Rate-Privacy-Storage Tradeoff in Federated Learning with Top r Sparsification

Sajani Vithana Sennur Ulukus
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
spallego@umd.edu *ulukus@umd.edu*

***Abstract*—We investigate the trade-off between rate, privacy and storage in federated learning (FL) with top r sparsification, where the users and the servers in the FL system only share the most significant r and r' fractions, respectively, of updates and parameters in the FL process, to reduce the communication cost. We present schemes that guarantee information theoretic privacy of the values and indices of the sparse updates sent by the users at the expense of a larger storage cost. To this end, we generalize the scheme to reduce the storage cost by allowing a certain amount of information leakage. Thus, we provide the general trade-off between the communication cost, storage cost, and information leakage in private FL with top r sparsification, along the lines of two proposed schemes.**

I. Introduction

In federated learning (FL) [1], [2], a set of users remotely train an ML model using their own local data in their own devices, and share only the gradient updates with the central server. This reduces the privacy leakage of users while decentralizing the processing power requirements of the central server. However, it has been shown that the gradients shared by a user leak information about the user's private data [3]–[8].

Apart from the privacy leakage, another drawback of FL is the significantly large communication cost incurred by sharing model parameters and updates with millions of users in multiple rounds. One solution to this problem is gradient sparsification [9]–[11], where the users only communicate a selected set of gradients and parameters as opposed to communicating all gradient updates and parameters. Top r sparsification [11] is a widely used sparsification technique, where only the most significant r fraction of parameters/updates are shared between the users and the central server, which significantly reduces the communication cost, since r is typically around 10^{-2} to 10^{-3}.

However, in sparsified FL, the values as well as the positions of the sparse updates leak information about the user's local data. Note that the positions of the sparse updates convey information about the most and least significant sets of parameters for a given user leaking information about its private data. Thus, in order to guarantee the privacy of users participating in the sparse FL process, two components need to be kept private, namely, 1) values of sparse updates, 2) positions of sparse updates. Reference [12] presents a scheme that achieves information theoretic privacy of the values and positions of the sparse updates in the context of federated submodel learning (FSL) [13]–[17]. The scheme in [12] incurs a significant storage cost. In this work, we extend the scheme in [12] to FL, with an additional variable that allows the storage cost to decrease at the expense of a certain amount of privacy leakage.

This work was supported by ARO Grant W911NF2010142.

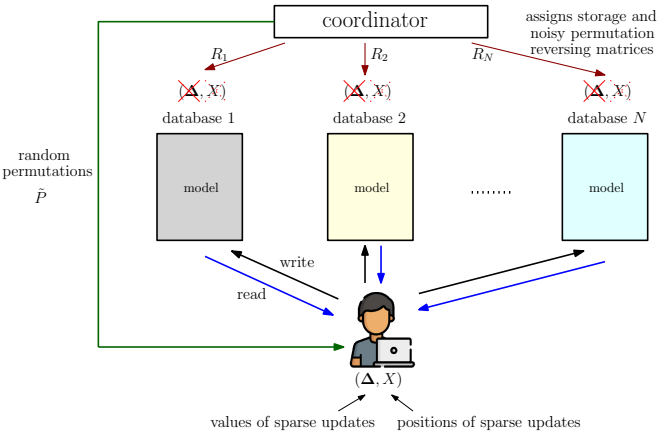

Fig. 1. System model.

In this paper, we consider an FL setting with multiple non-colluding databases storing the ML model, and a user that communicates with the databases as shown in Fig. 1. The schemes we propose are based on permutation techniques, where a coordinator initializes a random permutation of sets of parameters, and sends it to the users. The coordinator then places *noise added permutation reversing matrices* at each database in such a way that the databases learn nothing about the underlying permutation. All communications between the user and the databases take place in terms of the permuted indices, which guarantee the privacy of the positions of the sparse updates. Despite the added noise which ensures privacy, the parameters in each database get placed in the correct place.

The main challenge of this method is the significant storage cost incurred by the large permutation reversing matrices. We propose schemes that reduce the storage cost by reducing the size of the permutation reversing matrices, at the expense of a given amount of information leakage. This is achieved by dividing the ML model into multiple segments and carrying out permutations within each segment; see Figs. 2 and 3. The number of segments is chosen based on the allowed amount of information leakage and the storage capacity of the databases.

In this work, we propose two schemes to perform private FL with top r sparsification for uncoded storage. We present the

trade-off between the communication cost, storage complexity and information leakage in private FL with sparsification.

II. Problem Formulation

We consider an FL setting in which an ML model consisting of L parameters belonging to P subpackets is stored in N non-colluding databases. The parameters take values from a large enough finite field $\mathbb{F}_q$. A given user at a given time t trains the model using the user's local data. We consider sparsification in both uplink and downlink, to reduce the communication cost. In particular, the sparsification rates of the reading (downlink) and writing (uplink) phases are given by r' and r, respectively. In the reading (download) phase, the users only download a selected set of Pr' subpackets determined by the databases.[1] In the writing (upload) phase, each user only uploads the most significant Pr subpackets to the databases.[2]

Note that the users send no information to the databases in the reading phase. Therefore, no information about the user's local data is leaked to the databases in the reading phase. However, the users send the sparse updates and their positions (indices) to the databases in the writing phase to train the model. Information about the user's local data can be leaked to the databases from these updates and their indices. In this work, we consider the following privacy guarantees on the values and the positions of the sparse updates.

Privacy of the values of sparse updates: No information on the values of the sparse updates is allowed to leak to any of the databases, i.e.,

$$I(\Delta_i^{[t]}; G_n^{[t]}) = 0, \quad n \in \{1, \dots, N\}, \tag{1}$$

where $\Delta_i^{[t]}$ is the ith sparse (non-zero) update of a given user at time t and $G_n^{[t]}$ contains all the information sent by the user to database n at time t.

Privacy of the indices of sparse updates: The amount of information leaked on the positions of the sparse updates need to be maintained under a given privacy leakage budget, i.e.,

$$I(X^{[t]}; G_n^{[t]}) < \epsilon, \quad n \in \{1, \dots, N\}, \tag{2}$$

where $X^{[t]}$ is the set of indices of the sparse subpackets updated by a given user at time t. The system model with the privacy constraints is shown in Fig. 1. A coordinator is used to initialize the scheme. In addition to the privacy constraints, we require the following security and correctness conditions for the reliability of the scheme.

Security of the model: No information about the model parameters is allowed to leak to the databases, i.e.,

$$I(W^{[t]}; S_n^{[t]}) = 0, \quad n \in \{1, \dots, N\}, \tag{3}$$

where $W^{[t]}$ is the ML model and $S_n^{[t]}$ is the data content in database n at time t.

[1]These subpackets are determined based on the sparse updates received at time $t-1$, or by any other downlink sparsification protocol.

[2]We assume that all parameters in the sparse set of Pr subpackets in the writing phase have non-zero updates.

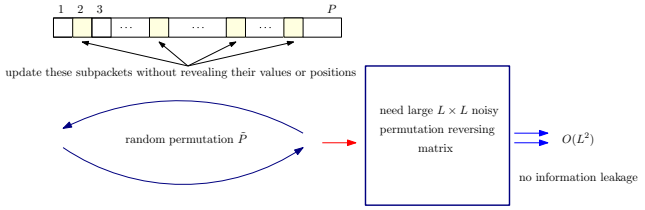

Fig. 2. Without segmentation, the process requires large noise added permutation reversing matrices, which increase the storage cost while guaranteeing zero privacy leakage.

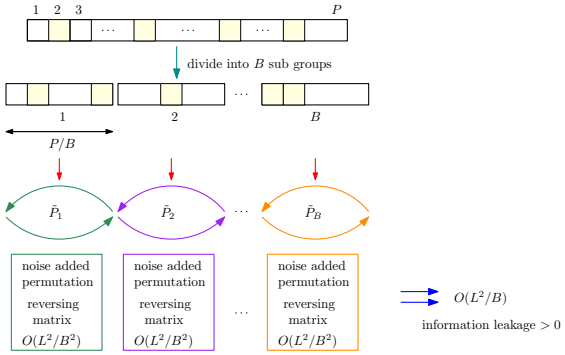

Fig. 3. With segmentation, the process requires small noisy permutation reversing matrices resulting in a lower storage cost at the expense of a positive information leakage.

Correctness in the reading phase: The user should be able to correctly decode the sparse set of subpackets of the model, denoted by J, determined by the downlink sparsification protocol, from the downloads in the reading phase, i.e.,

$$H(W_J^{[t-1]} | A_{1:N}^{[t]}) = 0, \tag{4}$$

where $W_J^{[t-1]}$ is the set of subpackets in set J of the model W at time $t-1$, i.e., before updating, and $A_n^{[t]}$ is the information downloaded from database n at time t.

Correctness in the writing phase: Let J' be the set of most significant Pr subpackets of the model, updated by a given user at time t. Then, the model must be correctly updated as,

$$W_s^{[t]} = \begin{cases} W_s^{[t-1]} + \Delta_s^{[t]}, & \text{if } s \in J' \\ W_s^{[t-1]}, & \text{if } s \notin J' \end{cases}, \tag{5}$$

where $W_s^{[t-1]}$ is the subpacket s of the ML model at time $t-1$ and $\Delta_s^{[t]}$ is the update of subpacket s at time t.

Reading and writing costs: The reading and writing costs are defined as $C_R = \frac{\mathcal{D}}{L}$ and $C_W = \frac{\mathcal{U}}{L}$, respectively, where $\mathcal{D}$ is the total number of symbols downloaded in the reading phase, $\mathcal{U}$ is the total number of symbols uploaded in the writing phase, and L is the size of the model. The total cost C_T is the sum of the reading and writing costs, $C_T = C_R + C_W$.

Storage cost: The storage cost is quantified by the order of the total number of symbols stored in each database.

In this work, we propose schemes to perform FL with top r sparsification, that result in the minimum total communication cost and storage complexity, while satisfying all privacy, security and correctness conditions described above.

TABLE I
ACHIEVABLE SETS OF COMMUNICATION COSTS, STORAGE COSTS, AND AMOUNTS OF INFORMATION LEAKAGE.

case	reading cost	writing cost	storage cost	information leakage
Case 1	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{2}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{2}{N}}$	$O(\frac{L^2}{B})$	$H(\hat{X}_1,\ldots,\hat{X}_B)$
Case 2	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{4}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{4}{N}}$	$\max\{O(\frac{L^2}{B}), O(\frac{L^2B^2}{N^2})\}$	$H(\tilde{X}_1,\ldots,\tilde{X}_B)$

III. MAIN RESULT

Theorem 1 *Consider an FL model stored in N non-colluding databases, consisting of L symbols from a finite field $\mathbb{F}_q$, which are included in P subpacekts. The model is divided into B segments of equal size ($1 \leq B < P$), such that each consecutive $\frac{P}{B}$ subpackets are included in each segment. Assume that the FL model is updated by users at each time instance with uplink and downlink sparsification rates (top r sparsification) of r and r', respectively. Let $\hat{X}_i$ be the random variable representing the number of subpackets with non-zero (sparse) updates of the ith segment generated by any given user, and let $(\tilde{X}_1,\ldots,\tilde{X}_B)$ be the random vector representing all distinct combinations of $(\hat{X}_1,\ldots,\hat{X}_B)$, irrespective of the segment index. Then, the reading/writing costs, storage complexities and amounts of information leaked on the indices of sparse updates presented in Table I are achievable.*

Remark 1 *When $B = 1$ (no segmentation), $\hat{X}_1 = \tilde{X}_1 = Pr$ and the corresponding infromation leakage is zero since Pr is fixed and $H(\hat{X}_1) = H(\tilde{X}_1) = 0$. That is, the schemes corresponding to the two cases achieve information theoretic privacy of the values and positions of the sparse updates while incurring the same communication costs stated in Table I.*

Remark 2 *For a given privacy budget on the positions of the sparse updates given by ϵ, the optimum number of segments B can be calculated by minimizing the storage complexity, such that $H(\hat{X}_1,\ldots,\hat{X}_B) < \epsilon$ or $H(\tilde{X}_1,\ldots,\tilde{X}_B) < \epsilon$ is satisfied.*

Remark 3 *Since $H(\hat{X}_1,\ldots,\hat{X}_B)$ considers all possible values of $\hat{X}_i$, while $H(\tilde{X}_1,\ldots,\tilde{X}_B)$ only considers distinct sets of $\{\hat{X}_i\}_{i=1}^B$, $H(\hat{X}_1,\ldots,\hat{X}_B) > H(\tilde{X}_1,\ldots,\tilde{X}_B)$.*

Remark 4 *Consider an example setting with $P = 12$ subpackets divided into $B = 1, 2, 3, 4, 6$ segments. Assume that each subpacket is equally probable to be selected to the set of most significant $Pr = 3$ subpackets. The behavior of the information leakage for each value of B is shown in Fig. 4.*

Remark 5 *For the first case, one can achieve a lower storage cost at the expense of a higher information leakage by increasing B and vice versa. This is because the number of different realizations of the placements of the Pr sparse subpackets at the B segments increases with B when all permutations of the placements are considered. However, when only the distinct placements are considered in case 2 (without permutations), after $B = Pr$, the probability of each realization increases, which in turn decreases the entropy $H(\tilde{X}_1,\ldots,\tilde{X}_B)$, i.e., the information leakage. Therefore, the storage-privacy leakage trade-off in case 2 also follows an inverse relation. The variation of the privacy leakage and the storage cost is controlled by the parameter B. The communication cost however is independent of B, which makes it independent of the storage cost and the privacy leakage.*

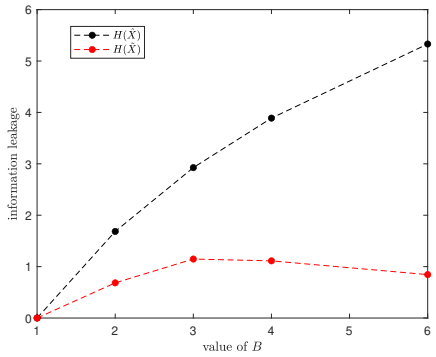

Fig. 4. Information leakage of an example setting with $P = 12$ versus B.

IV. PROPOSED SCHEME

We propose two schemes. Both schemes achieve privacy of the indices of the sparse subpackets by utilizing a permutation technique. In case 1, the model is divided into B segments, and the scheme only considers permutations among the subpackets within each segment while revealing the real segment indices of the sparse subpackets. The scheme in case 2 considers permutations within and among segments to reduce the information leakage further. The schemes are presented in terms of examples due to space limitations here.

A. Case 1: Single Stage Permutations

Consider an example setting with $P = 15$ and $B = 3$.

1) Initialization: The storage of a single subpacket s in case 1 is given by,

$$S_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[s]} + \sum_{j=0}^{x} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[s]} + \sum_{j=0}^{x} \alpha_n^j I_{\ell,j} \end{bmatrix}, \tag{6}$$

where $x = \ell$,[3] $W_i^{[s]}$ is the ith symbol of subpacket s, $I_{i,j}$ are random noise symbols and $\{f_i\}_{i=1}^{\ell}, \{\alpha_n\}_{n=1}^{N}$ are globally known distinct constants from $\mathbb{F}_q$. At the initialization stage the coordinator sends $B = 3$ randomly and independently

[3] ℓ is the subpacketization, for which an expression is derived at the end of Section IV-A2.

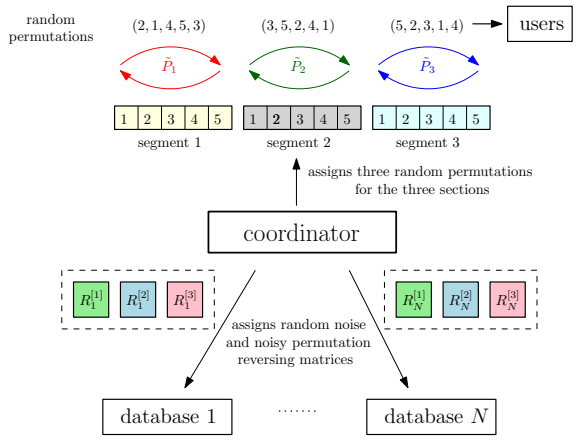

Fig. 5. Initialization of the scheme for case 1.

chosen permutations of the $\frac{P}{B} = 5$ subpackets in each of the $B = 3$ segments to all users, denoted by $\tilde{P}_1, \tilde{P}_2, \tilde{P}_3$ as shown in Fig. 5. The coordinator also sends the corresponding noise added permutation reversing matrices given by,

$$R_n^{[i]} = \tilde{R}_n^{[i]} + \bar{Z}_i, \quad i = 1, \ldots, B, \tag{7}$$

to database n, $n \in \{1, \ldots, N\}$, as shown in Fig. 5, where $\tilde{R}_n^{[i]}$ is the scaled permutation reversing matrix corresponding to the permutation $\tilde{P}_i$ and $\tilde{Z}_i$ is a random noise matrix of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$. Based on this example, the permutation reversing matrix for database n, $n \in \{1, \ldots, N\}$ corresponding to the first segment with permutation: $\tilde{P}_1 = (2, 1, 4, 5, 3)$ is given by,

$$R_n^{[1]} = \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \bar{Z}_1, \tag{8}$$

where $\Gamma_n = \text{diag}\{\frac{1}{f_1-\alpha_n}, \ldots, \frac{1}{f_\ell-\alpha_n}\}$ and $0_{\ell\times\ell}$ is the all zeros matrix of size $\ell \times \ell$.

2) Reading Phase: The databases decide the permuted indices of the Pr' sparse subpackets to be sent to the users at time t in the reading phase, by selecting the permuted indices of the most commonly updated Pr' subpackets by all users at time $t-1$. Note that the databases are unaware of the real indices of the sparse subpackets updated by the users in the writing phase at each time instance and only work with the permuted indices in both phases. For example, let the sparse set of permuted subpacket indices chosen by the databases to be sent to the users corresponding to the first segment be $\tilde{V}_1 = \{1, 3\}$. One designated database sends these permuted indices of each segment to the users. The users then find the real indices, using the known permutations, i.e., for segment 1, the real set of indices is given by $V_1 = \{2, 4\}$.

In order to send the ith sparse subpacket of segment j (permuted) denoted by $\tilde{V}_j(i)$, each database n, $n \in \{1, \ldots, N\}$ generates the following queries.

$$Q_n^{[\tilde{V}_j(i)]} = \sum_{k=1}^{\ell} R_n^{[j]}(:, (i-1)\ell + k). \tag{9}$$

For example, the query corresponding to the first sparse subpacket of the first segment is given by,

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, k) = \begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1, \tag{10}$$

where Z_1 is a random noise vector resulted by the noise component of $R_n^{[1]}$. Then, database $n \in \{1, \ldots, N\}$ sends the corresponding answer by calculating the dot product between the query and the scaled storage of the respective segment as,

$$A_n^{[\tilde{V}_1(1)]} = (D_n \times S_n)^T Q_n^{[\tilde{V}_1(1)]} \tag{11}$$

$$= \frac{1}{f_1-\alpha_n} W_1^{[2]} + \ldots + \frac{1}{f_\ell-\alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell+1), \tag{12}$$

where D_n is the diagonal matrix of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$ given by $D_n = \text{diag}\{\Gamma_n^{-1}, \ldots, \Gamma_n^{-1}\}$ and $P_{\alpha_n}(\ell+1)$ is a polynomial in α_n of degree $\ell+1$. Then, the users obtain the parameters of real subpacket 2, by solving the N equations, i.e., answers from N databases, of the form (12), given that $N = 2\ell + 2$, which determines the subpacketization as $\ell = \frac{N-2}{2}$. The same procedure is carried out for all sparse subpackets in all segments. The resulting reading cost is given by,

$$C_R = \frac{Pr'(\log_q \frac{P}{B} + \log_q B) + Pr'N}{L} = \frac{2r'(1 + \frac{\log_q P}{N})}{1 - \frac{2}{N}}. \tag{13}$$

3) Writing Phase: In the writing phase, each user generates non-zero updates for Pr most significant subpackets, and sends the noise added combined updates, i.e., single symbol per subpacket, along with their permuted subpacket indices and the segment indices to each of the databases. The combined update of some (real) subpacket i of segment j, sent to database n, $n \in \{1, \ldots, N\}$ is given by,

$$U_n^{[i,j]} = \sum_{k=1}^{\ell} \prod_{r=1, r\neq k}^{\ell} (f_r - \alpha_n)\tilde{\Delta}_k^{[i,j]} + \prod_{r=1}^{\ell} (f_r - \alpha_n) Z^{[i,j]}, \tag{14}$$

where $\tilde{\Delta}_k^{[i,j]} = \frac{\Delta_k^{[i,j]}}{\prod_{r=1, r\neq k}^{\ell}(f_r - f_k)}$ with $\Delta_k^{[i,j]}$ being the update of the kth symbol of subpacket i of segment j and $Z^{[i,j]}$ is a random noise symbol. Note that the addition of $Z^{[i,j]}$ to the updates in (14) guarantees information theoretic privacy of the values of updates from Shannon's one time pad theorem. For example, assume that a given user wants to update the real subpackets 2 and 4 from segment 1, subpacket 2 from segment 2, and subpacket 5 from segment 3. Based on the permutations considered in this example, i.e., $\tilde{P}_1 = \{2, 1, 4, 5, 3\}$, $\tilde{P}_2 = \{3, 5, 2, 4, 1\}$ and $\tilde{P}_3 = \{5, 2, 3, 1, 4\}$, the user generates the permuted (update, subpacket, segment) tuples given by $(U_n^{[2,1]}, 1, 1)$, $(U_n^{[4,1]}, 3, 1)$ for segment 1, $(U_n^{[2,2]}, 3, 2)$ for segment 2, and $(U_n^{[5,3]}, 1, 3)$ for segment 3. Note that there is

no permutation in the segment index, and only the subpacket indices within each segment are being permuted. Database n, $n \in \{1,\dots,N\}$ creates permuted update vectors for each segment upon receiving the Pr permuted (update, subpacket, segment) tuples. For segment 1, the permuted update vector is given by,

$$\hat{U}_n^{[1]} = [U_n^{[2,1]} \cdot 1_\ell^T,\ 0 \cdot 1_\ell^T,\ U_n^{[4,1]} \cdot 1_\ell^T,\ 0 \cdot 1_\ell^T,\ 0 \cdot 1_\ell^T]^T \quad (15)$$

where 1_ℓ is the all ones vector of size $\ell \times 1$. Next, the databases privately rearrange the updates in the real order and calculate the incremental updates of each segment. The incremental update calculation of segment 1 in database n is given by,

$$\bar{U}_n^{[1]} = R_n^{[1]} \hat{U}_n^{[1]} \quad (16)$$

$$= \left(\begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \bar{Z}_1 \right) \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[4,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \quad (17)$$

$$= \begin{bmatrix} 0_\ell \\ \frac{U_n^{[2,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{U_n^{[2,1]}}{f_\ell - \alpha_n} \\ 0_\ell \\ \frac{U_n^{[4,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{U_n^{[4,1]}}{f_\ell - \alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0_\ell \\ \frac{\Delta_1^{[2,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[2,1]}}{f_\ell - \alpha_n} \\ 0_\ell \\ \frac{\Delta_1^{[4,1]}}{f_1 - \alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[4,1]}}{f_\ell - \alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell), \quad (18)$$

where $P_{\alpha_n}(\ell)$ here is a vector of size $\frac{L}{B}$ consisting of polynomial in α_n of degree ℓ, and the last equality is obtained by applying [13, Lemma 1]. The same process is carried out for the other two segments as well. Since the incremental update is in the same form as the storage in (6), the storage of segment j, $j \in \{1,2,3\}$ at time t can be updated as,

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \quad n \in \{1,\dots,N\}. \quad (19)$$

Note from (18) that for segment 1, the two real sparse subpackets 2 and 4 have been correctly updated, while ensuring that the rest of the subpackets remain the same, without revealing the real subpacket indices 2 and 4 to any of the databases. The resulting writing cost is given by,

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} = \frac{2r(1+\log_q P)}{1 - \frac{2}{N}}. \quad (20)$$

The total storage cost is given by, $L + \frac{L^2}{B^2} \times B \approx O(\frac{L^2}{B})$.

B. Case 2: Two-Stage Permutations

In case 1, only the subpacket indices within each segment were permuted, and the real segment indices were uploaded to the databases by the users. In this case, we permute subpacket indices within segments as well as the segment indices to reduce the information leakage. However, this increases the storage cost since the permutation of segment indices requires an additional noise added permutation reversing matrix to be stored in the databases. For case 2, consider an example setting with $P = 12$ subpackets (with subpacketization ℓ) which are divided into and $B = 3$ equal segments.

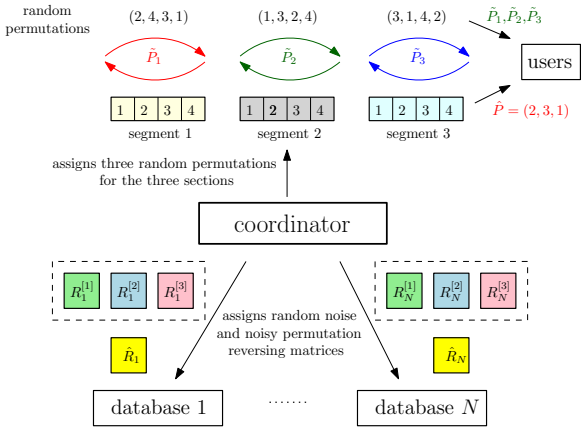

Fig. 6. Initialization of the scheme for case 2.

1) Initialization: The storage of a single subpacket in this case is the same as (6) with $x = \ell + 1$. The coordinator places the $B = 3$ permutations and the corresponding noisy permutation reversing matrices similar to case 1. In addition, the coordinator randomly and independently selects a permutation of the $B = 3$ segments $\hat{P}$ and sends it to the users, while placing the corresponding noise-added permutation reversing matrix $\hat{R}_n$ at database n, $n \in \{1,\dots,N\}$. Consider the example setting given in Fig. 6. The noise added permutation reversing matrix corresponding to the first segment with permutation $\tilde{P}_1 = \{2,4,3,1\}$ is given by,

$$R_n^{[1]} = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \bar{Z}_1, \quad (21)$$

with the same notation as in case 1. The noise added permutation reversing matrix at database n corresponding to the segmentwise permutation $\hat{P} = \{2,3,1\}$ is given by,

$$\hat{R}_n = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi \\ \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} \end{bmatrix} \hat{Z} \quad (22)$$

$$= \begin{bmatrix} b_{1,1}^{[n]} & b_{1,2}^{[n]} & b_{1,3}^{[n]} \\ b_{2,1}^{[n]} & b_{2,2}^{[n]} & b_{2,3}^{[n]} \\ b_{3,1}^{[n]} & b_{3,2}^{[n]} & b_{3,3}^{[n]} \end{bmatrix}, \quad (23)$$

where $\Phi = \text{diag}(1_\ell)$ and $\hat{Z}$ is a random noise matrix of size $B\ell \times B\ell$. Each matrix $\hat{R}_n$ is represented in blocks of size $\ell \times \ell$, as shown in (23).

2) Reading Phase: In the reading phase, the databases determine the permuted indices of the Pr' most significant subpackets to be sent to the users as described in case 1.

Assume that the permuted (subpacket, segment) pair of a chosen subpacket is $(\eta_p, \phi_p) = (1, 3)$. A designated database sends this information to the user and the user finds the corresponding real segment ϕ_r and subpacket η_r as $\phi_r = \hat{P}(\phi_p) = 1$ and $\eta_r = \tilde{P}_{\phi_r}(\eta_p) = 2$. To send the corresponding real subpacket, the databases first generate the combined noisy permutation reversing matrix given by,

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & R_n^{[2]} & \\ & & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix} & \begin{bmatrix} b_{1,2}^{[n]} & & \\ & \ddots & \\ & & b_{1,2}^{[n]} \end{bmatrix} & \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix} \\ \begin{bmatrix} b_{2,1}^{[n]} & & \\ & \ddots & \\ & & b_{2,1}^{[n]} \end{bmatrix} & \begin{bmatrix} b_{2,2}^{[n]} & & \\ & \ddots & \\ & & b_{2,2}^{[n]} \end{bmatrix} & \begin{bmatrix} b_{2,3}^{[n]} & & \\ & \ddots & \\ & & b_{2,3}^{[n]} \end{bmatrix} \\ \begin{bmatrix} b_{3,1}^{[n]} & & \\ & \ddots & \\ & & b_{3,1}^{[n]} \end{bmatrix} & \begin{bmatrix} b_{3,2}^{[n]} & & \\ & \ddots & \\ & & b_{3,2}^{[n]} \end{bmatrix} & \begin{bmatrix} b_{3,3}^{[n]} & & \\ & \ddots & \\ & & b_{3,3}^{[n]} \end{bmatrix} \end{bmatrix} \tag{24}$$

In order to send the permuted subpacket indicated by $(\eta_p, \phi_p) = (i, j)$, each database generates the query given by,

$$Q_n^{[i,j]} = \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}_{L \times L} \times \sum_{k=1}^{\ell} R_n(:, (j-1)\frac{P\ell}{B} + (i-1)\ell + k). \tag{25}$$

Then, the answer is generated by the dot product between the query and the storage as explained in case 1. In order for the user to be able to download the required subpacket using the N answers, the system should satisfy $N = 2\ell + 4$, fixing the subpacketization of case 2 at $\ell = \frac{N-4}{2}$, which results in the reading cost given in Table I for case 2, using a similar calculation as in (13).

3) Writing Phase: In the writing phase, the user sends the combined updates, permuted subpacket indices and permuted segment indices of the Pr most significant subpackets to all databases. Assume that a given user wants to update the Pr sparse subpackets identified by the real (subpacket, segment) pairs given by, $(\eta_r, \phi_r) = \{(2, 1), (1, 2), (3, 3)\}$. Based on within segment permutations given by $\tilde{P}_1 = (2, 4, 3, 1)$, $\tilde{P}_2 = (1, 3, 2, 4)$, $\tilde{P}_3 = (3, 1, 4, 2)$, and the segmentwise permutation given by $\hat{P} = (2, 3, 1)$, the user sends the following (permuted) information to database n, $n \in \{1, \ldots, N\}$,

$$(U_n^{[\eta_r, \phi_r]}, \eta_p, \phi_p) = \{(U_n^{[2,1]}, 1, 3), (U_n^{[1,2]}, 1, 1), (U_n^{[3,3]}, 1, 2)\} \tag{26}$$

where the combined updates $U_n^{[\eta_r, \phi_r]}$ are of the form (14). Once the databases receive all permuted (update, subpacket, segment) tuples, they construct the permuted update vector as,

$$\tilde{U}_n = [U_n^{[1,2]}, 0, 0, 0, U_n^{[3,3]}, 0, 0, 0, U_n^{[2,1]}, 0, 0, 0]^T \tag{27}$$

This vector is then scaled by an all ones vector of size $\ell \times 1$ to aid the rest of the calculations. The scaled permuted update vector is given by $\hat{U}_n = [\tilde{U}_n(1) \cdot 1_\ell^T, \ldots, \tilde{U}_n(12) \cdot 1_\ell^T]^T$. Then, database n, $n \in \{1, \ldots, N\}$ calculates the incremental update using the combined noisy permutation reversing matrix in (24) as $\bar{U}_n = R_n \times \hat{U}_n$, which is of the same form as the storage in (6) with $x = \ell + 1$. Therefore, the storage at time t, $S_n(t)$ can be updated as $S_n(t) = S_n(t-1) + \bar{U}_n$.

The numbers of symbols stored as data, noise added intra and inter segment permutation reversing matrices are given by, L, $\frac{L^2}{B}$ and $\ell^2 B^2 = \frac{L^2 B^2}{N^2}$, respectively. Therefore, the storage complexity is $\max\{O(\frac{L^2}{B}), O(\frac{L^2 B^2}{N^2})\}$.

In the two proposed schemes, there exists a positive information leakage on the indices of sparse updates when $B > 1$, since the numbers of subpackets with non-zero updates in each segment is revealed to the databases in permuted or non-permuted order, i.e., the databases learn the distribution of the Pr sparse subpackets among the B segments. This is the intuition behind the information leakage characterized in Table I for the two cases. The proofs are omitted in this paper due to space limitations.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication efficient learning of deep networks from decentralized data. *AISTATS*, April 2017.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Trans. on Intel. Systems and Technology*, 10(2):1–19, January 2019.

[3] 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.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE SSP*, May 2017.

[5] 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.

[6] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE SSP*, May 2019.

[7] J. Geiping, H. Bauermeister, H. Droge, and M. Moeller. Inverting gradients–how easy is it to break privacy in federated learning? Available online at arXiv:2003.14053.

[8] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *NeurIPS*, December 2019.

[9] J. Wangni, J. Wang, et al. Gradient sparsification for communication-efficient distributed optimization. In *NeurIPS*, December 2018.

[10] P Han, S. Wang, and K. Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In *IEEE ICDCS*, November 2020.

[11] L. Barnes, H. Inan, B. Isik, and A. Ozgur. rTop-k: A statistical estimation approach to distributed SGD. *IEEE JSAIT*, 1(3):897–907, November 2020.

[12] S. Vithana and S. Ulukus. Private federated submodel learning with sparsification. In *IEEE ITW*, November 2022.

[13] 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.

[14] S. Vithana and S. Ulukus. Efficient private federated submodel learning. In *IEEE ICC*, May 2022.

[15] S. Vithana and S. Ulukus. Private read update write (PRUW) with storage constrained databases. In *IEEE ISIT*, June 2022.

[16] S. Vithana and S. Ulukus. Rate distortion tradeoff in private read update write in federated submodel learning. In *Asilomar Conference*, October 2022.

[17] 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.