a single subpacket of the first, second and third message be denoted by  $W_1 = (a_1, a_2)$ ,  $W_2 = (b_1, b_2)$  and  $W_3 = (c_1, c_2)$ , respectively. Assume that the user's required message is  $W_2$ . A dummy bit is appended to each subpacket in each message as,  $W_1 = (a_0, a_1, a_2)$ ,  $W_2 = (b_0, b_1, b_2)$  and  $W_3 = (c_0, c_1, c_2)$ , where  $a_0 = b_0 = c_0 = 0$ . The first step is to choose a random key of length  $K - 1 = 2$ , from the set  $\{0, \dots, N - 1\}^{K-1} = \{0, 1, 2\}^2$ . Assume that the chosen random key is  $F = (0, 2)$ . Then, the query sent to database  $n$  is of the form  $Q_n = (\alpha, \beta, \gamma)$ , where  $\alpha = 0$ ,  $\gamma = 2$ ,  $\beta = (n - 1 - \sum_{i=1}^2 F(i))_N$ , where  $(\cdot)_N$  is the modulo  $N$  operation. In other words, all elements except the second (required message index) in each length  $K$  query vector are copied from the random key  $F$  while the second element is chosen such that each query satisfies,

$$\left( \sum_{i=1}^3 Q_n(i) \right)_N = n - 1, \quad n \in \{1, 2, 3\}. \quad (6)$$

The three query vectors sent to the three databases are,

$$Q_1 = (0, 1, 2), \quad Q_2 = (0, 2, 2), \quad Q_3 = (0, 0, 2). \quad (7)$$

Once database  $n$  receives the query  $Q_n$ , it calculates the answer as,  $A_n = a_{Q_n(1)} + b_{Q_n(2)} + c_{Q_n(3)}$ . The answers sent by the three databases are given by,

$$A_1 = a_0 + b_1 + c_2, \quad A_2 = a_0 + b_2 + c_2, \quad A_3 = a_0 + b_0 + c_2, \quad (8)$$

from which the user can find the two bits of  $W_2$  as,

$$b_1 = A_1 - A_3, \quad b_2 = A_2 - A_3, \quad (9)$$

since  $b_0 = 0$ . All possible random keys and the corresponding queries and answers of each database, when downloading  $W_2$ , are shown in Table IV.

$F$	DB 1		DB 2		DB 3	
	$q_1$	$A_1$	$q_2$	$A_2$	$q_3$	$A_3$
00	000	$a_0 + b_0 + c_0$	010	$a_0 + b_1 + c_0$	020	$a_0 + b_2 + c_0$
10	120	$a_1 + b_2 + c_0$	100	$a_1 + b_0 + c_0$	110	$a_1 + b_1 + c_0$
20	210	$a_2 + b_1 + c_0$	220	$a_2 + b_2 + c_0$	200	$a_2 + b_0 + c_0$
01	021	$a_0 + b_2 + c_1$	001	$a_0 + b_0 + c_1$	011	$a_0 + b_1 + c_1$
11	111	$a_1 + b_1 + c_1$	121	$a_1 + b_2 + c_1$	101	$a_1 + b_0 + c_1$
21	201	$a_2 + b_0 + c_1$	211	$a_2 + b_1 + c_1$	221	$a_2 + b_2 + c_1$
02	012	$a_0 + b_1 + c_2$	022	$a_0 + b_2 + c_2$	002	$a_0 + b_0 + c_2$
12	102	$a_1 + b_0 + c_2$	112	$a_1 + b_1 + c_2$	122	$a_1 + b_2 + c_2$
22	222	$a_2 + b_2 + c_2$	202	$a_2 + b_0 + c_2$	212	$a_2 + b_1 + c_2$

TABLE IV: Probabilistic scheme in [5] for  $N = 3$ ,  $K = 3$  to download message  $W_2$ .

Each random key is chosen with equal probability, and all except the first key requires the user to download one symbol each from all three databases while the first key requires the user to download a single symbol from only databases 2 and 3, since  $a_0 = b_0 = c_0 = 0$  is globally known. Therefore, the rate achieved in this example is given by,

$$R = \left( \frac{\frac{1}{9} \times 2 + \frac{1}{9} \times 3 \times 8}{2} \right)^{-1} = \frac{9}{13}, \quad (10)$$

which is exactly the capacity in (4) when  $N = 3$ ,  $K = 3$ . From the perspective of an individual database, each query from the possible sets of queries in Table IV is received with equal probability irrespective of the user's message requirement, which guarantees user privacy. This can be seen by comparing each  $q_i$  column in Tables IV and V for downloading message  $W_2$  and  $W_1$ , respectively. We left out the corresponding table for downloading message  $W_3$  for space limitations.

$F$	DB 1		DB 2		DB 3	
	$q_1$	$A_1$	$q_2$	$A_2$	$q_3$	$A_3$
00	000	$a_0 + b_0 + c_0$	100	$a_1 + b_0 + c_0$	200	$a_2 + b_0 + c_0$
10	210	$a_2 + b_1 + c_0$	010	$a_0 + b_1 + c_0$	110	$a_1 + b_1 + c_0$
20	120	$a_1 + b_2 + c_0$	220	$a_2 + b_2 + c_0$	020	$a_0 + b_2 + c_0$
01	201	$a_2 + b_0 + c_1$	001	$a_0 + b_0 + c_1$	101	$a_1 + b_0 + c_1$
11	111	$a_1 + b_1 + c_1$	211	$a_2 + b_1 + c_1$	011	$a_0 + b_1 + c_1$
21	021	$a_0 + b_2 + c_1$	121	$a_1 + b_2 + c_1$	221	$a_2 + b_2 + c_1$
02	102	$a_1 + b_0 + c_2$	202	$a_2 + b_0 + c_2$	002	$a_0 + b_0 + c_2$
12	012	$a_0 + b_1 + c_2$	112	$a_1 + b_1 + c_2$	212	$a_2 + b_1 + c_2$
22	222	$a_2 + b_2 + c_2$	022	$a_0 + b_2 + c_2$	122	$a_1 + b_2 + c_2$

TABLE V: Probabilistic scheme in [5] for  $N = 3$ ,  $K = 3$  to download message  $W_1$ .

Based on the PIR schemes discussed above, the suboptimality of the scheme in [3] (shown in Fig. 4) is caused by the inability of the users to explicitly download any non-required message symbols. Note that, the optimal scheme in [2] (shown in Fig. 5) lets the user download symbols from all messages explicitly, and use them as different pieces of side information, which allows the user to download multiple symbols of the required message privately. In contrast, the suboptimal scheme in [3] only generates one piece of side information ( $A_1$  in

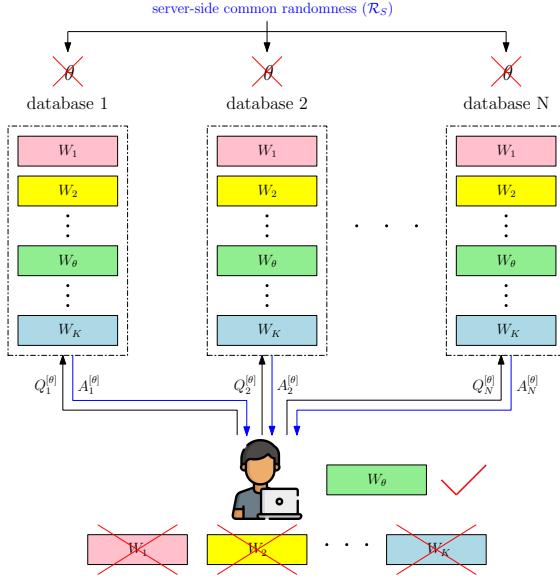


Fig. 6: The system model of SPIR.

Fig. 4), which limits the amount of required message bits that the user is allowed to download privately. However, the suboptimal scheme in [3] preserves the privacy of messages that are not required by the user to a certain extent, by not allowing the user to download any non-required message bits explicitly, as opposed to the optimal schemes. We will see that this will come handy in symmetric privacy formulation next.

### III. SPIR FORMULATION AND SCHEMES

The concept of maintaining a two-way privacy requirement, where the databases are not allowed to learn the user's required message index while the user is not allowed to learn any information of the non-required messages is referred to as *symmetric PIR (SPIR)* [6], which is a non-trivial extension of PIR. In SPIR, the symmetry comes from the fact that the privacy of the user and the databases are desired to be guaranteed simultaneously; see Fig. 6. The additional requirement that prohibits the user from learning anything beyond the required message is known as the database privacy constraint. It is proven that the user privacy constraint, database privacy constraint and correctness constraint in SPIR jointly form a contradiction when no additional parameters are utilized [6], [7]. A well-known approach to perform SPIR is to introduce shared server-side common randomness at the databases in the server, that is unknown to the user.

The SPIR scheme in [7] is constructed on the basis of the PIR schemes provided in [1] and [3] by appending extra server-side common randomness to the answers. Consider an SPIR example with \$N = 2\$ and \$K = 3\$, and assume that the user requires to download the second message \$W\_2\$. The user sends three randomly selected symbols from \$\mathbb{F}\_q\$, denoted by \$[h\_1, h\_2, h\_3]\$ to database 1, and \$[h\_1, h\_2+1, h\_3]\$ to database 2. After receiving the query, each database calculates the dot product of the query and its own message set first, and adds a server-side common randomness symbol \$S\$ uniformly selected from \$\mathbb{F}\_q\$ to the dot product. Then, each database transmits this information as answers back to the user. This process is

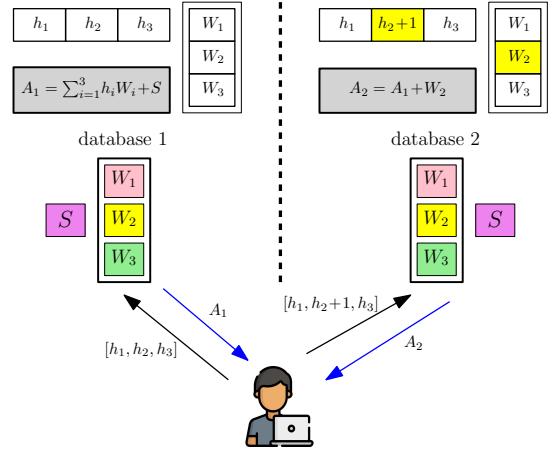


Fig. 7: SPIR scheme in [7] for \$N = 2\$ and \$K = 3\$.

shown in Fig. 7. User privacy and correctness constraints are satisfied the same way as in PIR (Section II). Database privacy is satisfied as the user is unable to learn any information about the other messages due to the existence of the unknown symbol \$S\$. The rate achieved in this example is \$R = \frac{1}{2}\$, which is the same as the rate of the example shown in Fig. 3.

Following the generalized PIR scheme in [3], this example can be generalized to SPIR with arbitrary \$K\$ and \$N\$ by adding server-side common randomness to each answer. The corresponding rate is \$R = 1 - \frac{1}{N}\$ as in (2). Using the converse result in [7], this rate is also the capacity of SPIR as long as sufficient server-side common randomness is available, i.e.,

$$C_{\text{SPIR}} = 1 - \frac{1}{N}. \quad (11)$$

Note that the capacity of SPIR does not depend on the number of messages \$K\$ and is strictly smaller than the capacity of PIR, i.e., \$C\_{\text{SPIR}} < C\_{\text{PIR}}\$, because of the additional database privacy constraint imposed. In addition, note that PIR capacity \$C\_{\text{PIR}}\$ decreases with the number of messages \$K\$, and \$C\_{\text{SPIR}} = \lim\_{K \rightarrow \infty} C\_{\text{PIR}}\$. Interestingly, it was shown in [8] that the SPIR capacity can be increased to the PIR capacity if the user is able to pre-fetch a random subset of server-side common randomness (sufficient amount) from the server.

The SPIR scheme presented above is deterministic. Using the probabilistic PIR scheme in [5], an alternative probabilistic SPIR approach is given in [9]. For \$N = 2\$ and \$K = 2\$, the set of possible queries in this approach is given in Table VI, where \$S\$ is the server-side common randomness symbol only known by the databases. User privacy and correctness conditions are guaranteed the same way as in [5], as described in Section II. Database privacy is satisfied due to the unknown symbol \$S\$. A general capacity-achieving SPIR scheme can be achieved by adding server-side common randomness to each query option in the generalized PIR scheme in [5].

Prob.	Retrieve \$W_1\$		Retrieve \$W_2\$	
	DB 1	DB 2	DB 1	DB 2
\$\frac{1}{2}\$	\$S\$	\$W_1+S\$	\$S\$	\$W_2+S\$
\$\frac{1}{2}\$	\$W_1+W_2+S\$	\$W_2+S\$	\$W_1+W_2+S\$	\$W_1+S\$

TABLE VI: Probabilistic SPIR scheme [9] for \$N = 2, K = 2\$.

#### IV. SYSTEMATIC EXTENSIONS OF PIR

In this section, we briefly describe the problem formulations and capacity results of some variants of PIR.

- 1) PIR with coded databases [10]: This problem considers  $N$  non-colluding databases storing  $K$  messages that are  $(N, M)$  MDS coded. The capacity of coded PIR is,

$$C_{coded} = \left( 1 + \frac{M}{N} + \cdots + \frac{M^{K-1}}{N^{K-1}} \right)^{-1}, \quad (12)$$

which is a generalization of the capacity of classical PIR with replicated storage. Replication corresponds to the special case of  $M = 1$ .

- 2) PIR with colluding databases [11]: This problem considers a system of  $N$  databases where up to  $T$  of them can collude. The capacity of colluded PIR is,

$$C_{colluded} = \left( 1 + \frac{T}{N} + \cdots + \frac{T^{K-1}}{N^{K-1}} \right)^{-1}, \quad (13)$$

which is equivalent to the capacity of classical PIR with  $\frac{N}{T}$  non-colluding databases.

- 3) PIR with Byzantine and colluding databases [12]: This problem considers the presence of  $B$  Byzantine databases out of the  $N$  databases while any  $T$  databases are allowed to collude. The capacity is given by,

$$C_B = \frac{N-2B}{N} \left( 1 + \frac{T}{N-2B} + \cdots + \frac{T^{K-1}}{(N-2B)^{K-1}} \right)^{-1} \quad (14)$$

which is equivalent to removing  $2B$  databases out of the  $N$  databases in a  $T$ -colluding setting. The scaling factor represents the fact that only  $N - 2B$  databases are of actual use even though all  $N$  databases are accessed.

- 4) Multi-message PIR (MM-PIR) [13]: Here the user wants to download  $P$  out of  $K$  messages at a time, without revealing their identities to any of the  $N$  databases. The capacity is given by,

$$C_{MM-PIR} = \begin{cases} \frac{1}{1 + \frac{K-P}{PN}}, & \text{if } P \geq \frac{K}{2} \\ \frac{1 - \frac{P}{N}}{1 - (\frac{1}{N})^{\frac{K}{P}}}, & \text{if } P \leq \frac{K}{2}, \frac{K}{P} \in \mathbb{N} \end{cases} \quad (15)$$

The capacity when  $P \leq \frac{K}{2}$  and  $\frac{K}{P} \in \mathbb{N}$  is equal to the capacity of classical PIR with  $\frac{K}{P}$  messages. This shows that using MM-PIR once is more efficient than using classical PIR  $P$  times, to download  $P$  messages.

- 5) Assymmetric leaky PIR (AL-PIR) [14]: This variant of PIR studies the potential increase in the capacity, when a pre-determined amount of information is allowed to leak. AL-PIR considers both user and database privacy (SPIR), and proposes a scheme that performs under arbitrary information leakage budgets.

#### V. CONCEPTUAL EXTENSIONS OF PIR

##### A. Private Set Intersection (PSI)

Private set intersection (PSI) refers to the problem in which two parties  $P_1$  and  $P_2$  wish to determine the common elements jointly within their element sets without leaking any further information to each other about the remaining elements in

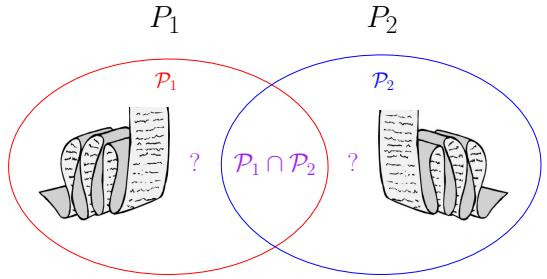


Fig. 8: The system model for two-party PSI.

their sets [15]. The element sets associated with  $P_1$  and  $P_2$  are denoted by  $\mathcal{P}_1$  and  $\mathcal{P}_2$ , which are selected from a global alphabet  $\mathcal{A}$ , based on an arbitrary statistical distribution. The basic two-party PSI system model is shown in Fig. 8.

The problem of PSI is motivated by practical security applications. For instance, consider a situation where an airline company has a list of its passengers while the national security agency (NSA) has a list of its suspected terrorists. NSA wants to check whether any of the terrorists is boarding a flight without revealing the entire list of terrorist suspects. Meanwhile, the airline company wishes to identify the terrorist suspects without revealing the entire list of its customers. Both parties are making an effort to determine the intersection of their respective lists in a private manner. Hence, three requirements are involved in the two-party PSI: First, at least one of the two parties should be able to decode the intersection correctly when the PSI process is complete. This requirement is called the PSI correctness constraint. Second and third, the privacy of the remaining elements in  $\mathcal{P}_1$  and  $\mathcal{P}_2$  must be guaranteed. These last two requirements are called the PSI  $P_1$  and  $P_2$  privacy constraints, respectively.

As discussed in Section III, there are three fundamental requirements in the SPIR problem, namely, correctness, user privacy, and database privacy. It was shown in [16] that the three requirements of the PSI map one-to-one to the three requirements of the SPIR. Precisely, assuming without loss of generality that  $P_1$  initiates the PSI process,  $P_1$  privacy in PSI corresponds to user privacy in SPIR;  $P_2$  privacy in PSI corresponds to database privacy in SPIR; and correctness in determining the set intersection in PSI corresponds to correctness in decoding in SPIR. Here, note that  $P_1$  wishes to learn whether its own elements are also in  $P_2$  without revealing the identity of its own elements (user privacy) and without learning anything further than the existence of those particular elements in  $P_2$  (database privacy). Further, note that, since  $P_1$  has multiple elements, it will want to check the existence of these multiple elements all at once, considering the fact that multi-message SPIR (MM-SPIR) may be more efficient than multiple application of single-message SPIR, as it happened in the case of multi-message PIR. Therefore, PSI is exactly equivalent to MM-SPIR. As a result, all known converse results as well as the existing achievable schemes for MM-SPIR translate directly into PSI.

Finally, the PSI problem (and the equivalent MM-SPIR) can be extended to information theoretic secure multi-party PSI (MP-PSI), where multiple (more than two) parties wish to jointly determine the intersection of their respective element

sets while protecting the privacy of their remaining elements [17]. This extension is achieved in [17] via the introduction of an intricate common randomness distribution scheme among the multiple parties before the MP-PSI process starts.

### B. Private Set Union (PSU)

Following the discussion in the last subsection, as a dual problem of PSI, private set union (PSU) refers to the problem in which two parties aim to compute the union of their element sets jointly without revealing anything beyond the union to each other [18]. Similar to the two-party PSI problem, there are three underlying requirements in the two-party PSU problem formulation: At least one party should be able to obtain the union without any error; this is called the PSU correctness constraint. The privacy of the remaining elements in  $P_1$  needs to be kept against  $P_2$ ; this is the PSU  $P_1$  privacy constraint. The privacy of the remaining elements in  $P_2$  needs to be kept against  $P_1$ ; this is the PSU  $P_2$  privacy constraint. The duality between PSU and PSI can be seen through the De Morgan's law:  $\overline{A \cup B} = \overline{A} \cap \overline{B}$ . Thus, we have  $A \cup B = \overline{\overline{A} \cap \overline{B}}$ , which implies that the set union can be found using a combination of set intersection and set complement. Therefore, by mapping SPIR correctness constraint, user privacy constraint and database privacy constraint to PSU correctness constraint,  $P_1$  privacy constraint and  $P_2$  privacy constraint respectively, the equivalence of PSU and SPIR is established in [19], which says that PSU is equivalent to MM-SPIR. Moreover, by using the scheme in [17] for reference, [18] shows that basic two-party PSU can be generalized to multi-party PSU (MP-PSU) in an information-theoretic secure sense.

### C. PSU-based Federated Submodel Learning (FSL)

Federated learning (FL) [20] is a framework where a central machine learning model is collectively trained by a large number of clients, using their local data. In FL, each client downloads the entire model, updates it, and uploads the updates back to the central server. This process is inefficient in terms of the communication and computation overhead, since each client downloads and uploads the entire model even if the data available at a given client is not sufficient to train the entire model. One solution to this problem is federated submodel learning (FSL) [21]–[23], where the central learning model is divided into multiple submodels based on different types of training data, as this allows each client to only download and update the submodel(s) relevant to the client's local data. However, the submodel indices requested by a client and the values of updates uploaded leak information about the client's private data. Hence, to ensure the privacy of the client's local data in FSL, the following two questions need to be answered. How can the clients download the desired submodels without revealing their indices to the server that stores the submodels? How can the clients update the desired submodels without revealing their indices or the values of the updates to the server? The first question refers to a *private read* problem which is the same as PIR, and the second question refers to a *private write* problem which is an important conceptual extension of PIR.

In classical FL, secure aggregation is used to guarantee privacy of the clients' data, in which the server can only learn the aggregate model from the clients and nothing beyond that [24]. As discussed in the last subsection, in MP-PSU, if one of the parties is selected as a leader party to derive the union, the MP-PSU privacy constraints require that this leader party can obtain only the union from the remaining parties and nothing beyond that. Therefore, [19] proposes a new private FSL approach which is based on PSU and SPIR. The core idea behind this scheme is to preserve client-side privacy by not allowing the server to learn any information beyond the subset of submodels (or the aggregation of updates of these submodels) collectively updated by the clients, i.e., the server learns nothing about any individual client's contribution to the ultimate result. Thus, no extra information about the clients' local training data is leaked to the server.

The sketch of this scheme is as follows. Within the initialization stage, two independent databases in the server store the replicated storage of all submodels and a sufficient amount of server-side common randomness.<sup>2</sup> Initially, the two databases distribute/duplicate client-side common randomness to each selected client using random SPIR [25] and classical one-time pads. Next, the clients utilize the established client-side common randomness to have the server reliably decode the union of indices of submodels to be updated collectively by the clients in a private manner. This phase is referred to as the FSL-PSU-read phase. Then, the two databases broadcast the current versions of the submodels in the set union to the clients. The clients update the desired submodels using their local training data. Finally, the clients use a variation of FSL-PSU to write the updates back to the databases privately. This phase basically performs secure aggregation, and is referred to as the PSU-write phase.

To improve the communication efficiency, instead of enforcing each client to send the same answer to both databases, all the selected clients are divided into two groups such that most of them merely communicate with the database they belong to. A very small subset of clients is randomly selected as intermediaries to route the information received by the two databases from their associated clients, to compensate for the absence of communication between the two databases. By utilizing the server-side and client-side common randomness carefully, the private FSL scheme in [19] is shown to be robust against client drop-outs, client late-arrivals and database drop-outs. Client drop-outs occur when some clients leave during the training process, and client late-arrivals occur when some clients come up with late answers that may leak additional information about these late clients to the databases because of the wrong judgement made by the server that these clients have already dropped-out. Database drop-outs occur when databases do not function smoothly at a given time instance.

### D. Private Read-Update-Write (PRUW)

Private read-update-write (PRUW) [26]–[29] is the concept where a user downloads, updates and uploads the updates back

<sup>2</sup>The case of two databases is considered here, and the achievable scheme works for any number of databases with minor modifications.

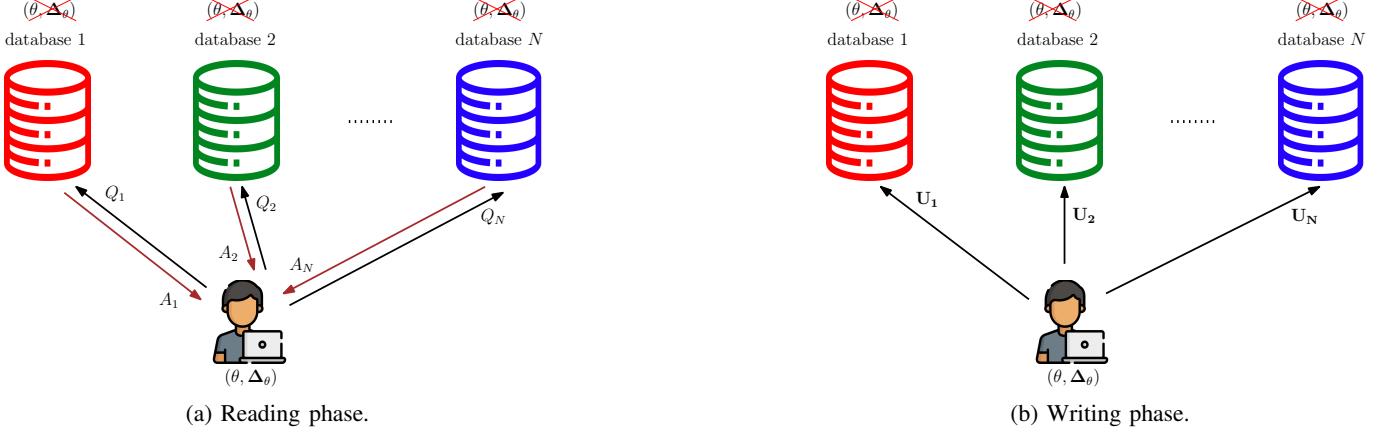


Fig. 9: System model of private FSL with PRUW.

to a chosen section of a data storage system, without revealing the content downloaded, uploaded or their positions, e.g., updated section index. PRUW is essentially the combination of *private reading* and *private writing* described previously. PRUW has two main applications in distributed learning, namely, private FSL and private FL with sparsification.

Private FSL, as discussed before, requires a client to download a required submodel, update it and upload the updates back to the submodel without revealing the submodel index or the values of updates. The method proposed in [19] achieves it by preventing the databases from learning any information beyond the union of submodels downloaded/updated by all clients. PRUW achieves stronger privacy guarantees in FSL by leaking zero information about any of the submodel indices or values of updates, updated by any individual or collection of clients. In particular, at any given time instance of the FSL process, the server has zero information on the submodels that have been updated so far, or the values of the updates.

The other application of PRUW is private FL with sparsification. Gradient sparsification [30]–[36] is a mechanism used in most learning tasks to reduce the communication overhead. In gradient sparsification, each client only uploads a selected set of updates (e.g., most significant, randomly chosen, etc.), along with their indices to the server to reduce the upload cost. In cases where the updates are selected based on their significance, the indices of the sparse updates leak information about the types of data that the client has. Moreover, it has been shown in [37]–[43] that the values of the updates can be used to infer information about the client's local data. This problem is solved by PRUW, which facilitates private communication of the sparse updates and parameters, by hiding their values and indices from the servers.

Next we describe how information-theoretic privacy is achieved by PRUW in FSL and FL with top  $r$  sparsification, followed by some other variants of PRUW.

**1) PRUW in FSL:** As explained in Section V-C, private FSL refers to the problem where a client reads (downloads), updates and writes (uploads) the updates back to a desired submodel out of multiple submodels in the FSL model, without revealing the submodel index or the values of updates. The two phases in the FSL process, where the client reads the parameters of the required submodel and writes the updates back to the

submodel are known as the reading phase and the writing phase, respectively. Formally, the privacy constraint on the submodel index requires the mutual information between the desired submodel index  $\theta$  and all the information sent by the client in both reading and writing phases at all time instances to be equal to zero, in the perspective of any individual database. Similarly, the privacy constraint on the values of updates requires the mutual information between the values of updates and all the information sent by the client at all time instances to be equal to zero. Moreover, there exists a security constraint, which prevents each individual database from learning any information about the submodels from its stored content. This is required for the privacy of the values of updates, since the changes in submodels at distinct time instances directly reveal information about the values of updates. The correctness constraint in the reading phase is the same as that of PIR, while the writing phase at time  $t$  requires the updating submodel at time  $t - 1$  to be added with the corresponding updates, while all other submodels remain the same. The reading and writing costs are defined as the total number of bits downloaded and uploaded, normalized by the size of a submodel. The total cost is the sum of the reading and writing costs.

To solve the basic private FSL problem, [26]–[29] use the system model shown in Fig. 9, consisting of  $N$  non-colluding databases, each storing  $M$  independent submodels consisting of symbols from a large enough finite field  $\mathbb{F}_q$ . Similar to PIR, each client sends queries to databases to download the required submodel in the reading phase, for which the databases send answers, as shown in Fig. 9a. In the writing phase, the client sends updates to all databases, which are added to the relevant submodel parameters, without revealing the submodel index  $\theta$  or the values of updates  $\Delta_\theta$ , see Fig. 9b. All schemes presented in [26]–[29] are based on the concept of cross subspace alignment [44]. These schemes are designed along the lines of private distributed coded computing, where the submodels, queries and updates are secured by adding random noise from the finite field in a specific way, which essentially makes them random noise variables from the perspective of an individual database, based on Shannon's one-time pad theorem [45]. Shannon's one-time pad theorem basically states that if  $X$  is a random variable with any arbitrary distribution, and  $Y$  is an

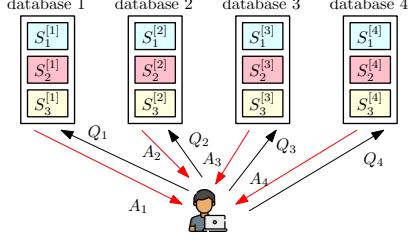


Fig. 10: PRUW in FSL: Reading phase.

independent and uniformly distributed random variable, both in the same finite field  $\mathbb{F}_q$ , the random variable  $Z = X + Y$  is uniformly distributed, and is independent of  $X$ .

The basic private FSL scheme proposed in [26], [28] adds random noise to all types of information that need to be kept private (submodel index, updates, submodels in storage), to guarantee information-theoretic privacy. This noise addition is done in terms of a carefully designed polynomial, and what is stored/sent to each database is a unique evaluation of this noise added polynomial. Each individual polynomial evaluation sent to a given database is simply random noise, while the collection of all  $N$  evaluations reveals the hidden value. In other words, the storage, queries and updates corresponding to each database are shares of secrets [46]. All computations in both reading and writing phases are carried out on these noisy polynomial evaluations at each database, which are designed in such a way that all the noise components resulting from the computations are aligned along a specific subspace in an  $N$  dimensional space, while the data components are aligned along specific directions that are linearly independent of the noise subspace. This makes it possible for a client to download the required submodel by only accessing the specific direction allocated to it in the  $N$  dimensional space in the reading phase, and to write the real updates back to the specific direction, while also writing additional noise elements into the noise subspace, to guarantee privacy in the writing phase.

As an example, consider an FSL system where four non-colluding databases store three submodels, denoted by  $W_1$ ,  $W_2$  and  $W_3$ . The three submodels are *secrets* that need to be hidden by the databases that store them. Therefore, they must be stored as shares of secrets in the four databases. Let the secret share corresponding to  $W_k$ , stored in database  $n$  be denoted by  $S_k^{[n]}$ , as shown in Fig. 10. The explicit form of  $S_k^{[n]}$  is given by,

$$S_k^{[n]} = w_k + (f - \alpha_n)(Z_k + \alpha_n Y_k) \quad (16)$$

where  $w_k$  denotes a single bit of  $W_k$ .  $f$  and  $\alpha_n$  are globally known distinct constants and  $Z_k$ ,  $Y_k$  are random noise bits from  $\mathbb{F}_q$ . Note that what is stored in database  $n$  is the evaluation of (16) at the corresponding  $\alpha_n$ ,<sup>3</sup> which is basically random noise due to the added noise component, from Shannon's one-time pad theorem. Assume that the client wants to download  $W_2$ . In the reading phase, the client sends queries to the four databases which are also shares of the secret that contains the required submodel index. The secret, i.e., submodel index 2, is represented by the vector  $[0 \ 1 \ 0]^T$ ,

<sup>3</sup>Each database has a distinct  $\alpha_n$  associated with it.

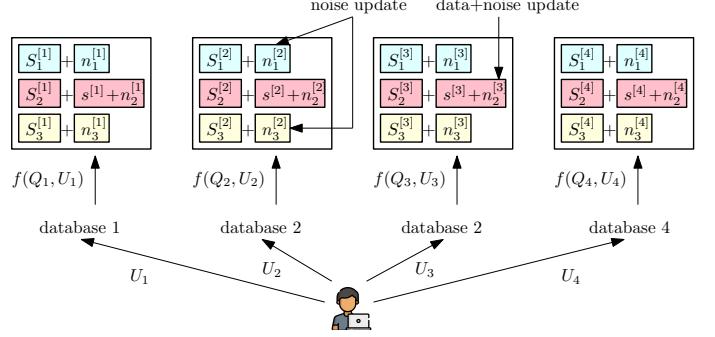


Fig. 11: PRUW in FSL: Writing phase.

since there are only three submodels, and the query sent to database  $n$  is given by,

$$Q_n = \frac{1}{f - \alpha_n} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} \bar{Z}_1 \\ \bar{Z}_2 \\ \bar{Z}_3 \end{bmatrix}, \quad (17)$$

where  $\bar{Z}_i$  are random noise bits. The evaluation of (17) at the respective  $\alpha_n$  is sent to each database. These are also random noise vectors due to the added noise, which guarantees the privacy of the required submodel index. Once the databases receive the queries from the client, they calculate the answers as the dot products of the storage and the received queries. The explicit form of the answer calculated by database  $n$  is,

$$A_n = \frac{1}{f - \alpha_n} w_2 + V_0 + \alpha_n V_1 + \alpha_n^2 V_2, \quad (18)$$

where  $V_0, V_1, V_2$  are obtained by combining all constant terms and coefficients of  $\alpha_n^i$  terms for  $i = 1, 2$ , respectively, of the dot product. The four answers obtained by the four databases, i.e., evaluation of (18) at four distinct  $\alpha_n$ s are used to retrieve  $w_2$ , along with  $V_0$ ,  $V_1$  and  $V_2$ .

In the writing phase, the client sends the update of  $w_2$  denoted by  $\Delta_2$  to the four databases as the evaluations of,

$$U_n = \Delta_2 + (f - \alpha_n) \dot{Z}, \quad (19)$$

where  $\dot{Z}$  is a random noise bit. The evaluations of (19) sent to the databases are simply random noise bits due to the added random noise, which guarantees the privacy of the update  $\Delta_2$ . At each database, the received noisy update is placed at the relevant position, i.e., at the second row in Fig. 10, with the aid of the query received in the reading phase as,

$$\bar{U}_n = (f - \alpha_n) \times U_n \times Q_n, \quad (20)$$

which is the incremental update that is added to the existing storage in each database. In this process, the noise component of the shares of secrets corresponding to  $W_1$  and  $W_3$ , stored in each database get updated while both the data and noise components get updated in the secret shares corresponding to  $W_2$  in all databases. This is shown in Fig. 11.

The efficiency of PRUW in FSL increases with the number of databases by allowing multiple updates to be combined to a single bit in the writing phase, which significantly reduces the writing cost. With large enough number of databases, PRUW in FSL can be performed by only downloading and uploading twice as many bits as the size of a submodel.

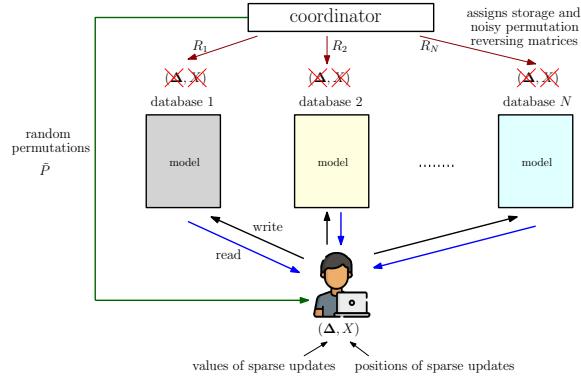


Fig. 12: PRUW in FL with sparsification.

2) *PRUW in FL With Top  $r$  Sparsification*: In FL with top  $r$  sparsification, clients only download and upload the most significant  $r'$  and  $r$  fractions of parameters and updates, respectively, to reduce the communication cost. Clients typically send the sparse updates along with their indices to databases when privacy is not a concern. However, as discussed before, revealing the values and the indices of the sparse updates compromise the client's privacy. Motivated by these privacy concerns, the problem of PRUW in FL with top  $r$  sparsification is introduced in [26], [47], [48]. The privacy constraint on the downloaded (uploaded) parameter (update) indices requires the mutual information between the real indices and all the information received by an individual database to be equal to zero. Privacy on the values of updates and security of the model parameters are the same as in PRUW in FSL. The correctness constraint in the reading phase requires the clients at time  $t$  to download the most commonly updated  $r'$  fraction of parameters at time  $t - 1$ , without revealing their indices, and the correctness constraint in the writing phase requires the sparse updates to be added to the respective real indices of the model while the rest of the parameters remain the same. The system model is shown in Fig. 12, for the communication between a single client and  $N$  non-colluding databases.

PRUW facilitates the sparse read-write process without revealing the values or the indices of the sparse updates [47], [48]. The values of the sparse updates are protected by adding random noise, as explained in Section V-D1. The privacy of the indices of sparse updates is achieved by considering a random permutation of all parameters of the model which is only known by the clients. The clients send the indices of sparse updates using this permuted order. In order to rearrange the updates in the correct order at the databases for correctness, a noise added permutation reversing matrix is stored at each database, corresponding to the chosen random permutation, from which no information on the underlying permutation can be learned by the databases. The noise added random permutation matrix for database  $n$  is denoted by  $R_n$  in Fig. 12.

The random permutation selection and the placement of the noisy permutation reversing matrix at each database is done at the initialization stage, before the learning process begins, with the help of a third party called the “coordinator” as shown in Fig. 12, which is also used in general PRUW to initialize the storage at each database prior to the learning process. The noisy permutation reversing matrix, along with the permuted

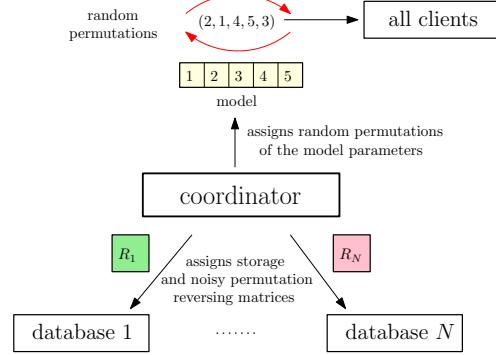


Fig. 13: The private permutation mechanism.

sparse update indices are used to privately rearrange the sparse updates in the correct order, so that the updates are correctly added to the respective parameters.

Consider a conceptual example where an FL model consists of five parameters as shown in Fig. 13. At the initialization stage, the coordinator sends a random permutation of the five parameters to all the clients involved in the learning process. Assume that this randomly chosen permutation is given by  $\tilde{P} = (2, 1, 4, 5, 3)$ . At the same time, the database stores a noise added permutation-reversing matrix at each database. These matrices are also shares of a secret from which the databases are unable to learn anything about the underlying permutation. The noise added permutation-reversing matrix stored at database  $n$  is of the form,<sup>4</sup>

$$R_n = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n X, \quad (21)$$

where  $X$  is a random noise matrix of size  $5 \times 5$ . Note that the first part of (21) is essentially the permutation reversing matrix corresponding to  $\tilde{P}$ , which is hidden from the databases by the added noise component, which ensures the privacy of the random permutation.

To illustrate how the updates are uploaded privately without revealing their values or indices, consider an example where a client wants to update parameters 2 and 3. Then, the client sends the noise added updates of the form (19) along with their permuted indices 1 and 5, since the real indices 2 and 3 are positioned at the first and fifth positions in  $\tilde{P}$ . Note that the values of the updates are protected by the noise added in (19). The privacy of the sparse indices 2 and 3 is guaranteed since the permutation  $\tilde{P}$  is randomly chosen. Once each database receives the two updates  $U_2$  and  $U_3$  (corresponding to the real indices 2 and 3) along with the permuted indices 1 and 5, it calculates the privately rearranged updates by multiplying the noise added permutation-reversing matrix of the form (21) by  $[U_2 \ 0 \ 0 \ U_3]^T$ . Note that this multiplication results in “[0  $U_2$   $U_3$  0 0]<sup>T</sup> + noise”, where the two updates corresponding to real indices 2 and 3 are correctly rearranged at the respective positions, while leaking no information about the real indices. Privately rearranged updates are added to the existing storage.

<sup>4</sup>The exact noisy permutation-reversing matrix is a scaled version of (21).

In the reading phase, the databases choose the most popular  $r'$  fraction of (permuted) indices received by the clients at the previous time instance, and send them to the clients. Note that the databases are still unaware of the corresponding real indices since the selection is done based on the permuted indices. Each client obtains the real indices corresponding to the received permuted indices using the known random permutation  $\tilde{P}$ . Now, for the client to access the real parameters corresponding to the permuted indices chosen by the databases, each database calculates the dot product between its storage, i.e., the five noise-added parameters in the Fig. 13, and the column of the noise added permutation-reversing matrix corresponding to each chosen permuted index, and sends it to the client. The client obtains the parameters from the answers received by all databases. Note that private FSL with top  $r$  sparsification [26], [30] can be performed by combining the above concepts with the scheme explained in Section V-D1. The resulting reading and writing costs of both private FL and FSL with top  $r$  sparsification when  $N$  is large, is approximately  $2r'$  and  $2r$ , respectively, which is significantly small since  $r', r \approx 10^{-2}$  in practice.

A main drawback of this process is the large storage cost caused by the significantly large noise-added permutation reversing matrices, which increases with the number of parameters in the FL model. The size of the noise added permutation-reversing matrices can be reduced at the expense of a certain amount of information leakage. This is achieved by dividing the FL model into multiple segments and carrying out permutations in each segment separately [47]–[49]. This idea is illustrated in Fig. 14.

When performing permutations in each segment separately as shown in Fig. 14b, the client sends the permuted indices of the sparse updates of each segment separately to the databases. This leaks information about the indices of the sparse updates, since the databases learn how the  $r$  fraction of updates are distributed among the segments, i.e., the number of sparse updates in each segment. The information leakage on the indices of the sparse updates is defined by the mutual information between the real indices of the sparse updates and all the information sent by the client to any individual database. Note that no information is leaked on the values of updates since the same method of adding random noise in (19) is used to hide the values of updates. With  $B$  segments, the information leakage is characterized by the joint entropy of the  $B$  random variables representing the numbers of sparse updates in the  $B$  segments. Therefore, the information leakage and the storage cost of  $O(\frac{L^2}{B})$ , Fig. 14, are inversely proportional.

The information leakage can be reduced further, by carrying out another stage of permutations. In addition to the  $B$  within segment permutations shown in Fig. 14, another round of permutations can be performed among the segments. While this method also leaks the number of sparse updates in each segment, it does not reveal the mapping between the number of updates and the segment index. Therefore, the information leakage is reduced to the joint entropy of the set of  $B$  random variables representing the distinct combinations of the numbers of sparse updates in the  $B$  segments, irrespective of the order. Fig. 15 shows the amounts of information leaked in the two

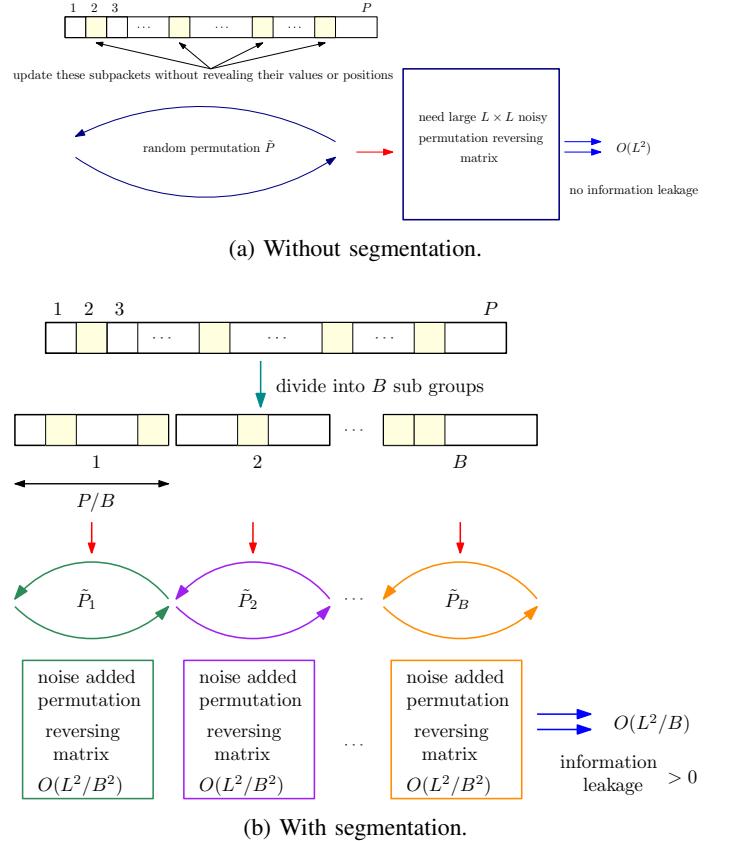


Fig. 14: Segmentation in permutation techniques.

cases (single-stage and two-stage permutations) for different number of segments in an example where the FL model consists of 12 subpackets. The read-write costs of both cases are not significantly affected by segmentation, and remain the same at  $\approx 2r'$  and  $\approx 2r$ .

**3) Rate-Distortion Trade Off in PRUW:** Although the communication cost of a private distributed learning process is significantly reduced by top  $r$  sparsification, the storage cost is high due to the noisy permutation reversing matrices, when using PRUW to guarantee privacy. By allowing a pre-determined amount of distortion in the learning process, PRUW can be used at a lower storage cost as well as a lower communication cost. This mechanism does not require any permutations, and reduces the communication cost by not downloading (uploading) a randomly selected set of parameters (updates) in the reading (writing) phase. This method can also be thought of as random sparsification. Note that the ignored parameters and updates in the reading and writing phases result in a certain amount of distortion. In other words, this method performs PRUW in distributed learning while achieving a lower communication cost, at the expense of a given amount of distortion in the reading and writing phases. The rate-distortion trade off in PRUW is quantified by analyzing the minimum communication cost at different amounts of allowed distortion. While the system model, privacy, security and correctness constraints are the same as before, this problem setting defines distortion budgets for the reading and writing phases. The goal is to achieve the minimum communication cost in the private read-write process while guaranteeing information-theoretic

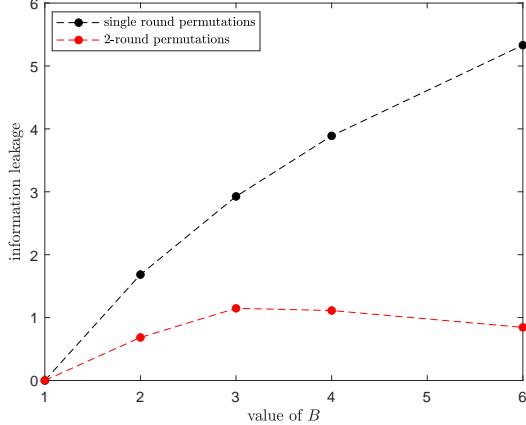


Fig. 15: Information leakage with varying number of segments.

privacy of client's local data and maintaining the distortion under the allowed budget. The distortion in reading (writing) phase is defined as the total number of parameters (updates) incorrectly downloaded (uploaded), normalized by the size of the model/submodel. It is shown in [26], [50] that the rate distortion trade off in PRUW is linear, i.e., the reading and writing costs are given by,

$$C_R = (1 - D_R)C_1, \quad C_W = (1 - D_W)C_2, \quad (22)$$

where  $C_R$ ,  $C_W$  represent the reading and writing costs,  $D_R$ ,  $D_W$  represent the distortion budgets in reading and writing phases, and  $C_1$ ,  $C_2$  are the reading and writing costs achieved by the basic PRUW without distortion.

Note that PRUW in FL with top  $r$  sparsification also introduces some amount of distortion. However, top  $r$  sparsification selects the most significant parameters and updates, which complicates the definition of *distortion*. In fact, in certain cases it has been shown that top  $r$  sparsification outperforms non-sparse FL. Therefore, both top  $r$  sparsification and random sparsification in private FL/FSL result in reduced communication costs, while the former suffers from increased storage cost and the latter suffers from decreased performance, i.e., accuracy of the model.

4) *Storage Constrained PRUW*: All of the previously mentioned PRUW schemes require databases, each with a storage capacity of at least the size of the training model. However, in practice, databases may not always have the space available to store an entire machine learning model. The given storage constraints of the databases can be either homogeneous or heterogeneous. The problem of private FSL with homogeneous storage constrained databases is studied in [51]. The system model consists of  $N$  non-colluding databases, each with a storage capacity of  $\mu M L$ , where  $M$  and  $L$  are the number of submodels and the size of a submodel and  $\mu$  is a fraction in the range  $\left[\frac{1}{N-3}, 1\right]$ . Private FSL with heterogeneous storage constraints is studied in [52], where the storage capacity of database  $n$  is given by  $\mu_n M L$  for each  $n$ , where each  $\mu_n \leq 1$  is allowed to be arbitrary. The privacy, security and correctness constraints in both homogeneous and heterogeneous system models, are the same as in Section V-D1.

There are two main methods to facilitate PRUW with storage constrained databases. 1) Storing only a fraction of each submodel in a given database, and replicating these fractions in only  $r$ ,  $r < N$  databases, i.e., *divided storage*. 2) Using MDS codes to combine multiple model parameters and store them as a single symbol in each database, i.e., *coded storage*. The schemes proposed in [51] and [52] for PRUW in FSL with storage constrained databases are based on the idea that combining both of the above methods, i.e., divided and coded storage, results in lower read-write costs, compared to only using either divided or coded storage. To this end, these schemes first find the optimum coding parameters and storage divisions that result in the minimum read-write costs for the given set of storage constraints, and use the basic PRUW scheme described in Section V-D1 to perform private FSL.

## VI. OPEN PROBLEMS AND FUTURE DIRECTIONS

- 1) Fundamental limits on performance metrics, i.e., converse results, have not yet been established for both PRUW and PSU based FSL. Investigating the fundamental limits is an interesting open problem, which might lead to capacity results of PRUW and PSU based FSL.
- 2) A practical and interesting future direction related to both PRUW and PSU based FSL is to develop private read-write schemes that are robust against adversaries or/and eavesdroppers.
- 3) All existing work on PRUW [26]–[28], [47], [48], [51], [52] consider a single client in the system model as shown in Fig. 9, i.e., all client-server communications are defined for a single client. A multi-client PRUW system model, where the private read-write process is defined from the perspective of a group of clients as opposed to defining it on an individual client with the same privacy, security and correctness requirements could be a promising future direction as it may result in a reduced communication cost, from combining queries/answers/updates of multiple clients.
- 4) In a setting where a given user has a wide variety of data to train multiple submodels in FSL at the same time, it would be interesting to see if there exists any method that decreases the total communication cost of privately reading/writing to multiple submodels simultaneously, compared to carrying out the PRUW process on each submodel individually.
- 5) The rate-privacy-storage trade off in [48] only allows information to be leaked on the indices of the sparse updates in FL with sparsification, which does not have a significant impact on the communication rate. An interesting direction is to investigate the rate-privacy trade off in PRUW by incorporating differential privacy as in [4], [14] to see the effect on the communication cost in private FSL/FL with sparsification when allowing a given amount of information to be leaked on the desired indices as well as the values of the selected updates/parameters.
- 6) The concept of PRUW can be used in other applications such as private epidemiological data collection. This

problem is introduced in [53] which contains three parties, namely, users, databases and the data collector. The problem setting contains both *reading* and *writing* components among different parties. The performance metric considered in [53] is the download cost at the data collector, i.e., the reading cost. An interesting future direction would be to consider the optimality of the writing cost, i.e., the upload cost of the users, which relates to the writing phase of PRUW.

## REFERENCES

- [1] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, November 1998.
- [2] H. Sun and S. A. Jafar. The capacity of private information retrieval. *IEEE Trans. on Info. Theory*, 63(7):4075–4088, July 2017.
- [3] N. Shah, K. Rashmi, and K. Ramchandran. One extra bit of download ensures perfectly private information retrieval. In *IEEE ISIT*, June 2014.
- [4] I. Samy, M. Attia, R. Tandon, and L. Lazos. Asymmetric leaky private information retrieval. *IEEE Trans. on Info. Theory*, 67(8):5352–5369, August 2021.
- [5] C. Tian, H. Sun, and J. Chen. Capacity-achieving private information retrieval codes with optimal message size and upload cost. *IEEE Trans. on Info. Theory*, 65(11):7613–7627, November 2019.
- [6] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, June 2000.
- [7] H. Sun and S. A. Jafar. The capacity of symmetric private information retrieval. *IEEE Trans. on Info. Theory*, 65(1):329–322, January 2019.
- [8] Z. Wang and S. Ulukus. Symmetric private information retrieval at the private information retrieval rate. *IEEE Jour. on Selected Areas in Info. Theory*, 3(2):350–361, June 2022.
- [9] Z. Wang and S. Ulukus. Communication cost of two-database symmetric private information retrieval: A conditional disclosure of multiple secrets perspective. In *IEEE ISIT*, 2022.
- [10] K. Banawan and S. Ulukus. The capacity of private information retrieval from coded databases. *IEEE Trans. on Info. Theory*, 64(3):1945–1956, March 2018.
- [11] H. Sun and S. A. Jafar. The capacity of robust private information retrieval with colluding databases. *IEEE Trans. on Info. Theory*, 64(4):2361–2370, April 2018.
- [12] K. Banawan and S. Ulukus. The capacity of private information retrieval from Byzantine and colluding databases. *IEEE Trans. on Info. Theory*, 65(2):1206–1219, February 2019.
- [13] K. Banawan and S. Ulukus. Multi-message private information retrieval: Capacity results and near-optimal schemes. *IEEE Trans. on Info. Theory*, 64(10):6842–6862, October 2018.
- [14] I. Samy, M. Attia, R. Tandon, and L. Lazos. Asymmetric leaky private information retrieval. *IEEE Trans. on Info. Theory*, 67(8):5352–5369, August 2021.
- [15] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, pages 1–19. Springer, 2004.
- [16] Z. Wang, K. Banawan, and S. Ulukus. Private set intersection: A multi-message symmetric private information retrieval perspective. *IEEE Trans. on Info. Theory*, 68(3):2001–2019, March 2022.
- [17] Z. Wang, K. Banawan, and S. Ulukus. Multi-party private set intersection: An information-theoretic approach. *IEEE Jour. on Selected Areas in Info. Theory*, 2(1):366–379, March 2021.
- [18] K. Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security*, pages 237–252, 2007.
- [19] Z. Wang and S. Ulukus. Private federated submodel learning via private set union. Available at arXiv:2301.07686.
- [20] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Trans. on Intelligent Systems and Technology*, 10(2):1–19, January 2019.
- [21] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *MobiCom*, April 2020.
- [22] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen. Secure federated submodel learning. Available online at arXiv:1911.02254.
- [23] M. Kim and J. Lee. Information-theoretic privacy in federated submodel learning. *ICT express*, February 2022.
- [24] K. Bonawitz, V. Ivanov, et al. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, October 2017.
- [25] Z. Wang and S. Ulukus. Digital blind box: Random symmetric private information retrieval. In *IEEE ITW*, pages 95–100, November 2022.
- [26] S. Vithana and S. Ulukus. Private read update write (PRUW) in federated submodel learning (FSL): Communication efficient schemes with and without sparsification. Available online at arXiv:2209.04421.
- [27] Z. Jia and S. A. Jafar.  $X$ -secure  $T$ -private federated submodel learning with elastic dropout resilience. *IEEE Trans. on Info. Theory*, 68(8):5418–5439, August 2022.
- [28] S. Vithana and S. Ulukus. Efficient private federated submodel learning. In *IEEE ICC*, May 2022.
- [29] Z. Jia and S. A. Jafar.  $X$ -secure  $T$ -private federated submodel learning. In *IEEE ICC*, June 2021.
- [30] J. Wangni, J. Wang, et al. Gradient sparsification for communication-efficient distributed optimization. In *NeurIPS*, December 2018.
- [31] S. Li, Q. Qi, et al. GGS: General gradient sparsification for federated learning in edge computing. In *IEEE ICC*, June 2020.
- [32] P. Han, S. Wang, and K. Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In *IEEE ICDCS*, November 2020.
- [33] S. Shi, K. Zhao, Q. Wang, Z. Tang, and X. Chu. A convergence analysis of distributed SGD with communication-efficient gradient sparsification. In *IJCAI*, August 2019.
- [34] Y. Sun, S. Zhou, Z. Niu, and D. Gunduz. Time-correlated sparsification for efficient over-the-air model aggregation in wireless federated learning. Available online at arXiv:2202.08420.
- [35] L. Barnes, H. Inan, B. Isik, and A. Ozgur. rTop- $k$ : A statistical estimation approach to distributed SGD. *IEEE JSAC*, 1(3):897–907, November 2020.
- [36] E. Ozfatura, K. Ozfatura, and D. Gunduz. Time-correlated sparsification for communication-efficient federated learning. In *IEEE ISIT*, July 2021.
- [37] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE SSP*, May 2017.
- [38] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE SSP*, May 2019.
- [39] J. Geiping, H. Bauermeister, H. Droege, and M. Moeller. Inverting gradients—how easy is it to break privacy in federated learning? In *NeurIPS*, December 2020.
- [40] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *NeurIPS*, December 2019.
- [41] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX*, April 2019.
- [42] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Infocom*, April-May 2019.
- [43] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE SSP*, May 2019.
- [44] Z. Jia, H. Sun, and S. A. Jafar. Cross subspace alignment and the asymptotic capacity of  $X$ -secure  $T$ -private information retrieval. *IEEE Trans. on Info. Theory*, 65(9):5783–5798, September 2019.
- [45] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, October 1949.
- [46] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.
- [47] S. Vithana and S. Ulukus. Model segmentation for storage efficient private federated learning with top sparsification. In *CIS*, March 2023.
- [48] S. Vithana and S. Ulukus. Rate-privacy-storage tradeoff in federated learning with top  $r$  sparsification. In *IEEE ICC*, May 2023.
- [49] S. Vithana and S. Ulukus. Private read-update-write with controllable information leakage for storage-efficient federated learning with top  $r$  sparsification. Available online at arXiv:2303.04123.
- [50] S. Vithana and S. Ulukus. Rate distortion tradeoff in private read update write in federated submodel learning. In *Asilomar Conference*, October 2022.
- [51] S. Vithana and S. Ulukus. Private read update write (PRUW) with storage constrained databases. In *IEEE ISIT*, June 2022.
- [52] S. Vithana and S. Ulukus. Private read update write (PRUW) with heterogeneous databases. In *IEEE ISIT*, June 2023.
- [53] J. Cheng, N. Liu, and W. Kang. On the asymptotic capacity of information theoretical privacy-preserving epidemiological data collection. Available online at arXiv:2212.05914.