Private Read-Update-Write With Controllable Information Leakage for Storage-Efficient Federated Learning With Top r Sparsification

Sajani Vithana, *Member, IEEE*, and Sennur Ulukus, *Fellow, IEEE*

***Abstract*— In federated learning (FL), a machine learning (ML) model is collectively trained by a large number of users, using their private data in their local devices. With top r sparsification in FL, the users only upload the most significant r fraction of updates, and download only the most significant r' fraction of parameters in order to reduce the communication cost. However, the values and the indices of the sparse updates and parameters leak information about the users' private data. In this work, we consider an FL setting where N non-colluding databases store the model to be trained, from which the users download and update sparse parameters privately, without revealing the values of the updates/parameters or their indices to the databases. We propose four schemes with different properties that are based on cross subspace alignment (CSA) and permutation techniques, to perform this task while achieving the minimum communication costs within the scope of CSA, and show that the information theoretic privacy of both the values and the positions of the sparse updates/parameters can be guaranteed. This is achieved at a considerable storage cost, though. To alleviate this, we generalize the schemes in such a way that the storage cost is reduced at the expense of a certain amount of information leakage, using a model segmentation mechanism. In general, we provide the trade-off between the communication cost, storage cost and information leakage in private FL with top r sparsification.**

***Index Terms*— Private read update write (PRUW), federated learning (FL), sparsification, information leakage versus storage efficiency.**

I. Introduction

FEDERATED learning (FL) [1], [2] is a widely used distributed learning technique where a set of users remotely train a 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 data providers while decentralizing the processing power requirements of the central server. However, it has been shown that the gradients shared by a user can be used to obtain information about the user's private data [3], [4], [5], [6], [7], [8], [9]. Cryptographic protocols as in secure aggregation [10], differential privacy (DP) [11] via noise addition, data sampling and data shuffling, e.g., [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], and [24] are some of the methods used to minimize this information leakage in FL. However, these methods do not guarantee information theoretic privacy of each individual user's local data.

Apart from the privacy leakage, another drawback of FL is the large communication cost incurred by sharing model parameters and updates with millions of users in multiple rounds. Some of the solutions to this problem include, gradient quantization [25], [26], [27], [28], federated submodel learning (FSL) [29], [30], [31], [32], [33], [34], [35], [36], [37], and gradient sparsification [38], [39], [40], [41], [42], [43], [44], [45]. In gradient quantization, the values of the gradients are quantized and represented with a fewer number of bits. In FSL, the ML model is divided into multiple submodels based on different types of data used to train the entire model, and each user only downloads and updates the submodel that can be updated by its own local data. In gradient sparsification, the users only communicate a selected set of gradients and parameters as opposed to communicating all gradient updates and parameters. Typically the sparsification rates (fraction of the parameters/updates communicated) are around 10^{-2} to 10^{-3}, which significantly reduces the communication cost.

Top r sparsification 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. In certain cases, it has been shown that top r sparsification outperforms classical FL. However, the values as well as the positions (indices) of the sparse updates leak information about the user's local data. Note that the positions of the sparse updates leak information about the most and least significant sets of parameters for a given user, which can be used to infer information about the user's 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) indices of sparse updates. In this work, we develop schemes to perform the user-database communications in FL with top r sparsification while guaranteeing information-theoretic privacy of the values and indices of the sparse updates.

We consider an FL setting with multiple non-colluding databases storing the ML model, and a single user

Manuscript received 5 March 2023; revised 8 October 2023; accepted 13 November 2023. Date of publication 12 December 2023; date of current version 23 April 2024. This work was supported by the Army Research Office (ARO) under Grant W911NF2010142. An earlier version of this paper was presented in part at the 2023 57th Annual Conference on Information Sciences and Systems and in part by the 2023 IEEE International Conference on Communications. *(Corresponding author: Sennur Ulukus.)*

The authors are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: spallego@umd.edu; ulukus@umd.edu).

Communicated by A. Sarwate, Associate Editor for Security and Privacy.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2023.3339450.

Digital Object Identifier 10.1109/TIT.2023.3339450

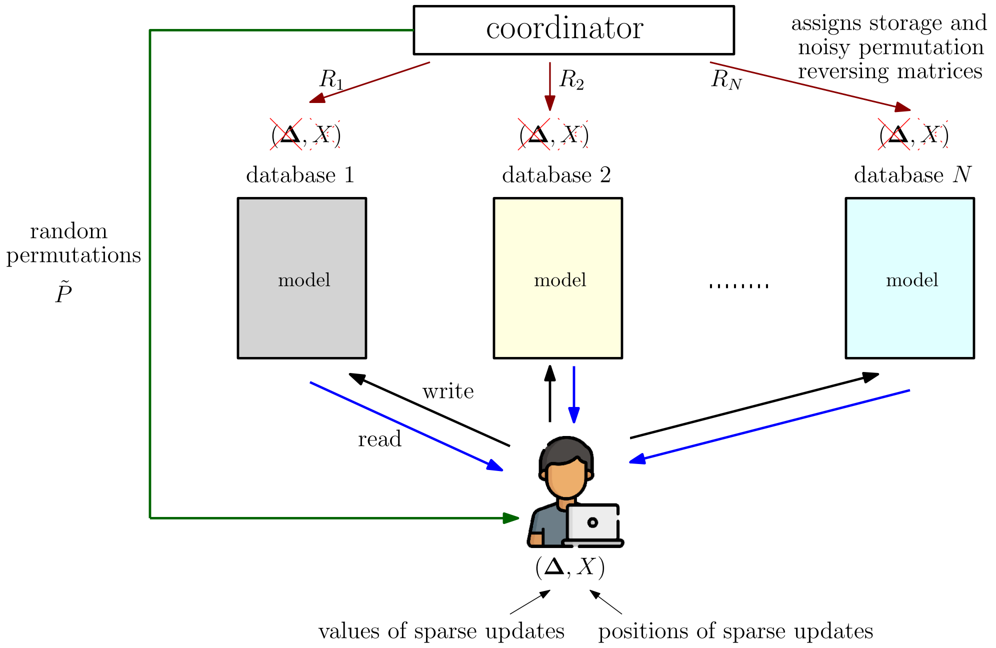

Fig. 1. System model: A user reads (downloads), updates, writes (uploads) a ML model.

communicating with each of the databases as shown in Fig. 1. The schemes we propose in this work 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 guarantees the privacy of the indices of the sparse updates.[1] However, the parameters in each database get updated in the correct order, with the aid of the noise added permutation reversing matrices. The main drawback of this method is the considerably large storage cost incurred by the large permutation reversing matrices. To that end, we propose schemes that reduce the storage cost by decreasing the size of the noise added 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. This is illustrated in Fig. 2. 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 four schemes to perform user-database communications in private FL with top r sparsification with different properties such as lower communication costs, lower storage costs or lower amounts of information leakage. The four schemes differ from each other based on the storage structure (MDS coded or uncoded) and the permutation mechanism (only within-segment permutations or within and inter-segment permutations) used. MDS coded storage decreases the storage cost while increasing the communication cost, and the two-stage permutations (within and inter-segment permutations) decrease the information leakage significantly compared to single-stage permutations (only within-segment permutations), while slightly increasing the communication cost. Based on the specifications and limitations of the given FL task, one can choose the most suitable scheme.

As the main technical contribution of this paper, we introduce a privacy technique that facilitates computation on a certain set of data corresponding to a selected set of indices in a larger dataset, without revealing any information on the selected indices or the corresponding values to the computing/storage nodes. This is useful in the sparse FL application considered in this paper to guarantee the privacy of the indices and the values of the sparse parameters and updates, downloaded and uploaded by the user, respectively. We divide the main technical contribution into five components, namely, 1) a zero-error private permutation reversal (PPR) mechanism that takes a selected set of randomly permuted indices as the input, and performs computations on the corresponding real indices (permutation-reversed) without knowing/learning the real indices or the underlying permutation, 2) extensions of the proposed PPR mechanism to uncoded and MDS coded storage, 3) the concept of data segmentation to reduce the storage complexity of the proposed PPR mechanism at the expense of a controllable amount of information leakage, 4) the concept of two-stage permutations to reduce the privacy leakage, along with the corresponding (two-stage) PPR mechanism, and 5) explicit derivations of the amounts of information leaked in each proposed mechanism. The above five components collectively define the different schemes proposed in this paper to perform the user-database communications in private FL with top r sparsification. This further characterizes an achievable trade-off between the communication cost, storage complexity and information leakage in private FL with top r sparsification.

[1] Rigorous proofs on privacy/information leakage are provided in Section IV-B.

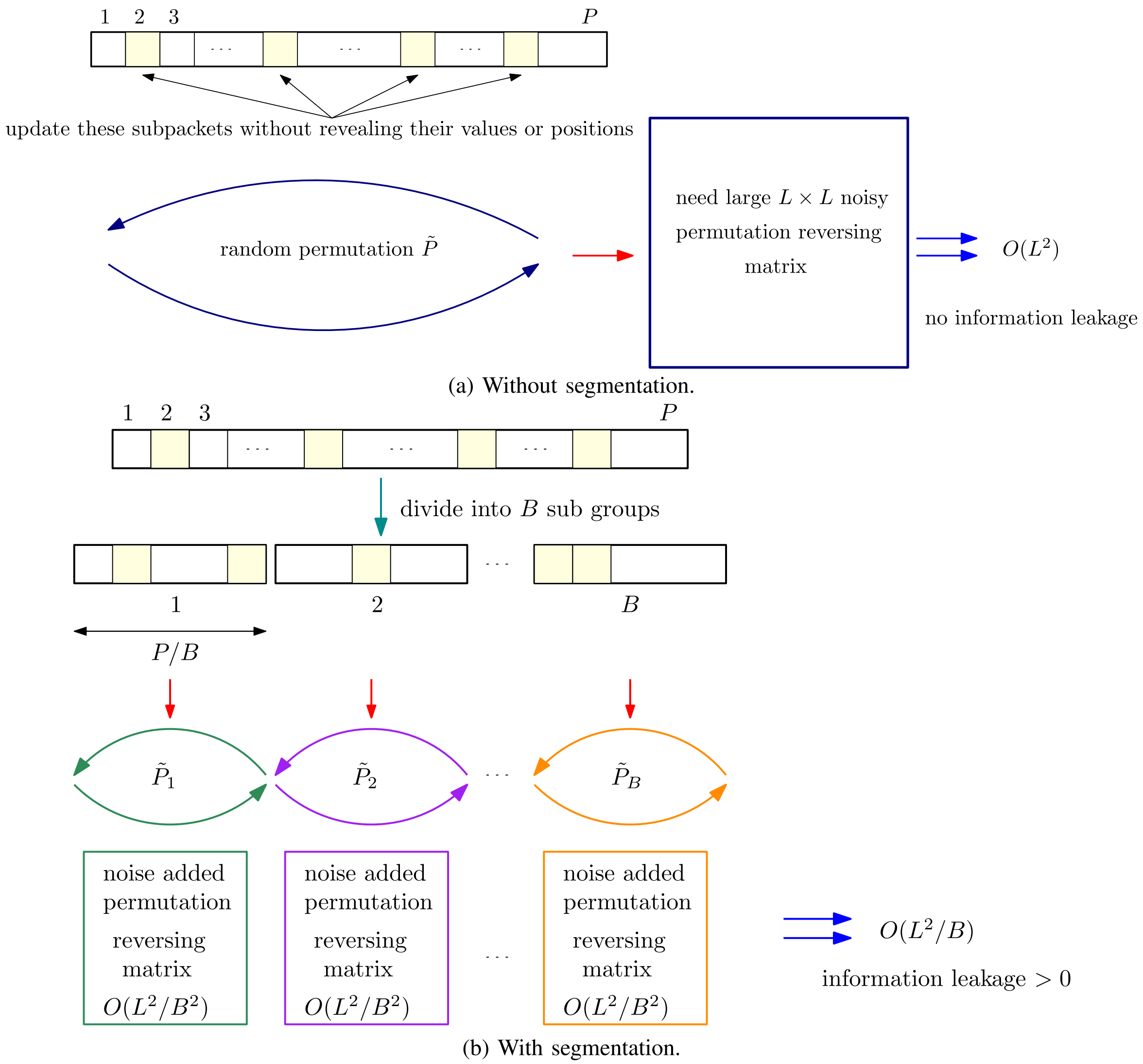
1
2
3
P
update these subpackets without revealing their values or positions
random permutation $\tilde{P}$
need large $L \times L$ noisy permutation reversing matrix
$O(L^2)$
no information leakage
(a) Without segmentation.
1
2
3
P
divide into B sub groups
1
2
B
P/B
$\tilde{P}_1$
$\tilde{P}_2$
$\tilde{P}_B$
noise added permutation reversing matrix $O(L^2/B^2)$
noise added permutation reversing matrix $O(L^2/B^2)$
noise added permutation reversing matrix $O(L^2/B^2)$
$O(L^2/B)$
information leakage > 0
(b) With segmentation.

Fig. 2. Motivation for segmentation in permutation techniques: (a) Permutation of the entire model without segmentation. (b) Permutation within segments with segmentation.

II. Problem Formulation

We consider a FL setting in which a 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 reads (downloads) the required parameters of the model from the databases, trains the model using the user's local data, and writes (uploads) the most significant r fraction of updates back to all databases. In this work, 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 other words, in the reading (download) phase, the users only download a selected set of Pr' subpackets determined by the databases.[2] Once the model is trained locally, each user only uploads the most significant Pr set of updates (corresponding subpackets) to the databases in the writing (upload) phase.[3]

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. 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.[4] In this work, we consider the following privacy guarantees on the values and the indices 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

[2] These subpackets could be determined by the databases based on the sparse updates received at the previous time step, or by any other downlink sparsification protocol. For example, the databases can choose the most commonly updated Pr' subpackets in the writing phase of time $t-1$ to be sent to the users in the reading phase at time t.

[3] We assume that the values of the updates corresponding to all parameters in the sparse set of Pr subpackets in the writing phase are non-zero, i.e., belong to the most significant r fraction.

[4] The positions (indices) of the sparse updates leak information about the most and least significant parameters in the model for a given user, which may leak information about the user's local data.

of the databases, i.e.,

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

where $\Delta_i^{[t]}$ is the value of 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 Positions (Indices) of Sparse Updates: The amount of information leaked on the indices of the sparse updates need to be maintained under a given privacy leakage budget ϵ, i.e.,

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

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 FL process.[5] 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.

Correctness in the Reading Phase: The user should be able to correctly decode the sparse set of subpackets (denoted by J) of the model, 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 $t-1$ (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. The model should 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 subpacket s of the model at time $t-1$ and $\Delta_s^{[t]}$ is the corresponding 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 Complexity: The storage complexity 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 costs and storage complexities, while satisfying all privacy, security and correctness conditions described above.

III. Main Result

Theorem 1: Consider a FL model stored in N non-colluding databases, consisting of L parameters with values from a finite field $\mathbb{F}_q$, which are included in P subpackets. The model is divided into B segments of equal size ($1 \leq B < P$), such that each consecutive $\frac{P}{B}$ subpackets constitute each segment. Assume that the FL model is being 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 sparse (non-zero) updates in the ith segment, uploaded by any given user, and let $(\tilde{X}_1, \dots, \tilde{X}_B)$ be the general vector representing all distinct combinations of $(\hat{X}_1, \dots, \hat{X}_B)$, irrespective of the segment index.[6] Then, the reading/writing costs, storage complexities and amounts of information leakage presented in Table I are achievable in a single round of the FL process in the perspective of a single user.

Remark 1: The information leakage in Table I corresponds to the amount of information leaked on the indices of the sparse updates.[7] For a given privacy leakage budget on the indices of the sparse updates given by ϵ, the optimum number of segments B can be calculated by minimizing the storage complexity, such that $H(\hat{X}_1, \dots, \hat{X}_B) \leq \epsilon$ or $H(\tilde{X}_1, \dots, \tilde{X}_B) \leq \epsilon$ is satisfied (based on the considered case).[8] This is valid for all four cases.

Remark 2: When $B = 1$ (no segmentation present), $\hat{X}_1 = \tilde{X}_1 = Pr$ and the corresponding information leakage is zero since Pr is fixed and $H(\hat{X}_1) = H(\tilde{X}_1) = 0$, i.e., the four schemes corresponding to the four cases achieve information theoretic privacy of the values and positions of the sparse updates while incurring the same communication costs stated in Table I, when $B = 1$. However, in this case, the storage costs increase to either $O(L^2)$ or $O\left(\frac{L^2}{N^2}\right)$.

Remark 3: $H(\hat{X}_1, \dots, \hat{X}_B) > H(\tilde{X}_1, \dots, \tilde{X}_B)$ since $H(\hat{X}_1, \dots, \hat{X}_B)$ considers all possible values of $\hat{X}_i$, while $H(\tilde{X}_1, \dots, \tilde{X}_B)$ only considers distinct sets of $\{\hat{X}_i\}_{i=1}^B$. For example, if $B = 2$ and $Pr = 3$, $H(\hat{X}_1, \hat{X}_2)$ considers both permutations $\{1, 2\}$ and $\{2, 1\}$ of $(\hat{X}_1, \hat{X}_2)$ with their separate probabilities[9] while $H(\tilde{X}_1, \tilde{X}_2)$ only takes one of them into account, with the two corresponding probabilities combined i.e., the probabilities considered in $H(\tilde{X}_1, \dots, \tilde{X}_B)$ are more dense and concentrated compared to that of $H(\hat{X}_1, \dots, \hat{X}_B)$.

[5] The coordinator is only available at the initialization stage, and will not be part of the system model once the FL process begins.

[6] To explain the definitions of $\hat{X}_i$ and $\tilde{X}_i$ further, consider J_i to be the set of indices of the subpackets in segment i with non-zero (sparse) updates. Then, $\hat{X}_i$ is defined as, $\hat{X}_i = \sum_{k=1}^{\frac{P}{B}} \mathbf{1}_{\{k \in J_i\}} = |J_i|$, for each $i \in \{1, \dots, B\}$, where $\mathbf{1}_{\{\cdot\}}$ is the indicator function and $|\cdot|$ represents the cardinality. Each realization of $(\tilde{X}_1, \dots, \tilde{X}_B)$ is a representation of all permutations of the same realization of $(\hat{X}_1, \dots, \hat{X}_B)$. For example, consider a setting with $B = 2$. If the total number of sparse subpackets updated is $Pr = 3$, all realizations of $(\hat{X}_1, \hat{X}_2)$ and $(\tilde{X}_1, \tilde{X}_2)$ are given by $\{(0, 3), (3, 0), (1, 2), (2, 1)\}$ and $\{(0, 3), (1, 2)\}$, respectively.

[7] Information theoretic privacy of the values of updates is guaranteed, as stated in the problem formulation.

[8] As the storage complexity and the information leakage are inversely proportional in all four cases (see Table I), the optimum B is obtained by solving $H(\hat{X}_1, \dots, \hat{X}_B) = \epsilon$ or $H(\tilde{X}_1, \dots, \tilde{X}_B) = \epsilon$ (or the closest to ϵ) based on the chosen case, for any given distribution of $(\hat{X}_1, \dots, \hat{X}_B)$.

[9] The same is applicable for the other set of realizations $(0, 3), (3, 0)$ as well.

TABLE I

ACHIEVABLE SETS OF COMMUNICATION COSTS, STORAGE COSTS AND INFORMATION LEAKAGE

case	reading cost	writing cost	storage complexity	information leakage
1	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{2}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{2}{N}}$	$O(\frac{L^2}{B})$	$H(\hat{X}_1,\dots,\hat{X}_B)$
2	$\frac{3r'(1+\frac{\log_q P}{N})}{1-\frac{1}{N}}$	$\frac{3r(1+\log_q P)}{1-\frac{1}{N}}$	$O(\frac{L^2}{BN^2})$	$H(\hat{X}_1,\dots,\hat{X}_B)$
3	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{4}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{4}{N}}$	$\max\{O(\frac{L^2}{B}),O(N^2B^2)\}$	$H(\tilde{X}_1,\dots,\tilde{X}_B)$
4	$\frac{5r'(1+\frac{\log_q P}{N})}{1-\frac{1}{N}}$	$\frac{5r(1+\log_q P)}{1-\frac{1}{N}}$	$\max\{O(\frac{L^2}{N^2B}),O(B^2)\}$	$H(\tilde{X}_1,\dots,\tilde{X}_B)$

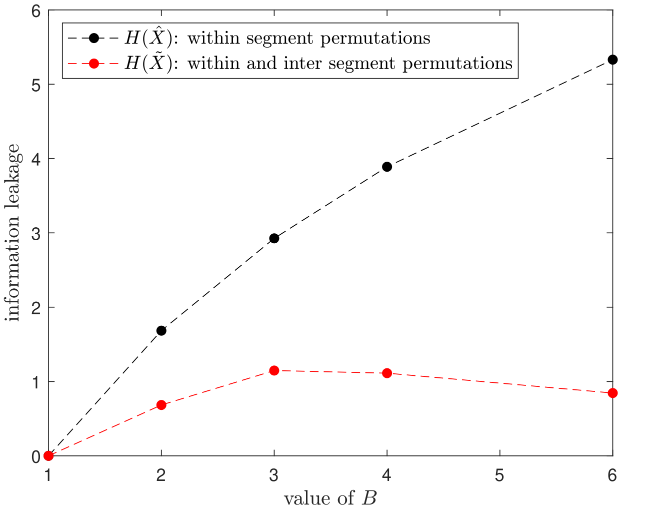

Fig. 3. Information leakage (in bits) of an example setting with $P = 12$ for different values of B.

Remark 4: Cases 1-4 are achieved by schemes that utilize both cross subspace alignment (CSA) [46] and permutation techniques which are described in detail in Section IV. The schemes for cases 1 and 2 use a single round permutation technique (only within-segment permutations) while cases 3 and 4 use a two-round permutation technique (both within and inter-segment permutations) which reduces the information leakage further. Cases 3 and 4 are extensions of cases 1 and 2, respectively, with the additional permutation round. Cases 3 and 4 incur larger communication costs compared to cases 1 and 2, while resulting in lower amounts of information leakage.

Remark 5: The four cases (schemes) have different properties. Cases 1 and 3 result in the lowest communication costs at the expense of a larger storage complexity resulted by replicated storage and larger noisy permutation reversing matrices. Cases 2 and 4 use MDS coded storage and compact permutation reversing matrices, which reduces the storage complexity at the expense of larger communication costs.

Remark 6: The communication cost does not depend on the number of segments B.

Remark 7: 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. 3.

IV. PROPOSED SCHEMES

A. General Schemes With Examples

In this section, we provide the proposed schemes for all four cases. In all four schemes, we divide the P subpackets into B non-overlapping equal-sized segments to control the storage cost and the information leakage. The parameter B is a variable that can be chosen based on the given privacy leakage budget and the limitations on the storage capacities. In this section, we present the general schemes for arbitrary values of B, P, r and r'. As a further illustration, we provide examples along with the general scheme for all four cases. In the two examples corresponding to cases 1 and 2, we assume the same setting with $P = 15$ subpackets (subpacketization ℓ), divided into $B = 3$ equal segments as shown in Fig. 4.

Case 1: Uncoded[10] storage and larger permutation reversing matrices are used in this case to reduce the communication cost, at the expense of a larger storage cost.

Initialization: A single subpacket (subpacket s) in case 1 is stored in database n, $n \in \{1, \dots, N\}$ as,

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

where $W_i^{[s]}$ is the ith parameter of subpacket s, $Z_{i,j}^{[s]}$ are random noise symbols and $\{f_i\}_{i=1}^{\ell}, \{\alpha_n\}_{n=1}^{N}$ are globally known distinct constants from $\mathbb{F}_q$. Each subpacket consists of ℓ parameters, and the subpackets in each segment are stacked one after the other in the order of subpacket 1 through subpacket $\frac{P}{B}$. At the initialization stage, the coordinator sends B ($B = 3$ for the example considered) randomly and independently chosen permutations of the $\frac{P}{B}$ ($\frac{P}{B} = 5$ for the example considered) subpackets in each of the B segments to all users. These permutations are denoted by $\tilde{P}_1, \dots, \tilde{P}_B$. The coordinator also sends the B corresponding noise added permutation reversing matrices given by,

$$R_n^{[i]} = (\tilde{R}^{[i]} \otimes \Gamma_n) + \tilde{Z}^{[i]}, \quad i = 1, \dots, B, \tag{7}$$

to database n, $n \in \{1, \dots, N\}$, as shown in Fig. 4, where $\tilde{R}^{[i]}$ is the permutation reversing matrix corresponding to the

[10] Even though the model parameters and noise symbols are combined together (coded form) in the storage in (6), each parameter is not combined with other parameters, resulting in uncoded storage.

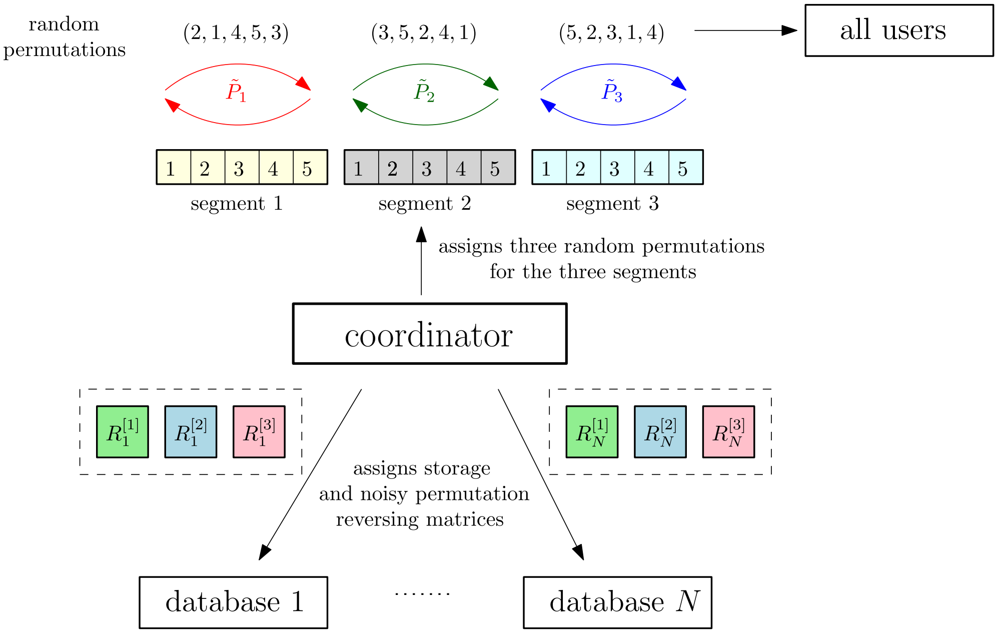

Fig. 4. Initialization of the scheme for cases 1 and 2.

permutation $\tilde{P}_i$, Γ_n is the diagonal matrix given by,

$$\Gamma_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} & & \\ & \ddots & \\ & & \frac{1}{f_\ell-\alpha_n} \end{bmatrix}, \tag{8}$$

and $\tilde{Z}^{[i]}$ is a random noise matrix of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$. Note that the databases are unaware of the underlying permutations, from Shannon's one-time-pad theorem. Based on the example considered, the permutation reversing matrix for database n, $n \in \{1, \dots, N\}$ corresponding to the first segment (permutation: $\tilde{P}_1 = (2, 1, 4, 5, 3)$) is given by,

$$R_n^{[1]} = \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \otimes \Gamma_n \right) + \tilde{Z}^{[1]} \tag{9}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]}, \tag{10}$$

Similarly, for the second segment (permutation:$\tilde{P}_2 = (3, 5, 2, 4, 1)$), the permutation reversing matrix for database n, $n \in \{1, \dots, N\}$ is given by,

$$R_n^{[2]} = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[2]}. \tag{11}$$

The coordinator leaves the system once the storage, permutations and noise added permutation reversing matrices are initialized and the system is ready to begin the FL process. All subsequent communications take place only between individual users and databases in terms of permuted subpacket indices. The databases never learn the underlying permutations despite having access to the noise added permutation reversing matrices, since the added noise $\tilde{Z}^{[i]}$ makes the noisy matrices independent of the original permutation reversing matrix from Shannon's one time pad theorem.

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, based on the permuted subpacket indices received in the writing phase at time $t-1$. For example, the databases consider the permuted indices of the subpackets updated by all users at time $t-1$, and select the most popular Pr' of them to be sent to the users in the reading phase of time t. Note that the databases are unaware of the real indices of the sparse subpacket indices updated by users in the writing phase at each time instance and only work with the permuted indices in both phases. We denote the permuted indices of the sparse subpackets to be sent to the users from segment j as $\tilde{V}_j$ for $j \in \{1, \dots, B\}$. For this example, let the sparse set of permuted subpacket indices corresponding to the first segment be $\tilde{V}_1 = \{1, 3\}$.[11] One designated database sends these permuted indices of each segment to the users. The users then find the real indices, using the known permutations as $V_j(i) = \tilde{P}_j(\tilde{V}_j(i))$ for each sparse subpacket i in segment $j \in \{1, \dots, B\}$. For this example (segment 1), the real set of indices is given by,

$$V_1(i) = \tilde{P}_1(\tilde{V}_1(i)), \quad i = 1, 2 \tag{12}$$

$$V_1 = \{2, 4\}. \tag{13}$$

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

[11] Two similar sets (with same or different cardinalities, such that the sum of all three cardinalities equals Pr') exist for segments 2 and 3 as well.

query.

$$Q_n^{[\tilde{V}_j(i)]} = \sum_{k=1}^{\ell} R_n^{[j]}(:, (i-1)\ell + k). \tag{14}$$

For example, the query corresponding to the first sparse subpacket of the first segment (i.e., $\tilde{V}_1(1) = 1$) is given by,

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, k) = \begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1, \tag{15}$$

where Z_1 is a random noise vector resulted by the noise component of $R_n^{[1]}$. Similarly, the query for the second sparse subpacket of segment 1 (i.e., $\tilde{V}_1(2) = 3$) is given by,

$$Q_n^{[\tilde{V}_1(2)]} = Q_n^{[3]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, 2\ell + k) = \begin{bmatrix} 0_\ell \\ 0_\ell \\ 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_\ell \end{bmatrix} + Z_2. \tag{16}$$

Note that the reversal of the permutations is hidden from the databases by the random noise vectors Z_1 and Z_2. Then, database n, $n \in \{1, \dots, N\}$ sends the answer corresponding to the ith sparse subpacket of segment j to all users by calculating the dot product between the queries and the scaled storage as,

$$\begin{aligned} A_n^{[\tilde{V}_j(i)]} &= (D_n \times S_n)^T Q_n^{[\tilde{V}_j(i)]}, \quad j \in \{1, \dots, B\} && (17) \\ &= \frac{1}{f_1 - \alpha_n} W_1^{[V_j(i)]} + \dots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[V_j(i)]} \\ &\quad + P_{\alpha_n}(\ell + 1), && (18) \end{aligned}$$

where D_n is the diagonal matrix of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$ given by,

$$D_n = I_{\frac{P}{B}} \otimes \Gamma_n^{-1} = \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}, \tag{19}$$

with $I_{\frac{P}{B}}$ being the identity matrix of size $\frac{P}{B} \times \frac{P}{B}$, S_n is the concatenation of the $\frac{P}{B}$ subpackets of the form (6) in the segment under consideration, and $P_{\alpha_n}(\ell + 1)$ is a polynomial in α_n of degree $\ell + 1$. For example, the answer of database n,$n \in \{1, \dots, N\}$ corresponding to the first sparse subpacket of segment 1 (i.e., $\tilde{V}_1(1) = 1$) is given by,

$$\begin{aligned} & A_n^{[\tilde{V}_1(1)]} \\ &= (D_n \times S_n)^T Q_n^{[\tilde{V}_1(1)]} && (20) \\ &= \left(\begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[1]} + \sum_{j=0}^{\ell} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[1]} + \sum_{j=0}^{\ell} \alpha_n^j I_{\ell,j} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[5]} + \sum_{j=0}^{\ell} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[5]} + \sum_{j=0}^{\ell} \alpha_n^j I_{\ell,j} \end{bmatrix} \end{bmatrix} \right)^T \\ &\quad \times \left(\begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1 \right) && (21) \\ &= \frac{1}{f_1 - \alpha_n} W_1^{[2]} + \dots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell + 1). && (22) \end{aligned}$$

Now, the users obtain the parameters of real subpacket 2 of segment 1, (i.e., $V_1(1) = \tilde{P}_1(\tilde{V}_1(1)) = 2$) by solving,

$$\begin{aligned} & \begin{bmatrix} A_1^{\tilde{V}_1(1)} \\ \vdots \\ A_N^{\tilde{V}_1(1)} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \dots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{\ell+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \dots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{\ell+1} \end{bmatrix} \begin{bmatrix} W_1^{[2]} \\ \vdots \\ W_\ell^{[2]} \\ \xi_0 \\ \vdots \\ \xi_{\ell+1} \end{bmatrix}, && (23) \end{aligned}$$

where ξ_i are the coefficents of the polynomial $P_{\alpha_n}(\ell + 1)$ in (22). Note that (23) (and the corresponding general set of equations in (18)) is solvable given that $N = 2\ell + 2$, which determines the subpacketization as $\ell = \frac{N-2}{2}$. The same procedure described above is carried out for all sparse subpackets in each of the B segments. The resulting reading cost (including both data and permuted index downloads) is given by,

$$\begin{aligned} C_R &= \frac{Pr'(\log_q \frac{P}{B} + \log_q B) + Pr'N}{L} && (24) \\ &= \frac{Pr'(\log_q P + N)}{P\frac{N-2}{2}} && (25) \\ &= \frac{2r'(1 + \frac{\log_q P}{N})}{1 - \frac{2}{N}}. && (26) \end{aligned}$$

Writing Phase: In the writing phase, each user selects the Pr subpackets with the most significant updates and sends the corresponding noise added combined updates (single bit per subpacket) along with their permuted subpacket indices to each of the databases. The noise added combined update of real subpacket i of segment j (assuming this subpacket is among the Pr selected subpackets) sent to database n, $n \in \{1,\dots,N\}$) is given by,

$$U_n^{[i,j]} = \sum_{k=1}^{\ell} \prod_{r=1, r\neq k}^{\ell} (f_r - \alpha_n)\tilde{\Delta}_k^{[i,j]} + \prod_{r=1}^{\ell}(f_r-\alpha_n)Z^{[i,j]}, \tag{27}$$

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 bit of the sparse subpacket i of segment j and $Z^{[i,j]}$ is a random noise bit. To determine the permuted subpacket index of subpacket i of segment j, consider the permutation assigned for segment j (i.e., $\tilde{P}_j$) to be a one-to-one mapping from the set $\{1,\dots,\frac{P}{B}\}$ to the set $\tilde{P}_j$ in the exact order. Then, the permuted subpacket index corresponding to subpacket i of segment j is given by,

$$Y^{[i,j]} = \tilde{P}_j^{-1}(i), \quad j \in \{1,\dots,B\}. \tag{28}$$

Once the combined updates and permuted subpacket indices corresponding to all Pr chosen subpackets are computed, the user uploads the Pr (update, subpacket, segment) tuples to all databases.[12]

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 combined updates $U_n^{[2,1]}$, $U_n^{[4,1]}$, $U_n^{[2,2]}$ and $U_n^{[5,3]}$ which are of the form (27). The corresponding permuted subpacket indices are given by,

$$Y^{[2,1]} = \tilde{P}_1^{-1}(2) = 1 \tag{29}$$

$$Y^{[4,1]} = \tilde{P}_1^{-1}(4) = 3 \tag{30}$$

$$Y^{[2,2]} = \tilde{P}_2^{-1}(2) = 3 \tag{31}$$

$$Y^{[5,3]} = \tilde{P}_3^{-1}(5) = 1. \tag{32}$$

Therefore, the two permuted (update, subpacket, segment) tuples corresponding to segment 1, sent by the user to database n are given by, $(U_n^{[2,1]},1,1)$ and $(U_n^{[4,1]},3,1)$. Similarly, the permuted (update, subpacket, segment) tuples corresponding to segments 2 and 3 are given by $(U_n^{[2,2]},3,2)$ and $(U_n^{[5,3]},1,3)$, respectively.[13] Once database n, $n \in \{1,\dots,N\}$ receives the Pr (update, subpacket, segment) tuples, it creates the permuted update vectors $\hat{Y}_n^{[j]}$ for each segment $j \in \{1,\dots,B\}$ given by,

$$\hat{Y}_n^{[j]} = \sum_{i=1}^{\frac{P}{B}} U_n^{[i,j]} e_{\frac{P}{B}}(Y^{[i,j]}), \tag{33}$$

where $e_{\frac{P}{B}}(Y^{[i,j]})$ is the all zeros vector of size $\frac{P}{B}\times 1$ with a 1 at the $Y^{[i,j]}$th position. Note that we consider $U_n^{[i,j]} = 0$ for those values of i, $i \in \{1,\dots,\frac{P}{B}\}$ whose corresponding subpackets are not included in the set of Pr selected subpackets. In order to reverse the permutation privately, each database creates,

$$\hat{U}_n^{[j]} = \hat{Y}_n^{[j]} \otimes 1_\ell = [\hat{Y}_n^{[j]}(1)\cdot 1_\ell, \dots, \hat{Y}_n^{[j]}(\frac{P}{B})\cdot 1_\ell]^T, \tag{34}$$

for each segment j, where 1_ℓ is the all ones vector of size $\ell \times 1$. For the example considered, the $\hat{U}_n^{[j]}$ vectors for the three segments, generated by database n, $n \in \{1,\dots,N\}$ based on the received information $(U_n^{[2,1]},1,1)$, $(U_n^{[4,1]},3,1)$,$(U_n^{[2,2]},3,2)$ and $(U_n^{[5,3]},1,3)$ are given by,

$$\hat{U}_n^{[1]} = \begin{bmatrix} U_n^{[2,1]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ U_n^{[4,1]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix}, \hat{U}_n^{[2]} = \begin{bmatrix} 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ U_n^{[2,2]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix}, \hat{U}_n^{[3]} = \begin{bmatrix} U_n^{[5,3]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix}. \tag{35}$$

Next, the databases privately rearrange the updates in the real order and calculate the incremental updates of each segment as $\bar{U}_n^{[j]} = R_n^{[j]}\hat{U}_n^{[j]}$ for $j \in \{1,\dots,B\}$, and add it to the jth segment of the existing storage to obtain the updated storage. Consider the incremental update calculation of segment 1 in database n, $n \in \{1,\dots,N\}$ for the example considered,

$$\bar{U}_n^{[1]} = R_n^{[1]}\hat{U}_n^{[1]} \tag{36}$$

$$= \left(\begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right) \begin{bmatrix} U_n^{[2,1]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ U_n^{[4,1]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \tag{37}$$

$$= \begin{bmatrix} 0_\ell \\ \frac{U_n^{[2,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{U_n^{[2,1]}}{f_\ell-\alpha_n} \\ 0_\ell \\ \frac{U_n^{[4,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{U_n^{[4,1]}}{f_\ell-\alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0_\ell \\ \frac{\Delta_1^{[2,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[2,1]}}{f_\ell-\alpha_n} \\ 0_\ell \\ \frac{\Delta_1^{[4,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[4,1]}}{f_\ell-\alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell), \tag{38}$$

where $P_{\alpha_n}(\ell)$ here are vectors of size $\frac{P\ell}{B}$ consisting of polynomials in α_n of degree ℓ, and the last equality is obtained by applying Lemma 1 in [47]. 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,

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \quad n \in \{1,\dots,N\}. \tag{39}$$

Note from (38) 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

[12] Note that the 'subpacket' and 'segment' elements in the (update, subpacket, segment) refer to the permuted subpacket index and the real segment index, respectively.

[13] Note that there is no permutation in the segment index, and only the subpacket indices within each segment is being permuted.

the real subpacket indices 2 and 4 to any of the databases. The resulting writing cost is given by,

$$C_W = \frac{PrN(1+\log_q B + \log_q \frac{P}{B})}{L} \tag{40}$$

$$= \frac{PrN(1+\log_q P)}{P^{\frac{N-2}{2}}} \tag{41}$$

$$= \frac{2r(1+\log_q P)}{1-\frac{2}{N}}. \tag{42}$$

The total storage complexity (including both data and the permutation reversing matrices) is given by $O(L)+O(\frac{L^2}{B^2}\times B) = O(\frac{L^2}{B})$. The information leakage is derived in Section IV-B.

Case 2: MDS coded storage and smaller permutation reversing matrices are used in this case to reduce the storage cost, at the expense of a larger communication cost. The information leakage is the same for both cases 1 and 2, since they both use only within-segment permutations. The same example considered for case 1 (shown in Fig. 4) is considered in this case as well.

Initialization: A single subpacket s in case 2 is stored in database n, $n \in \{1,\dots,N\}$ as,

$$S_n^{[s]} = \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[s]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[s]}. \tag{43}$$

Therefore, the storage of segment j, $j \in \{1,\dots,B\}$ is given by,

$$S_n^{[j]} = \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,j]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[1,j]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[\frac{P}{B},j]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[\frac{P}{B},j]} \end{bmatrix}, \tag{44}$$

where $W_i^{[s,j]}$ is the ith parameter of subpacket s in segment j and $Z_i^{[s,j]}$ are random noise symbols. Note that $\frac{P}{B}=5$ for the example considered. Similar to case 1, the coordinator initializes all noise terms in storage, assigns B permutations $\tilde{P}_i$, $i \in \{1,\dots,B\}$ of the subpackets in each of the B segments and sends them to the users, and sends the corresponding B noise added permutation reversing matrices $R_n^{[i]}$, $i \in \{1,\dots,B\}$ to database n, $n \in \{1,\dots .N\}$, as shown in Fig. 4. The noise added permutation reversing matrices are of the form,

$$R_n^{[i]} = \bar{R}^{[i]} + \alpha_n^{\ell} \bar{Z}^{[i]}, \quad i \in \{1,\dots,B\}, \tag{45}$$

where $\bar{R}^{[i]}$ is the permutation reversing matrix corresponding to the ith permutation $\tilde{P}_i$, and $\bar{Z}^{[i]}$ is a random noise matrix, both of size $\frac{P}{B}\times\frac{P}{B}$. Note that the databases are unaware of the underlying permutations, from Shannon's one-time-pad theorem. The noise added permutation reversing matrices corresponding to the three segments, sent to database n, $n \in \{1,\dots,N\}$ for the example in Fig. 4 are given by (recall: $\tilde{P}_1=(2,1,4,5,3)$, $\tilde{P}_2=(3,5,2,4,1)$, $\tilde{P}_3=(5,2,3,1,4)$),

$$R_n^{[1]} = \begin{bmatrix} 0&1&0&0&0\\ 1&0&0&0&0\\ 0&0&0&0&1\\ 0&0&1&0&0\\ 0&0&0&1&0 \end{bmatrix} + \alpha_n^{\ell}\bar{Z}^{[1]} \tag{46}$$

$$R_n^{[2]} = \begin{bmatrix} 0&0&0&0&1\\ 0&0&1&0&0\\ 1&0&0&0&0\\ 0&0&0&1&0\\ 0&1&0&0&0 \end{bmatrix} + \alpha_n^{\ell}\bar{Z}^{[2]} \tag{47}$$

$$R_n^{[3]} = \begin{bmatrix} 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&1\\ 1&0&0&0&0 \end{bmatrix} + \alpha_n^{\ell}\bar{Z}^{[3]}. \tag{48}$$

As explained in case 1, the coordinator leaves the system once the FL process begins, and all subsequent communications take place only between the users and databases using permuted subpacket indices.

Reading Phase: As described in case 1, let $\tilde{V}_j$ be the set of permuted indices of the sparse subpackets chosen from segment j to be sent to the users for $j \in \{1,\dots,B\}$. For example, let $\tilde{V}_1 = \{1,3\}$ be the permuted set of sparse subpackets of segment 1 that needs to be sent to the users at time t. One designated database sends the permuted subpacket indices of each segment (segment 1: $\tilde{V}_1 = \{1,3\}$) to the users, from which the users identify the corresponding real sparse subpacket indices using the known permutations using $V_j(i) = \tilde{P}_j(\tilde{V}_j(i))$, where V_j is the vector containing the real indices of the sparse subpackets in segment j. In particular, the users perform the same calculation in (13) for segment 1 as well as for the other two segments, based on the received sets $\tilde{V}_2$ and $\tilde{V}_3$. Similar to case 1, each database generates a query to send each of the chosen subpackets. The query corresponding to the ith permuted sparse subpacket of segment j is given by,

$$Q_n^{[\tilde{V}_j(i)]} = R_n^{[j]}(:,\tilde{V}_j(i)), \quad j \in \{1,\dots,B\}. \tag{49}$$

Following are the two queries generated by database n, $n \in \{1,\dots,N\}$ to send the two sparse subpackets (with permuted indices $\tilde{V}_1 = \{1,3\}$) of segment 1 to the users,

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = R_n^{[1]}(:,1) = \begin{bmatrix} 0\\1\\0\\0\\0 \end{bmatrix} + \alpha_n^{\ell}\hat{Z}_1 \tag{50}$$

$$Q_n^{[\tilde{V}_1(2)]} = Q_n^{[3]} = R_n^{[1]}(:,3) = \begin{bmatrix} 0\\0\\0\\1\\0 \end{bmatrix} + \alpha_n^{\ell}\hat{Z}_3, \tag{51}$$

where $\hat{Z}_1$ and $\hat{Z}_3$ are the first and third columns of $\bar{Z}^{[1]}$ in (46). Then, database n, $n \in \{1,\dots,N\}$ sends the answers corresponding to each sparse subpacket of each segment to the users as,

$$A_n^{[\tilde{V}_j(i)]} = (S_n^{[j]})^T Q_n^{[\tilde{V}_j(i)]}, \quad j \in \{1,\dots,B\} \tag{52}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[V_j(i),j]} + P_{\alpha_n}(2\ell), \tag{53}$$

where $P_{\alpha_n}(2\ell)$ is a polynomial in α_n of degree 2ℓ. For example, the answer corresponding to the first sparse subpacket of segment 1 ($\tilde{V}_1(1) = 1$), sent by database n, $n \in \{1, \dots, N\}$ to the users is given by,

$$A_n^{[\tilde{V}_1(1)]} = (S_n^{[1]})^T Q_n^{[\tilde{V}_1(1)]} \tag{54}$$

$$= \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,1]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[1,1]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[5,1]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[5,1]} \end{bmatrix}^T \left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_1 \right) \tag{55}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[2,1]} + P_{\alpha_n}(2\ell). \tag{56}$$

The users obtain the parameters of the real subpacket 2 of segment 1, (i.e., $V_1(1) = \tilde{P}_1(\tilde{V}_1(1)) = 2$) by solving,

$$\begin{bmatrix} A_1^{\tilde{V}_1(1)} \\ \vdots \\ A_N^{\tilde{V}_1(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_1^{\ell}} & \dots & \frac{1}{\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{2\ell} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\alpha_N^{\ell}} & \dots & \frac{1}{\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{2\ell} \end{bmatrix} \begin{bmatrix} W_{\ell}^{[2,1]} \\ \vdots \\ W_1^{[2,1]} \\ \xi_0 \\ \vdots \\ \xi_{2\ell} \end{bmatrix} \tag{57}$$

where ξ_i are the coefficients of the polynomial in (56). Note that (57) (and also the general answers in (53)) is solvable given that $N = 3\ell + 1$, which determines the subpacketization as $\ell = \frac{N-1}{3}$. The same procedure described above is carried out for all sparse subpackets in each of the B segments. The resulting reading cost is given by,

$$C_R = \frac{Pr' \log_q P + Pr'N}{L} \tag{58}$$

$$= \frac{Pr'(\log_q P + N)}{P\frac{N-1}{3}} \tag{59}$$

$$= \frac{3r'(1 + \frac{\log_q P}{N})}{1 - \frac{1}{N}}. \tag{60}$$

Writing Phase: In the writing phase, the user generates Pr combined updates corresponding to the Pr subpackets with the most significant updates. The combined update of the ith subpacket of segment j is defined as (assuming this subpacket is among the Pr selected subpackets),

$$U_n^{[i,j]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} \Delta_k^{[i,j]} + Z^{[i,j]}, \tag{61}$$

where $\Delta_k^{[i,j]}$ is the update of the kth bit of the ith subpacket of segment j and $Z^{[i,j]}$ is a random noise symbol. Similar to case 1, the user generates the permuted subpacket indices corresponding to the real subpacket indices i of each segment j, of the selected Pr subpackets, using (28). Once the permuted subpacket indices are generated, the user sends the permuted (update, subpacket, segment) tuples of the Pr selected subpackets to all databases similar to case 1. For the same example considered in case 1 where the user wants to update the real subpackets 2 and 4 from segment 1, subpacket 2 from segment 2 and subpacket 5 from segment 3, the user sends the same permuted (update, subpacket, segment) tuples sent in case 1 given by, $(U_n^{[2,1]}, 1, 1), (U_n^{[4,1]}, 3, 1)$, $(U_n^{[2,2]}, 3, 2)$, $(U_n^{[5,3]}, 1, 3)$ to database n, $n \in \{1, \dots, N\}$ assuming the same three permutations given by $\tilde{P}_1 = \{2, 1, 4, 5, 3\}$, $\tilde{P}_2 = \{3, 5, 2, 4, 1\}$ and $\tilde{P}_3 = \{5, 2, 3, 1, 4\}$, for the three segments. Once the databases receive the Pr (update, subpacket, segment) tuples, they create the permuted update vectors $\hat{Y}_n^{[j]}$ for each segment $j \in \{1, \dots, B\}$ using (33). For the example considered, the three permuted update vectors created at database n are given by,

$$\hat{Y}_n^{[1]} = \begin{bmatrix} U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \\ 0 \end{bmatrix}, \quad \hat{Y}_n^{[2]} = \begin{bmatrix} 0 \\ 0 \\ U_n^{[2,2]} \\ 0 \\ 0 \end{bmatrix}, \quad \hat{Y}_n^{[3]} = \begin{bmatrix} U_n^{[5,3]} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{62}$$

based on the received information $(U_n^{[2,1]}, 1, 1), (U_n^{[4,1]}, 3, 1)$, $(U_n^{[2,2]}, 3, 2)$, $(U_n^{[5,3]}, 1, 3)$. Using the permuted update vectors $\hat{Y}_n^{[j]}$, $j \in \{1, \dots, B\}$, database n, $n \in \{1, \dots, N\}$ privately calculates the correctly rearranged incremental update vector of each segment as $\bar{U}_n^{[j]} = R_n^{[j]} \hat{Y}_n^{[j]}$, $j \in \{1, \dots, B\}$. The resulting incremental update is of the same form as the storage in (44), and therefore can be added to the existing storage to obtain the updated storage of each segment. To explain the above process in terms of an example, consider the incremental update of segment 1 in the same example considered so far,

$$\bar{U}_n^{[1]} = R_n^{[1]} \hat{Y}_n^{[1]} \tag{63}$$

$$= \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]} \right) \begin{bmatrix} U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \\ 0 \end{bmatrix} \tag{64}$$

$$= \begin{bmatrix} 0 \\ U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0 \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,1]} \\ 0 \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[4,1]} \\ 0 \end{bmatrix} + P_{\alpha_n}(\ell), \tag{65}$$

where $P_{\alpha_n}(\ell)$ here is a vector of size 5×1, consisting of polynomials in α_n of degree ℓ. Since the incremental update of each segment (i.e., (65)) is in the same form as the storage in (44), the incremental update is directly added to the existing storage to obtain it's updated version, i.e.,

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \; j \in \{1, \dots, B\}, \; n \in \{1, \dots, N\}. \tag{66}$$

The writing cost for case 2 is given by,

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} \tag{67}$$

$$= \frac{PrN(1 + \log_q P)}{P\frac{N-1}{3}} \tag{68}$$

$$= \frac{3r(1+\log_q P)}{1-\frac{1}{N}}. \tag{69}$$

The total storage complexity (including both data and the permutation reversing matrices) is given by $O(P) + O(\frac{P^2}{B^2} \times B) = O(\frac{P^2}{B}) = O(\frac{L^2}{BN^2})$. The information leakage is derived in Section IV-B.

Case 3: In this case, we use uncoded storage with large permutation reversing matrices. Note that in both cases 1 and 2, 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 further. However, this increases the storage cost since the permutation of segment indices requires an additional noise added permutation reversing matrix to be stored at the databases. The communication cost is not significantly affected by the additional round of permutation, compared to case 1 (lowest communication cost thus far).

For cases 3 and 4, we present the general scheme along with the example setting with $P = 12$ subpackets (with subpacketization ℓ) which are divided into and $B = 3$ equal segments, as shown in Fig. 5.

Initialization: The storage of a single subpacket (subpacket s) in case 3 is given by,

$$S_n^{[s]} = \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[s]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[s]} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[s]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[s]} \end{bmatrix}, \tag{70}$$

with the same notation as in case 1. The subpackets are stacked one after the other (subpacket 1 through subpacket $\frac{P}{B}$ in each segment) in the order of segment 1 through segment B. At the initialization stage, the coordinator sends B randomly and independently chosen permutations of the $\frac{P}{B}$ subpackets in each of the B segments, $\tilde{P}_1, \ldots, \tilde{P}_B$, as well as a randomly and independently chosen permutation of the B segments $\hat{P}$ to the users. The coordinator also places the corresponding noise added permutation reversing matrices (corresponding to B within-segments permutations: $R_n^{[1]}, \ldots, R_n^{[B]}$ and one inter-segment permutation: $\hat{R}_n$) at each database n, $n \in \{1, \ldots, N\}$. The noise added permutation reversing matrix corresponding to the ith segment, $i \in \{1, \ldots, B\}$ stored in database n, $n \in \{1, \ldots, N\}$ is given by,

$$R_n^{[i]} = (\tilde{R}^{[i]} \otimes \Gamma_n) + \tilde{Z}^{[i]}, \tag{71}$$

with the same notation as in case 1. The noise added permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$ stored in database n, $n \in \{1, \ldots, N\}$ is given by,

$$\hat{R}_n = (\bar{R} \otimes I_\ell) + (I_B \otimes \Gamma_n^{-1})\hat{Z} = \begin{bmatrix} b_{1,1}^{[n]} & \cdots & b_{1,B}^{[n]} \\ \vdots & \ddots & \vdots \\ b_{B,1}^{[n]} & \cdots & b_{B,B}^{[n]} \end{bmatrix}, \tag{72}$$

where $\bar{R}$ is the permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$, I_k is the identity matrix of size $k \times k$, Γ_n^{-1} is the diagonal matrix given by,

$$\Gamma_n^{-1} = \begin{bmatrix} f_1 - \alpha_n & & \\ & \ddots & \\ & & f_\ell - \alpha_n \end{bmatrix} \tag{73}$$

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 the last equality in (72). Note that the databases are unaware of the underlying permutations, based on Shannon's one-time-pad theorem. According to the given permutations in Figure 5, the noise added permutation reversing matrix corresponding to the first within-segment permutation ($\tilde{P}_1 = (2,4,3,1)$), i.e., $R_n^{[1]}$ is given by,

$$R_n^{[1]} = \left(\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \otimes \Gamma_n \right) + \tilde{Z}^{[1]} \tag{74}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]}. \tag{75}$$

Similarly, $R_n^{[2]}$ and $R_n^{[3]}$ are given by,

$$R_n^{[2]} = \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} + \tilde{Z}^{[2]} \tag{76}$$

$$R_n^{[3]} = \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[3]}, \tag{77}$$

For the same example, the inter-segment noise added permutation reversing matrix for database n, $n \in \{1, \ldots, N\}$ is given by,

$$\hat{R}_n = \left(\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes I_\ell \right) + \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \Gamma_n^{-1} \right) \hat{Z} \tag{78}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & I_\ell \\ I_\ell & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & I_\ell & 0_{\ell\times\ell} \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} \end{bmatrix} \hat{Z} \tag{79}$$

$$= \begin{bmatrix} b_{1,1}^{[n]} & b_{1,2}^{[n]} & b_{1,3}^{[n]} \\ b_{2,1}^{[n]} & b_{2,2}^{[n]} & b_{2,3}^{[n]} \\ b_{3,1}^{[n]} & b_{3,2}^{[n]} & b_{3,3}^{[n]} \end{bmatrix}, \tag{80}$$

where I_ℓ is the identity matrix of size $\ell \times \ell$. Each matrix $\hat{R}_n$ is represented in blocks of size $\ell \times \ell$, as shown in (80), which is useful in the subsequent calculations. The coordinator leaves the system once the storage and permutations are initialized, and the noise added permutation reversing matrices are placed at the databases, before the FL process begins. All communications in the FL process are carried out between the

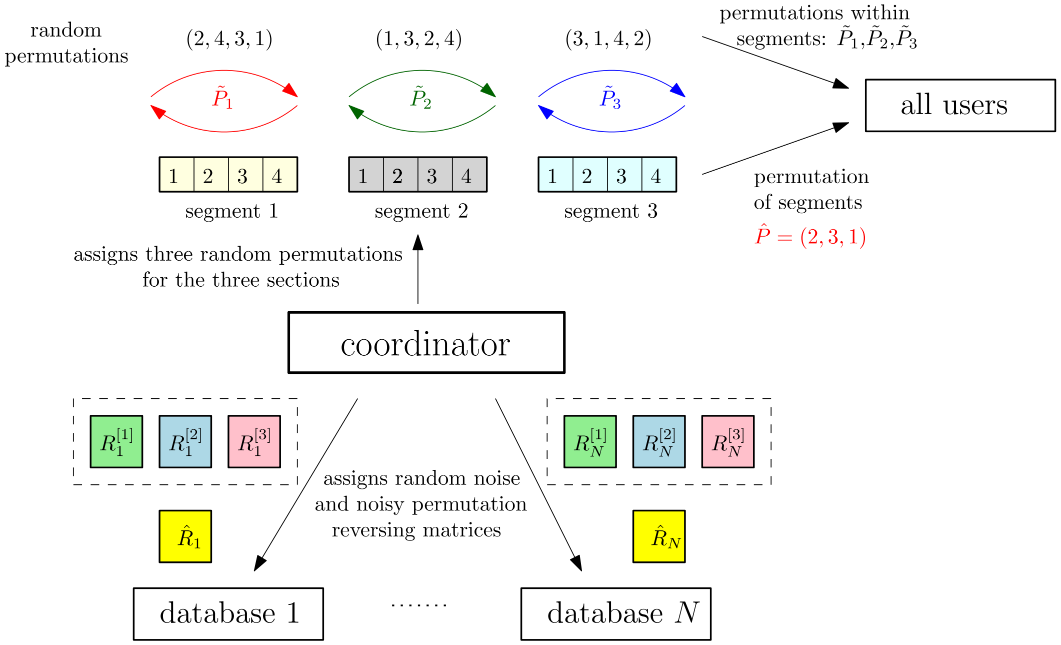
random permutations
$(2,4,3,1)$
$(1,3,2,4)$
$(3,1,4,2)$
permutations within segments: $\tilde{P}_1$,$\tilde{P}_2$,$\tilde{P}_3$
all users
$\tilde{P}_1$
$\tilde{P}_2$
$\tilde{P}_3$
1 2 3 4
segment 1
1 2 3 4
segment 2
1 2 3 4
segment 3
permutation of segments
$\hat{P}=(2,3,1)$
assigns three random permutations for the three sections
coordinator
$R_1^{[1]}$ $R_1^{[2]}$ $R_1^{[3]}$
$R_N^{[1]}$ $R_N^{[2]}$ $R_N^{[3]}$
assigns random noise and noisy permutation reversing matrices
$\hat{R}_1$
$\hat{R}_N$
database 1
....
database N

Fig. 5. Initialization of the scheme for cases 3 and 4.

users and databases using permuted subpacket and segment indices.

Reading Phase: The databases determine the set of Pr' subpackets to be sent to the users at time t, based on the permuted information received from the users in the writing phase at time $t-1$. For example, the databases consider the permuted indices of the subpackets updated by all users at time $t-1$, and select the most popular Pr' of them (in terms of permuted indices) to be sent to the users in the reading phase of time t. In case 3, the databases are unaware of the real indices of subpackets within each segment, as well as the corresponding real inidces of the segments, updated by the users. However, the databases can communicate the sparse subpacket and segment indices with the users in their permuted versions (same as what was received in the writing phase at time $t-1$). The users are able to convert the permuted indices into their real versions as all permutations are known by the users. Let the permuted (subpacket, segment) information of each of the Pr' subpackets be indicated by (η_p, ϕ_p). This information is sent to all users at time t by one designated database. The users can convert each permuted (η_p, ϕ_p) into its real versions (η_r, ϕ_r) using,

$$\phi_r = \hat{P}(\phi_p) \tag{81}$$

$$\eta_r = \tilde{P}_{\phi_r}(\eta_p). \tag{82}$$

For the example in Fig. 5, assume that the following permuted (subpacket, segment) pairs are received by a given user from the designated database,

$$(\eta_p, \phi_p) = \{(1,3),(1,1),(1,2)\}. \tag{83}$$

Recall that the permutations considered in this example are given by $\tilde{P}_1 = (2,4,3,1)$, $\tilde{P}_2 = (1,3,2,4)$, $\tilde{P}_3 = (3,1,4,2)$ and $\hat{P} = (2,3,1)$. In order to see how the real subpacket and segment indices are obtained, consider the first permuted pair $(1,3)$. Since the permuted segment index is $\phi_p = 3$, the corresponding real segment index is $\phi_r = \hat{P}(3) = 1$. Then, the user can decode the subpacket index within the first segment as, $\eta_r = \tilde{P}_1(1) = 2$. Therefore, the real (subpacket, segment) pair corresponding to the permuted (subpacket, segment) pair $(\eta_p, \phi_p) = (1,3)$ is given by $(\eta_r, \phi_r) = (2,1)$. Similarly, the real set of (subpacket, segment) pairs corresponding to the three permuted pairs are given by,

$$(\eta_r, \phi_r) = \{(2,1),(1,2),(3,3)\}. \tag{84}$$

Once the real indices of the sparse subpackets and segments are obtained, the user downloads the corresponding subpackets, one by one. To perform the calculations in the reading phase, each database first generates a combined noise added permutation reversing matrix, that combines the within-segment and inter-segment noise added permutation reversing matrices into a single noise added permutation reversing matrix, to facilitate the subsequent calculations. The combined noise added permutation reversing matrix of database n, $n \in \{1,\dots,N\}$ is given by,

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times \begin{bmatrix} I_{\frac{P}{B}} \otimes b_{1,1}^{[n]} & \dots & I_{\frac{P}{B}} \otimes b_{1,B}^{[n]} \\ \vdots & \vdots & \vdots \\ I_{\frac{P}{B}} \otimes b_{B,1}^{[n]} & \dots & I_{\frac{P}{B}} \otimes b_{B,B}^{[n]} \end{bmatrix} \tag{85}$$

$$= \begin{bmatrix} R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix} & \dots & R_n^{[1]} \begin{bmatrix} b_{1,B}^{[n]} & & \\ & \ddots & \\ & & b_{1,B}^{[n]} \end{bmatrix} \\ & \ddots & \\ R_n^{[B]} \begin{bmatrix} b_{B,1}^{[n]} & & \\ & \ddots & \\ & & b_{B,1}^{[n]} \end{bmatrix} & \dots & R_n^{[B]} \begin{bmatrix} b_{B,B}^{[n]} & & \\ & \ddots & \\ & & b_{B,B}^{[n]} \end{bmatrix} \end{bmatrix} \tag{86}$$

$$= \dot{R}_n + P_{\alpha_n}(1), \tag{87}$$

where $\dot{R}_n$ is the combined permutation reversing matrix obtained by replacing the 1 in the ith row of $\bar{R}$ in (72) by $R^{[i]} \otimes \Gamma_n$ in (71), and the zeros by all zeros matrices of size $\frac{P\ell}{B} \times \frac{P\ell}{B}$. $P_{\alpha_n}(1)$ is a matrix of size $L \times L$ consisting of elements that are degree 1 polynomials in α_n.

As an example, consider the generation of the combined noise added permutation reversing matrix of database n, $n \in \{1, \dots, N\}$, for the example setting in Fig. 5,

$$R_n = \begin{bmatrix} R_n^{[1]} & 0 & 0 \\ 0 & R_n^{[2]} & 0 \\ 0 & 0 & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} I_4 \otimes b_{1,1}^{[n]} & \dots & I_4 \otimes b_{1,3}^{[n]} \\ \vdots & \vdots & \vdots \\ I_4 \otimes b_{3,1}^{[n]} & \dots & I_4 \otimes b_{3,3}^{[n]} \end{bmatrix} \tag{88}$$

Note that,

$$\begin{aligned} &R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \\ &= \left(\begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right) \\ &\quad \times \begin{bmatrix} \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} \end{bmatrix} \qquad (90) \\ &= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \hat{Z}_{1,1} \\ \hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \hat{Z}_{1,1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + P_{\alpha_n}(1) \qquad (91) \\ &= P_{\alpha_n}(1) \qquad (92) \end{aligned}$$

where $\hat{Z}_{1,1}$ is the submatrix of $\hat{Z}$ in (79) consisting of the first ℓ rows and first ℓ columns, $P_{\alpha_n}(1)$ here are matrices of size $4\ell \times 4\ell$, consisting of polynomials in α_n of degree 1 and $0_{\ell\times\ell}$ is the all zeros matrix of size $\ell \times \ell$. Next, consider the calculation of,

$$\begin{aligned} &R_n^{[1]} \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \\ &= \left(\begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right) \\ &\times \left(\begin{bmatrix} I_\ell & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & I_\ell & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & I_\ell & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & I_\ell \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1}\hat{Z}_{1,3} & & \\ & \ddots & \\ & & \Gamma_n^{-1}\hat{Z}_{1,3} \end{bmatrix} \right) \qquad (93) \\ &= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + P_{\alpha_n}(1), \qquad (94) \end{aligned}$$

where $\hat{Z}_{1,3}$ is the submatrix of $\hat{Z}$ in (79) consisting of the first ℓ rows and the column indices given by $2\ell+1$ to 3ℓ and $P_{\alpha_n}(1)$ is a matrix of size $4\ell \times 4\ell$ consisting of polynomials of α_n of degree 1. Based on similar calculations, we can write the combined permutation reversing matrix R_n in (89) as (95), shown at the bottom of the next page. The $P_{\alpha_n}(1)$ term in (95) is a matrix of size $12\ell \times 12\ell$, whose elements are polynomials in α_n of degree 1. Note that the first part of the combined noise added permutation reversing matrix R_n is simply the 1 in the ith row of $\bar{R}$ (permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$) replaced by $\tilde{R}^{[i]} \otimes \Gamma_n$, where $\tilde{R}^{[i]}$ is the permutation reversing matrix corresponding to the within-segment permutation $\tilde{P}_i$, for each $i \in \{1, 2, 3\}$.

In order to download the subpacket corresponding to the permuted pair (η_p, ϕ_p), each database generates the query given by,

$$Q_n^{[\eta_p,\phi_p]} = (I_P \otimes \Gamma_n^{-1}) \times \sum_{k=1}^{\ell} R_n\left(:, (\phi_p - 1)\frac{P}{B}\ell + (\eta_p - 1)\ell + k\right), \tag{96}$$

where I_P is the identity matrix of size $P \times P$. Then, database n, $n \in \{1, \dots, N\}$ sends the answer corresponding to the permuted subpacket (η_p, ϕ_p) as,

$$\begin{aligned} A_n^{[\eta_p,\phi_p]} &= S_n^T Q_n^{[\eta_p,\phi_p]} \qquad (97) \\ &= \frac{1}{f_1 - \alpha_n} W_1^{[\eta_r,\phi_r]} + \dots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[\eta_r,\phi_r]} \\ &\quad + P_{\alpha_n}(\ell + 3), \qquad (98) \end{aligned}$$

where S_n is the concatenation of all P subpackets of the form (70), $P_{\alpha_n}(\ell+3)$ is a polynomial in α_n of degree $\ell+3$. Then, the user can obtain the values of the corresponding real subpacket indicated by (η_r, ϕ_r), i.e., $W_1^{[\eta_r,\phi_r]}, \dots, W_\ell^{[\eta_r,\phi_r]}$ using all answers received by the N databases as,

$$\begin{bmatrix} A_1^{[\eta_p,\phi_p]} \\ \vdots \\ A_N^{[\eta_p,\phi_p]} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \dots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{\ell+3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \dots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{\ell+3} \end{bmatrix} \times \begin{bmatrix} W_1^{[\eta_r,\phi_r]} \\ \vdots \\ W_\ell^{[\eta_r,\phi_r]} \\ \xi_{0:\ell+3} \end{bmatrix}, \tag{99}$$

where each ξ_i corresponds to the ith coefficient of the polynomial $P_{\alpha_n}(\ell+3)$ in (98). The equation in (99) is solvable if $N = 2\ell + 4$, which determines the subpacketization as $\ell = \frac{N-4}{2}$.

As an example, consider the download of the permuted subpacket indicated by $(\eta_p, \phi_p) = (1, 3)$, from the same example setting in Fig. 5. Database n, $n \in \{1, \dots, N\}$ creates the query given by,

$$Q_n^{[1,3]} = (I_{12} \otimes \Gamma_n^{-1}) \times \sum_{k=1}^{\ell} R_n(:, 8\ell + k) \tag{100}$$

$$= \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}_{12\ell \times 12\ell} \times \left(\begin{bmatrix} 0_\ell \\ \frac{1}{f_1 - \alpha_n} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} \\ 0_{10\ell} \end{bmatrix} + \dot{P}_{\alpha_n}(1) \right) \tag{101}$$

$$= \begin{bmatrix} 0_\ell \\ 1_\ell \\ 0_{10\ell} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1 - \alpha_n) P_{\alpha_n}(1) \\ \vdots \\ (f_\ell - \alpha_n) P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_1 - \alpha_n) P_{\alpha_n}(1) \\ \vdots \\ (f_\ell - \alpha_n) P_{\alpha_n}(1) \end{bmatrix} \end{bmatrix}_{L \times 1}, \tag{102}$$

where $\dot{P}_{\alpha_n}(1)$ is a vector of size $12\ell \times 1$, consisting of polynomials in α_n of degree 1, and $P_{\alpha_n}(1)$ are polynomials of α_n of degree 1. Note that 0_k and 1_k refer to all zeros and all ones vectors of size $k \times 1$, respectively. The answer corresponding to the above query, sent to the users by database n, $n \in \{1, \dots, N\}$ is given by,

$$A_n^{[1,3]} = S_n^T Q_n^{[1,3]} \tag{103}$$

$$= \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1 - \alpha_n} W_1^{[1]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[1]} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} W_\ell^{[1]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_1 - \alpha_n} W_1^{[12]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[12]} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} W_\ell^{[12]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[12]} \end{bmatrix} \end{bmatrix}^T \times \left(\begin{bmatrix} 0_\ell \\ 1_\ell \\ 0_{10\ell} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1 - \alpha_n) P_{\alpha_n}(1) \\ \vdots \\ (f_\ell - \alpha_n) P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_1 - \alpha_n) P_{\alpha_n}(1) \\ \vdots \\ (f_\ell - \alpha_n) P_{\alpha_n}(1) \end{bmatrix} \end{bmatrix}_{L \times 1} \right) \tag{104}$$

$$= \frac{1}{f_1 - \alpha_n} W_1^{[2]} + \dots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell + 3), \tag{105}$$

$$R_n = \begin{bmatrix} R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[1]} \begin{bmatrix} b_{1,2}^{[n]} & & \\ & \ddots & \\ & & b_{1,2}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[1]} \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \\ R_n^{[2]} \begin{bmatrix} b_{2,1}^{[n]} & & \\ & \ddots & \\ & & b_{2,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[2]} \begin{bmatrix} b_{2,2}^{[n]} & & \\ & \ddots & \\ & & b_{2,2}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[2]} \begin{bmatrix} b_{2,3}^{[n]} & & \\ & \ddots & \\ & & b_{2,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \\ R_n^{[3]} \begin{bmatrix} b_{3,1}^{[n]} & & \\ & \ddots & \\ & & b_{3,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[3]} \begin{bmatrix} b_{3,2}^{[n]} & & \\ & \ddots & \\ & & b_{3,2}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[3]} \begin{bmatrix} b_{3,3}^{[n]} & & \\ & \ddots & \\ & & b_{3,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \end{bmatrix}. \tag{89}$$

$$R_n = \begin{bmatrix} 0_{4\ell \times 4\ell} & 0_{4\ell \times 4\ell} & \begin{bmatrix} 0_{\ell \times \ell} & 0_{\ell \times \ell} & 0_{\ell \times \ell} & \Gamma_n \\ \Gamma_n & 0_{\ell \times \ell} & 0_{\ell \times \ell} & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & 0_{\ell \times \ell} & \Gamma_n & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & \Gamma_n & 0_{\ell \times \ell} & 0_{\ell \times \ell} \end{bmatrix} \\ \begin{bmatrix} \Gamma_n & 0_{\ell \times \ell} & 0_{\ell \times \ell} & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & 0_{\ell \times \ell} & \Gamma_n & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & \Gamma_n & 0_{\ell \times \ell} & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & 0_{\ell \times \ell} & 0_{\ell \times \ell} & \Gamma_n \end{bmatrix} & 0_{4\ell \times 4\ell} & 0_{4\ell \times 4\ell} \\ 0_{4\ell \times 4\ell} & \begin{bmatrix} 0_{\ell \times \ell} & \Gamma_n & 0_{\ell \times \ell} & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & 0_{\ell \times \ell} & 0_{\ell \times \ell} & \Gamma_n \\ \Gamma_n & 0_{\ell \times \ell} & 0_{\ell \times \ell} & 0_{\ell \times \ell} \\ 0_{\ell \times \ell} & 0_{\ell \times \ell} & \Gamma_n & 0_{\ell \times \ell} \end{bmatrix} & 0_{4\ell \times 4\ell} \end{bmatrix} + P_{\alpha_n}(1), \tag{95}$$

from which the (real) second subpacket of segment 1, i.e., $(\eta_r, \phi_r) = (2,1)$ can be obtained using all answers of the N databases if $N = 2\ell + 4$ is satisfied, which determines the subpacketization as $\ell = \frac{N-4}{2}$. The resulting reading cost is given by,

$$C_R = \frac{Pr'(N + \log_q B + \log_q \frac{P}{B})}{L} \tag{106}$$

$$= \frac{Pr'(N + \log_q P)}{P\frac{N-4}{2}} \tag{107}$$

$$= \frac{2r'(1 + \frac{\log_q P}{N})}{1 - \frac{4}{N}}. \tag{108}$$

Writing Phase: In the writing phase, the user sends the combined updates, permuted subpacket indices and permuted segment indices of the Pr subpackets with the most significant updates to all databases. Let $(\eta_r^{[i]}, \phi_r^{[i]})$, $i \in \{1, \dots, Pr\}$ be the real (subpacket, segment) pair corresponding to the ith selected subpacket. For each of the Pr selected subpackets, the user generates a combined update bit given by,

$$U_n^{[\eta_r^{[i]},\phi_r^{[i]}]} = \sum_{k=1}^{\ell} \prod_{j=1, j\neq k}^{\ell} (f_j - \alpha_n)\tilde{\Delta}_k^{[\eta_r^{[i]},\phi_r^{[i]}]} + \prod_{j=1}^{\ell}(f_j - \alpha_n) Z^{[\eta_r^{[i]},\phi_r^{[i]}]}, \quad i \in \{1, \dots, Pr\}, \tag{109}$$

where $\tilde{\Delta}_k^{[\eta_r^{[i]},\phi_r^{[i]}]} = \frac{\Delta_k^{[\eta_r^{[i]},\phi_r^{[i]}]}}{\prod_{j=1, j\neq k}^{\ell}(f_j - f_k)}$, with $\Delta_k^{[\eta_r^{[i]},\phi_r^{[i]}]}$ being the update of the kth parameter of subpacket $\eta_r^{[i]}$ of segment $\phi_r^{[i]}$ and $Z^{[\eta_r^{[i]},\phi_r^{[i]}]}$ is a random noise symbol. The user sends the permuted (update, subpacket, segment) tuple given by $(U_n^{[\eta_r^{[i]},\phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$ for the ith selected subpacket where $U_n^{[\eta_r^{[i]},\phi_r^{[i]}]}$ is the combined update of the subpacket of the form (109), $\eta_p^{[i]}$ is the permuted subpacket index obtained by,

$$\eta_p^{[i]} = \tilde{P}_{\phi_r^{[i]}}^{-1}(\eta_r^{[i]}) \tag{110}$$

and $\phi_p^{[i]}$ is the permuted segment index obtained by,

$$\phi_p^{[i]} = \hat{P}^{-1}(\phi_r^{[i]}) \tag{111}$$

where $\tilde{P}_{\phi_r^{[i]}}$ and $\hat{P}$ are considered to be the one-to-one mappings from j to $\tilde{P}_{\phi_r^{[i]}}(j)$ for $j = 1, \dots, \frac{P}{B}$ and i to $\hat{P}(i)$, for $i = 1, \dots, B$, respectively. For the example in Fig. 5, assume that a given user wants to update the Pr sparse subpackets identified by the real (subpacket, segment) pairs given by, $(\eta_r, \phi_r) = \{(2,1), (2,2), (3,3)\}$. Based on the 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 segment-wise permutation given by $\hat{P} = (2,3,1)$, the user sends the following (permuted) information to database n, $n \in \{1, \dots, N\}$,

$$(U_n^{[\eta_r,\phi_r]}, \eta_p, \phi_p) = \{(U_n^{[2,1]}, \tilde{P}_1^{-1}(2), \hat{P}^{-1}(1)), (U_n^{[2,2]}, \tilde{P}_2^{-1}(2), \hat{P}^{-1}(2)), (U_n^{[3,3]}, \tilde{P}_3^{-1}(3), \hat{P}^{-1}(3))\} \tag{112}$$

$$= \{(U_n^{[2,1]}, 1, 3), (U_n^{[2,2]}, 3, 1), (U_n^{[3,3]}, 1, 2)\}, \tag{113}$$

where the three $U_n^{[\eta_r,\phi_r]}$ terms are generated as in (109). As an illustration, the real (subpacket, segment) pair given by $(2,1)$, is converted to the permuted pair $(1,3)$ as follows. Note that in the segment-wise permutation $\hat{P} = (2,3,1)$, the real segment index 1 lies in the third position. Therefore, $\phi_r = 1$ is converted to $\phi_p = 3$. Next, in the permutation corresponding to segment 1 ($\tilde{P}_1 = (2,4,3,1)$), the subpacket index 2 lies in the first position. Therefore, $\eta_r = 2$ is converted to $\eta_p = 1$.

Once the databases receive all permuted (update, subpacket, segment) tuples, they construct the permuted update vector as,

$$\tilde{U}_n = \sum_{i=1}^{Pr} U_n^{[\eta_r^{[i]},\phi_r^{[i]}]} e_P((\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]}), \tag{114}$$

where $e_p((\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]})$ is the all zeros vector of size P with a 1 at the $(\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]}$th position. This vector is then scaled by an all ones vector of size $\ell \times 1$ (i.e., 1_ℓ) to aid the rest of the calculations. The scaled permuted update vector is given by,

$$\hat{U}_n = \tilde{U}_n \otimes 1_\ell = \begin{bmatrix} \tilde{U}_n(1) \cdot 1_\ell \\ \vdots \\ \tilde{U}_n(P) \cdot 1_\ell \end{bmatrix}. \tag{115}$$

Then, database n, $n \in \{1, \dots, N\}$ calculates the incremental update using the combined noise added permutation reversing matrix in (87) as,

$$\bar{U}_n = R_n \times \hat{U}_n, \tag{116}$$

which is of the same form as the storage in (70). Therefore, the storage at time t, $S_n^{[t]}$ can be updated as,

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n. \tag{117}$$

For the example considered, the scaled permuted update vector based on the information received by the databases in (113) is given by,

$$\hat{U}_n = (U_n^{[2,1]} e_{12}(9) + U_n^{[2,2]} e_{12}(3) + U_n^{[3,3]} e_{12}(5)) \otimes 1_\ell \tag{118}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[2,2]} \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[3,3]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \end{bmatrix} \tag{119}$$

Then, database n, $n \in \{1, \dots, N\}$ computes the incremental update using the combined noise added permutation reversing matrix in (95) as shown in (120)-(123), at the bottom of the next page, where $P_{\alpha_n}(\ell + 1)$ is a vector of size $L \times 1$ whose elements are polynomials in α_n of degree $\ell + 1$, and (123) follows from Lemma 1 in [47]. Note from (123) that the (real) subpacket 2 of segment 1, subpacket 2 of segment 2 and subpacket 3 of segment 3 are correctly updated, without revealing these real indices to any of the databases.[14] Since $\bar{U}_n$ is in the same form as the storage in (70), the storage at time t can be updated by adding $\bar{U}_n$ to the existing storage. The resulting writing cost is given by,

$$C_W = \frac{PrN(1+\log_q B + \log_q \frac{P}{B})}{L} \tag{124}$$

$$= \frac{PrN(1+\log_q P)}{P^{\frac{N-4}{2}}} \tag{125}$$

$$= \frac{2r(1+\log_q P)}{1-\frac{4}{N}}. \tag{126}$$

The storage complexities of data, within-segment noise added permutation reversing matrices and the inter-segment noise added permutation reversing matrix are given by $O(L)$, $O(\frac{L^2}{B})$ and $O(\ell^2 B^2) = O(N^2B^2)$, respectively. Therefore the storage complexity is $\max\{O(\frac{L^2}{B}), O(N^2B^2)\}$. The information leakage (on the indices of sparse updates) is derived in Section IV-B.

Case 4: In this case, we consider coded storage and smaller permutation reversing matrices to reduce the storage cost. Both within-segment and inter-segment permutations are considered in this case to reduce the information leakage.

Initialization: The storage of a single subpacket s in database n, $n \in \{1,\dots,N\}$ is given by,

$$S_n^{[s]} = \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[s]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[s]}, \tag{127}$$

with the same notation used in case 2. Therefore, the storage of a given segment j, $j \in \{1,\dots,B\}$ is given by,

$$S_{n,j} = \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,j]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,j]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[\frac{P}{B},j]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[\frac{P}{B},j]} \end{bmatrix}, \tag{128}$$

where $W_i^{[k,j]}$ is the ith parameter of subpacket k of segment j and $Z_i^{[k,j]}$ are random noise symbols. The segments are stacked one after the other in the order of segment 1 through segment B. As described in case 3, the coordinator sends the within-segment and inter-segment permutations $\tilde{P}_1, \dots, \tilde{P}_B$ and $\hat{P}$ to the users and the corresponding noise added permutation reversing matrices given by $R_n^{[1]}, \dots, R_n^{[B]}$ and $\hat{R}_n$ to database n, $n \in \{1,\dots,N\}$. The noise added permutation reversing matrices for the within segment permutations $\tilde{P}_i$ are of the form (45), and the noise added permutation reversing matrix corresponding to the inter-segment permutation $\hat{P}$ is of the form,

$$\hat{R}_n = \hat{R} + \alpha_n^{\ell} Z, \tag{129}$$

where $\hat{R}$ is the permutation reversing matrix corresponding to $\hat{P}$ and Z is a random noise matrix, both of size $B \times B$. Note that the databases are unaware of the underlying permutations, from Shannon's one-time-pad theorem. For the example considered in Figure 5, the noise added permutation reversing matrices corresponding to $\tilde{P}_1 = \{2,4,3,1\}$, $\tilde{P}_2 = \{1,3,2,4\}$, $\tilde{P}_3 = \{3,1,4,2\}$ and $\hat{P} = \{2,3,1\}$, stored in database n,

$$\bar{U}_n = R_n \times \hat{U}_n \tag{120}$$

$$= \begin{bmatrix} 0_{4\ell\times 4\ell} & 0_{4\ell\times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} \\ \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} & 0_{4\ell\times 4\ell} & 0_{4\ell\times 4\ell} \\ 0_{4\ell\times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} & 0_{4\ell\times 4\ell} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ U_n^{[2,2]}\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[3,3]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[2,1]}\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \\ 0\cdot 1_\ell \end{bmatrix} \end{bmatrix}$$

$$+ P_{\alpha_n}(\ell+1) \tag{121}$$

$$= \left[0_\ell, \frac{U_n^{[2,1]}}{f_1-\alpha_n}, \dots, \frac{U_n^{[2,1]}}{f_\ell-\alpha_n}, 0_{3\ell}, \frac{U_n^{[2,2]}}{f_1-\alpha_n}, \dots, \frac{U_n^{[2,2]}}{f_\ell-\alpha_n}, 0_{4\ell}, \frac{U_n^{[3,3]}}{f_1-\alpha_n}, \dots, \frac{U_n^{[3,3]}}{f_\ell-\alpha_n}, 0_\ell\right]^T + P_{\alpha_n}(\ell+1) \tag{122}$$

$$= \left[0_\ell, \frac{\Delta_1^{[2,1]}}{f_1-\alpha_n}, \dots, \frac{\Delta_\ell^{[2,1]}}{f_\ell-\alpha_n}, 0_{3\ell}, \frac{\Delta_1^{[2,2]}}{f_1-\alpha_n}, \dots, \frac{\Delta_\ell^{[2,2]}}{f_\ell-\alpha_n}, 0_{4\ell}, \frac{\Delta_1^{[3,3]}}{f_1-\alpha_n}, \dots, \frac{\Delta_\ell^{[3,3]}}{f_\ell-\alpha_n}, 0_\ell\right]^T + P_{\alpha_n}(\ell+1), \tag{123}$$

[14] Note that the vector $P_{\alpha_n}(\ell+1)$ in (123) hides these non-zero updates from the databases.

$n \in \{1, \dots, N\}$ are given by,

$$R_n^{[1]} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[1]} \tag{130}$$

$$R_n^{[2]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[2]} \tag{131}$$

$$R_n^{[3]} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[3]} \tag{132}$$

and

$$\hat{R}_n = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \alpha_n^\ell Z. \tag{133}$$

The initialization stage ends and the coordinator leaves the system once the storage, permutations and the noise added permutation reversing matrices are initialized.

To aid the calculations in the reading and writing phases described next, the databases compute a combined noise added permutation reversing matrix as described in case 3. This matrix for database n, $n \in \{1, \dots, N\}$ is given by,

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times (\hat{R}_n \otimes I_{\frac{P}{B}}) \tag{134}$$

$$= \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times \begin{bmatrix} \hat{R}_n(1,1) I_{\frac{P}{B}} & \dots & \hat{R}_n(1,B) I_{\frac{P}{B}} \\ \vdots & \vdots & \vdots \\ \hat{R}_n(B,1) I_{\frac{P}{B}} & \dots & \hat{R}_n(B,B) I_{\frac{P}{B}} \end{bmatrix} \tag{135}$$

where $I_{\frac{P}{B}}$ is the identity matrix of size $\frac{P}{B} \times \frac{P}{B}$. The combined matrix R_n places $R_n^{[i]}$ at the position of the 1 in the ith row of $\hat{R}$ in (129), with added noise. The combined noise added permutation reversing matrix for the example considered in Fig. 5 for database n, $n \in \{1, \dots, N\}$ is calculated in (136)-(138), shown at the bottom of the next page, where $\tilde{Z}$ in (137) is a random noise matrix of size 12×12, resulted by the noise component of $\hat{R}_n$ and $P_{\alpha_n}(\ell)$ in (138) is a matrix of size 12×12 with entries consisting of polynomials in α_n of degree ℓ. Note that the combined noise added permutation reversing matrix in (138) places each $R_n^{[i]}$ at the position of the 1 in the ith row of the permutation reversing matrix in (133) for $i = 1, 2, 3$, with added noise.

Reading Phase: As explained in case 3, the databases determine the permuted (subpacket, segment) tuples given by (η_p, ϕ_p) for the Pr' sparse subpackets and send them to the users. The users obtain the real (subpacket, segment) information of the (η_p, ϕ_p) tuples from (81) and (82). For the example considered in Fig. 5, the (η_p, ϕ_p) and (η_r, ϕ_r) pairs in (83) and (84) considered in case 3 are valid for case 4 as well.

In order to send the subpacket corresponding to the permuted (subpacket, segment) pair (η_p, ϕ_p), database n, $n \in \{1, \dots, N\}$ creates a query given by,

$$Q_n^{[\eta_p, \phi_p]} = R_n(:, (\phi_p - 1)\frac{P}{B} + \eta_p). \tag{139}$$

For $(\eta_p, \phi_p) = (1, 3)$, the corresponding query is given by,

$$Q_n^{[1,3]} = R_n(:, 9) = \begin{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ 0_8 \end{bmatrix} + \alpha_n^\ell P_{\alpha_n}(\ell), \tag{140}$$

where $P_{\alpha_n}(\ell)$ here is a vector of size 12×1 with entries consisting of polynomials in α_n of degree ℓ. The databases send the answer to each query corresponding to (η_p, ϕ_p) as,

$$A_n^{[\eta_p, \phi_p]} = S_n^T Q_n^{[\eta_p, \phi_p]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} W_k^{[\eta_r, \phi_r]} + P_{\alpha_n}(4\ell), \tag{141}$$

where S_n is the concatenation of all B segments of the form (128), $P_{\alpha_n}(4\ell)$ is a polynomial in α_n of degree 4ℓ. The users can obtain the parameters of the corresponding real (subpacket, segment) pair (η_r, ϕ_r) using,

$$\begin{bmatrix} A_1^{[\eta_p, \phi_p]} \\ \vdots \\ A_N^{[\eta_p, \phi_p]} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_1^\ell} & \dots & \frac{1}{\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{4\ell} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\alpha_N^\ell} & \dots & \frac{1}{\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{4\ell} \end{bmatrix} \begin{bmatrix} W_\ell^{[\eta_r, \phi_r]} \\ \vdots \\ W_1^{[\eta_r, \phi_r]} \\ \xi_{0:4\ell} \end{bmatrix}, \tag{142}$$

where ξ_i are the coefficients of the polynomial in (141), if $N = 5\ell + 1$ is satisfied. This determines the subpacketization as $\ell = \frac{N-1}{5}$. For the example considered, the answer for $(\eta_p, \phi_p) = (1, 3)$ from database n, $n \in \{1, \dots, N\}$ is given by,

$$A_n^{[1,3]} = S_n^T Q_n^{[1,3]} \tag{143}$$

$$= \begin{bmatrix} \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,1]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,1]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,1]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,1]} \end{bmatrix} \\ \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,2]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,2]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,2]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,2]} \end{bmatrix} \\ \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,3]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,3]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,3]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,3]} \end{bmatrix} \end{bmatrix}^T \times \left(\begin{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ 0_8 \end{bmatrix} + \alpha_n^\ell P_{\alpha_n}(\ell) \right) \tag{144}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[2,1]} + P_{\alpha_n}(4\ell). \tag{145}$$

The users can obtain the parameters of the (real) second subpacket of segment 1 (since the real indices corresponding to permuted $(\eta_p, \phi_p) = (1, 3)$ are $(\eta_r, \phi_r) = (2, 1)$ from (83) and (84).) using the N answers received if $N = 5\ell+1$ is satisfied. This defines the subpacketization for case 4 as $\ell = \frac{N-1}{5}$. The resulting reading cost is given by,

$$C_R = \frac{Pr'(N + \log_q B + \log_q \frac{P}{B})}{L} \tag{146}$$

$$= \frac{Pr'(N + \log_q P)}{P\frac{N-1}{5}} \tag{147}$$

$$= \frac{5r'(1 + \frac{\log_q P}{N})}{1 - \frac{1}{N}}. \tag{148}$$

Writing Phase: Similar to case 3, the user selects the Pr subpackets with the most significant updates and let $(\eta_r^{[i]}, \phi_r^{[i]})$, $i \in \{1, \dots, Pr\}$ be the real (subpacket, segment) information of the ith selected subpacket. For each such subpacket, the user generates a combined update (single symbol) given by,

$$U_n^{[\eta_r^{[i]},\phi_r^{[i]}]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} \Delta_k^{[\eta_r^{[i]},\phi_r^{[i]}]} + Z^{[\eta_r^{[i]},\phi_r^{[i]}]}, \tag{149}$$

where $\Delta_k^{[\eta_r^{[i]},\phi_r^{[i]}]}$ is the update of the kth parameter of subpacket $\eta_r^{[i]}$ of segment $\phi_r^{[i]}$, and $Z^{[\eta_r^{[i]},\phi_r^{[i]}]}$ is a random noise symbol. The user sends the permuted (update, subpacket, segment) tuple given by $(U_n^{[\eta_r^{[i]},\phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$ for the ith sparse subpacket for $i \in \{1, \dots, Pr\}$ where $U_n^{[\eta_r^{[i]},\phi_r^{[i]}]}$ is the

$$R_n = \begin{bmatrix} R_n^{[1]} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & R_n^{[2]} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} \hat{R}_n(1,1)I_4 & \hat{R}_n(1,2)I_4 & \hat{R}_n(1,3)I_4 \\ \hat{R}_n(2,1)I_4 & \hat{R}_n(2,2)I_4 & \hat{R}_n(2,3)I_4 \\ \hat{R}_n(3,1)I_4 & \hat{R}_n(3,2)I_4 & \hat{R}_n(3,3)I_4 \end{bmatrix} \tag{136}$$

$$= \begin{bmatrix} \begin{bmatrix} 0&0&0&1\\1&0&0&0\\0&0&1&0\\0&1&0&0 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[1]} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[2]} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[3]} \end{bmatrix}$$

$$\times \left(\begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 1&&\\&\ddots&\\&&1 \end{bmatrix}_{4\times4} \\ \begin{bmatrix} 1&&\\&\ddots&\\&&1 \end{bmatrix}_{4\times4} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 1&&\\&\ddots&\\&&1 \end{bmatrix}_{4\times4} & 0_{4\times4} \end{bmatrix} + \alpha_n^\ell \tilde{Z} \right) \tag{137}$$

$$= \begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0&0&0&1\\1&0&0&0\\0&0&1&0\\0&1&0&0 \end{bmatrix} \\ \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1 \end{bmatrix} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0 \end{bmatrix} & 0_{4\times4} \end{bmatrix} + \alpha_n^\ell P_{\alpha_n}(\ell), \tag{138}$$

combined update of the subpacket of the form (149), $\eta_p^{[i]}$ is the permuted subpacket index obtained by $\eta_p^{[i]} = \tilde{P}^{-1}_{\phi_r^{[i]}}(\eta_r^{[i]})$ and $\phi_p^{[i]}$ is the permuted segment index obtained by $\phi_p^{[i]} = \hat{P}^{-1}(\phi_r^{[i]})$, with the same notation used in the description of case 3. For the example considered, assume that a user wants to update real (subpacket, segment) pairs given by $(\eta_r, \phi_r) = \{(2,1),(2,2),(3,3)\}$. Based on the 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 inter-segment permutation given by $\hat{P} = (2,3,1)$, the user sends the following (permuted) information to database n, $n \in \{1,\dots,N\}$, as described in case 3,

$$(U_n, \eta_p, \phi_p) = \{(U_n^{[2,1]},1,3),(U_n^{[2,2]},3,1),(U_n^{[3,3]},1,2)\}. \tag{150}$$

Based on the permuted sparse update tuples $(U_n^{[\eta_r^{[i]},\phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$, $i \in \{1,\dots,Pr\}$ received, database n, $n \in \{1,\dots,N\}$ constructs the permuted update vector given in (114). Then, database n, $n \in \{1,\dots,N\}$ calculates the permutation-reversed incremental update as,

$$\bar{U}_n = R_n \tilde{U}_n, \tag{151}$$

which is of the same form as the storage in (128). Therefore, the storage at time t can be updated as,

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n. \tag{152}$$

For the example considered, the permuted update vector from (114) is given by,

$$\tilde{U}_n = [0,0,U_n^{[2,2]},0,U_n^{[3,3]},0,0,0,U_n^{[2,1]},0,0,0]^T. \tag{153}$$

Then, each database calculates the permutation-reversed incremental update as shown in (154)-(157), shown at the bottom of the next page, where $P_{\alpha_n}(2\ell)$ is a vector of size 12×1, consisting of polynomials in α_n of degree 2ℓ. Note that the (real) subpacket 2 of segment 1, subpacket 2 of segment 2 and subpacket 3 of segment 3, i.e., $(\eta_r, \phi_r) = \{(2,1),(2,2),(3,3)\}$, are correctly placed in (157), without revealing the real indices to the databases.[15] Since the incremental update in (157) is of the same form as (128), it is directly added to the existing storage to obtain the updated storage. The writing cost is given by,

$$C_W = \frac{PrN(1+\log_q B + \log_q \frac{P}{B})}{L} \tag{158}$$

$$= \frac{PrN(1+\log_q P)}{P^{\frac{N-1}{5}}} \tag{159}$$

$$= \frac{5r(1+\log_q P)}{1-\frac{1}{N}}. \tag{160}$$

The storage complexities of data, noise added within and inter-segment permutation reversing matrix are given by $O(P) = O(\frac{L}{N})$, $O(\frac{P^2}{B}) = O(\frac{L^2}{N^2 B})$ and $O(B^2)$, respectively. Therefore, the storage complexity is $\max\{O(\frac{L^2}{N^2 B}), O(B^2)\}$. The information leakage on the indices of the sparse updates is derived in Section IV-B.

B. Information Leakage

In this section, we quantify the information leakage of the four cases for a given FL setting with N databases, P subpackets, B segments and uplink and downlink sparsficiation rates give by r, r'. In the proposed scheme, the users send no information to the databases in the reading phase, and the databases determine the sparse set of subpackets and send them to the users. Therefore, there is no information leakage in the reading phase. In the writing phase, the users send Pr permuted tuples of the form (update, subpacket, segment) to databases, from which a given amount of information about the sparse subpacket indices updated by a given user is allowed to leak.

There are two types of information within a given user's uploads that leak information about the user's local data, namely, 1) values of sparse updates, 2) positions of sparse updates. Note that in the proposed scheme, a random noise symbol is added to all combined sparse updates sent to the databases in all four cases. Therefore, from Shannon's one time pad theorem, the combined update values sent by the user to all databases are random noise symbols which are independent of the values of the sparse updates included in it. Therefore, the amount of information leaked by the values of the sparse updates in this work is zero.

From the set of permuted (update, subpacket, segment) tuples sent by a given user at time t, only the (subpacket, segment) pairs may possibly contain information about the real positions of the sparse updates, since the *update* component is simply random noise that is independent of the real positions of the sparse updates. Let $Y^{[t]}$ be the set of Pr subpacket indices corresponding to the set of permuted (subpacket, segment) pairs sent by a given user at time t. Let $X^{[t]}$ be the set of real indices of the Pr sparse subpackets of the model, chosen to be updated by the user at time t. Therefore, the amount of information leaked to the databases about the real positions of the sparse updates at time t is quantified by the mutual information between $X^{[t]}$ and $Y^{[t]}$, i.e., $I(X^{[t]};Y^{[t]})$. In this section, we quantify this mutual information for all four cases.

Cases 1 and 2: In cases 1 and 2, permutations exist only within each segment and not among segments.[16] To simplify the notation, we drop the time index in the following calculation, and assume that each X and Y correspond to real and permuted quantities at time t. In order to quantify $I(X;Y)$, we first derive the following conditional probability.

$$P(X=x|Y=y)$$
$$= \frac{\sum_{\tilde{p}_1,\dots,\tilde{p}_B} P(X=x, Y=y, \tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B)}{P(Y=y)} \tag{161}$$

$$= \left(\sum_{\tilde{p}_1,\dots,\tilde{p}_B} P(Y=y|X=x,\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B) \times P(X=x)\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i)\right) / P(Y=y) \tag{162}$$

[15] The sparse updates in the first part of (157) are hidden from the databases by the noise vector $P_{\alpha_n}(2\ell)$.

[16] Recall that these random permutations are not known by the databases.

$$= \left(P(X=x)\sum_{\tilde{p}_1,\dots,\tilde{p}_B}\mathbf{1}_{\{Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B\}}\right.$$

$$\left.\prod_{i=1}^{B}P(\tilde{P}_i=\tilde{p}_i)\right)/P(Y=y) \tag{163}$$

$$= \begin{cases} \frac{P(X=x)\prod_{i=1}^{B}\hat{y}_i!\prod_{i=1}^{B}(\frac{P}{B}-\hat{y}_i)!}{(\frac{P}{B})!^{B}P(Y=y)}, & \text{for } x,y:\hat{x}_i=\hat{y}_i,\ \forall\ i\\ 0, & \text{otherwise}\end{cases} \tag{164}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\prod_{i=1}^{B}\frac{\hat{y}_i!(\frac{P}{B}-\hat{y}_i)!}{(\frac{P}{B})!}, & \text{for } x,y:\hat{x}_i=\hat{y}_i,\ \forall\ i\\ 0, & \text{otherwise}\end{cases} \tag{165}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}, & \text{for } x,y:\hat{x}_i=\hat{y}_i,\ \forall\ i\\ 0, & \text{otherwise}\end{cases} \tag{166}$$

where $\hat{x}_i$ and $\hat{y}_i$ are the numbers of real and permuted sparse subpackets in segment i, respectively, and (162) is obtained by the mutual independence of the permutations of the B segments and the real positions of the sparse updates. (164) is derived by counting the number of all possible permutations that result in the given $Y=y$ from the given $X=x$. Next, we compute the probability,

$$P(Y=y) = \sum_{\tilde{p}_1,\dots,\tilde{p}_B}\sum_{x}P(Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B) \tag{167}$$

$$= \sum_{\tilde{p}_1,\dots,\tilde{p}_B}\sum_{x}P(Y=y|X=x,\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B)$$

$$\times P(X=x)\prod_{i=1}^{B}P(\tilde{P}_i=\tilde{p}_i) \tag{168}$$

$$= \sum_{x}P(X=x)$$

$$\times \sum_{\tilde{p}_1,\dots,\tilde{p}_B}\mathbf{1}_{\{Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B\}}\prod_{i=1}^{B}P(\tilde{P}_i=\tilde{p}_i) \tag{169}$$

$$= \sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)\frac{\prod_{i=1}^{B}\hat{y}_i!\prod_{i=1}^{B}(\frac{P}{B}-\hat{y}_i)!}{(\frac{P}{B})!^{B}} \tag{170}$$

$$= \prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x) \tag{171}$$

for each y such that $\sum_{i=1}^{B}\hat{y}_i = Pr$. Therefore, from (166),

$$P(X=x|Y=y)$$

$$= \begin{cases} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)}, & \text{for } x,y:\hat{x}_i=\hat{y}_i,\ \forall i\\ 0, & \text{otherwise.}\end{cases} \tag{172}$$

Then,

$$H(X|Y=y)$$

$$= -\sum_{x}P(X=x|Y=y)\log P(X=x|Y=y) \tag{173}$$

$$= -\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)}\log\frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)}, \tag{174}$$

from which we obtain,

$$H(X|Y)$$

$$= \sum_{y}P(Y=y)H(X|Y=y) \tag{175}$$

$$= -\sum_{y}\left(\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}\sum_{x:\hat{x}_i=\hat{y}_i,\forall i}P(X=x)\right)$$

$$\bar{U}_n = R_n\tilde{U}_n \tag{154}$$

$$= \left(\begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix}0&0&0&1\\1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix}\\ \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} & 0_{4\times4} & 0_{4\times4}\\ 0_{4\times4} & \begin{bmatrix}0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0\end{bmatrix} & 0_{4\times4}\end{bmatrix} + \alpha_n^{\ell}P_{\alpha_n}(\ell)\right)\times\begin{bmatrix}0\\0\\U_n^{[2,2]}\\0\\U_n^{[3,3]}\\0\\0\\0\\U_n^{[2,1]}\\0\\0\\0\end{bmatrix} \tag{155}$$

$$= [0,U_n^{[2,1]},0,0,0,U_n^{[2,2]},0,0,0,0,U_n^{[3,3]},0]^T + P_{\alpha_n}(2\ell) \tag{156}$$

$$= \left[0,\sum_{i=1}^{\ell}\frac{1}{\alpha_n^i}\Delta_i^{[2,1]},0,0,0,\sum_{i=1}^{\ell}\frac{1}{\alpha_n^i}\Delta_i^{[2,2]},0,0,0,0,\sum_{i=1}^{\ell}\frac{1}{\alpha_n^i}\Delta_i^{[3,3]},0\right]^T + P_{\alpha_n}(2\ell) \tag{157}$$

$$\times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \tag{176}$$

$$= -\sum_{\hat{y}:\sum_{i=1}^B \hat{y}_i=Pr} \prod_{i=1}^B \binom{P/B}{\hat{y}_i} \left(\prod_{i=1}^B \frac{1}{\binom{P/B}{\hat{y}_i}} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)\right)$$
$$\times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \tag{177}$$

$$= -\sum_{\hat{y}:\sum_{i=1}^B \hat{y}_i=Pr} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)$$
$$\times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \tag{178}$$

$$= -\sum_{\hat{x}:\sum_{i=1}^B \hat{x}_i=Pr} \sum_{x\in\hat{x}} P(X=x)$$
$$\times \sum_{x\in\hat{x}} \frac{P(X=x)}{\sum_{x\in\hat{x}} P(X=x)} \log \frac{P(X=x)}{\sum_{x\in\hat{x}} P(X=x)}, \tag{179}$$

where $\hat{y}=(\hat{y}_1,\dots,\hat{y}_B)$ and $\hat{x}=(\hat{x}_1,\dots,\hat{x}_B)$ are specific realizations of the numbers of permuted and real sparse subpackets in each of the B segments, respectively, and $x\in\hat{X}$ corresponds to each realization of X that result in $\hat{x}_1,\dots,\hat{x}_B$ numbers of sparse subpackets in the B segments. In general, the random variable $\hat{X}=(\hat{X}_1,\dots,\hat{X}_B)$ represents the numbers of (real) sparse subpackets updated by the user in each of the B segments, such that they sum up to Pr. Note that we do not assume any specific distribution of X in this calculation. Observing that $\sum_{x\in\hat{x}} P(X=x)=P(\hat{X}=\hat{x})$, the conditional entropy in (179) simplifies to,

$$H(X|Y)$$
$$= \sum_{\hat{x}:\sum_{i=1}^B \hat{x}_i=Pr} P(\hat{X}=\hat{x})\log P(\hat{X}=\hat{x})$$
$$- \sum_{\hat{x}:\sum_{i=1}^B \hat{x}_i=Pr} \sum_{x\in\hat{x}} P(X=x)\log P(X=x) \tag{180}$$
$$= -H(\hat{X})+H(X), \tag{181}$$

since all realizations of X satisfy $\sum_{i=1}^B \hat{x}_i=Pr$, based on the given uplink sparsification rate. Therefore,

$$I(X;Y)=H(X)-H(X|Y)=H(\hat{X})=H(\hat{X}_1,\dots,\hat{X}_B). \tag{182}$$

Cases 3 and 4: In cases 3 and 4, we consider permutations within segments as well as among segments to reduce the information leakage further. Recall that the permutations within the B segments are denoted by $\{\tilde{P}_i\}_{i=1}^B$ and the permutation among segments is denoted by $\hat{P}$. As explained in the schemes, none of these permutations are known by the databases. Similar to the above calculation, in order to calculate the information leakage $I(X;Y)$, we first compute the conditional distribution given by,

$$P(X=x|Y=y)$$
$$= \left(\sum_{\tilde{p}_1,\dots,\tilde{p}_B} \sum_{\hat{p}} P(Y=y|X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B)\right.$$
$$\left.\times P(X=x)P(\hat{P}=\hat{p})\prod_{i=1}^B P(\tilde{P}_i=\tilde{p}_i)\right)/P(Y=y) \tag{183}$$

$$= \frac{P(X=x)}{P(Y=y)}\frac{1}{B!}\frac{1}{(P/B)!^B}$$
$$\times \sum_{\tilde{p}_1,\dots,\tilde{p}_B} \sum_{\hat{p}} \mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B\}} \tag{184}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\frac{1}{B!}\frac{1}{(P/B)!^B} \\ \quad \times \prod_{i=1}^B \hat{y}_i!(\frac{P}{B}-\hat{y}_i)! \prod_{i=1}^B K_{\hat{y}_i}!, & \text{for } x,y:\{\hat{x}\}=\{\hat{y}\} \\ 0, & \text{otherwise} \end{cases} \tag{185}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\frac{1}{B!}\prod_{i=1}^B \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}!, & \text{for } x,y:\{\hat{x}\}=\{\hat{y}\} \\ 0, & \text{otherwise} \end{cases} \tag{186}$$

where $K_{\hat{y}_i}=\sum_{j=1}^B \mathbf{1}_{\hat{y}_j=\hat{y}_i}$, (i.e., number of segments with equal number of sparse subpackets as that of segment i) and the notation $\{\hat{x}\}=\{\hat{y}\}$ implies that the two sets $\hat{x}$ and $\hat{y}$ are the same, irrespective of their order, i.e., if $\{\hat{x}\}=\{\hat{y}\}$, for each $\hat{x}_i$ in $\hat{x}$, there exist some $\hat{y}_j$ in $\hat{y}$, such that $\hat{x}_i=\hat{y}_j$, and vice versa. Next we calculate $P(Y=y)$ for any y such that $\sum_{i=1}^{Pr}\hat{y}_i=Pr$ as,

$$P(Y=y)$$
$$= \sum_x \sum_{\tilde{p}_1,\dots,\tilde{p}_B} \sum_{\hat{p}} P(Y=y|X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B)$$
$$\times P(X=x)P(\hat{P}=\hat{p})\prod_{i=1}^B P(\tilde{P}_i=\tilde{p}_i) \tag{187}$$

$$= \sum_x P(X=x)\frac{1}{B!}\frac{1}{(P/B)!^B}$$
$$\times \sum_{\tilde{p}_1,\dots,\tilde{p}_B} \sum_{\hat{p}} \mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\dots,\tilde{P}_B=\tilde{p}_B\}} \tag{188}$$

$$= \frac{1}{B!}\prod_{i=1}^B \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}! \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \tag{189}$$

Therefore, from (186),

$$P(X=x|Y=y)$$
$$= \begin{cases} \frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)}, & \text{for } x,y:\{\hat{x}\}=\{\hat{y}\} \\ 0, & \text{otherwise.} \end{cases} \tag{190}$$

Then,

$$H(X|Y=y)$$
$$= -\sum_x P(X=x|Y=y)\log P(X=x|Y=y) \tag{191}$$
$$= -\sum_{x:\{\hat{x}\}=\{\hat{y}\}} \frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \tag{192}$$

$$
\begin{aligned}
&= \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\quad - \frac{1}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x),
\end{aligned} \tag{193}
$$

from which we obtain,

$$
\begin{aligned}
&H(X|Y) \\
&= \sum_{y} P(Y=y) H(X|Y=y) \qquad (194) \\
&= \sum_{y} \frac{1}{B!} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\quad - \sum_{y} \frac{1}{B!} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \frac{1}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)} \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x) \qquad (195) \\
&= \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \prod_{i=1}^{B} \binom{P/B}{\hat{y}_i} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\quad - \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \prod_{i=1}^{B} \binom{P/B}{\hat{y}_i} \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x) \qquad (196) \\
&= \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \prod_{i=1}^{B} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\quad - \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \prod_{i=1}^{B} K_{\hat{y}_i}! \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x) \\
&\qquad (197) \\
&= \frac{1}{B!} \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \frac{B!}{\prod_{i=1}^{B} K_{\hat{y}_i}!} \prod_{i=1}^{B} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\quad - \frac{1}{B!} \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \frac{B!}{\prod_{i=1}^{B} K_{\hat{y}_i}!} \prod_{i=1}^{B} K_{\hat{y}_i}! \\
&\quad \times \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x) \qquad (198) \\
&= \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\quad - \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x), \\
&\qquad (199)
\end{aligned}
$$

where $\tilde{y}$ introduced in (198) and $\tilde{x}$ are the realizations of corresponding random variables $\tilde{Y}$ and $\tilde{X}$ representing all distinct sets of $\hat{y}$ and $\hat{x}$, respectively. For example, if $B = 2$, $(1, 2)$ and $(2, 1)$ are considered to be two different realizations of $\hat{y}$ (or $\hat{x}$), while it is the same realization of $\tilde{y}$ (or $\tilde{x}$). Moreover,

$$
\begin{aligned}
P(\tilde{X}=\tilde{x}) &= \sum_{x\in\tilde{x}} P(X=x) \qquad (200) \\
&= \sum_{\text{all permutations of } x\in\hat{x}} P(X=x) \qquad (201)
\end{aligned}
$$

With this notation, the above entropy is further simplified to,

$$
\begin{aligned}
&H(X|Y) \\
&= \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\right) \\
&\qquad - \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x)\log P(X=x) \\
&\qquad (202) \\
&= \sum_{\tilde{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} P(\tilde{X}=\tilde{y})\log P(\tilde{X}=\tilde{y}) + H(X) \qquad (203) \\
&= -H(\tilde{X}) + H(X). \qquad (204)
\end{aligned}
$$

Therefore, the information leakage is given by,

$$
I(X;Y) = H(X) - H(X|Y) = H(\tilde{X}) = H(\tilde{X}_1, \ldots, \tilde{X}_B). \tag{205}
$$

Remark 8: An interesting observation is that the above calculations for the information leakage $I(X;Y)$ can be used to obtain closed form expressions for the MaxL information leakage [48] as well, i.e., to characterize $\text{MaxL}(X;Y) = \log_2 \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} P_{Y|X}(y|x)$, where $\mathcal{Y}$ and $\mathcal{X}$ are the sample spaces corresponding to the random variables Y and X, respectively. For example, the MaxL leakage (in contrast to the mutual information based leakage above) for cases 1 and 2 is given by,

$$
MaxL(X;Y) = \log_2 \left| (\hat{y}_1, \ldots, \hat{y}_B) : \sum_{i=1}^{B} \hat{y}_i = Pr \right|, \tag{206}
$$

where $\{(\hat{y}_1, \ldots, \hat{y}_B) : \sum_{i=1}^{B} \hat{y}_i = Pr\}$ denotes the set of all possible $(\hat{y}_1, \ldots, \hat{y}_B)$ with $0 \leq \hat{y}_i \leq \frac{P}{B}$ for all i satisfying $\sum_{i=1}^{B} \hat{y}_i = Pr$, with the same notation used in the above calculations. Therefore, the proposed scheme can also be used in a setting where the problem is defined based on the maximum information leakage metric, as opposed to the mutual information based metric in (2).

V. Conclusion

In this work, we considered the problem of private FL with top r sparsification. In FL with top r sparsification, the values and the positions of the sparse updates leak information about the user's private data. We proposed four schemes with different properties to perform FL with top r sparsification without revealing the values or the positions of the sparse updates to the databases. The schemes follow a permutation technique which requires a large storage cost. To this end, we generalized the schemes to incur a reduced storage cost at the expense of a certain amount of information leakage, using a model segmentation mechanism. In general, this work presents the tradeoff between the communication cost, storage complexity and information leakage in private FL with top r sparsification.

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," in *Proc. AISTATS*, Apr. 2017.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 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 *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 739–753.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[5] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 691–706.

[6] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *Proc. USENIX*, Apr. 2019.

[7] J. Geiping, H. Bauermeister, H. Droge, and M. Moeller, "Inverting gradients—How easy is it to break privacy in federated learning?" in *Proc. NeurIPS*, Dec. 2020.

[8] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. NeurIPS*, Dec. 2019.

[9] 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 *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 2512–2520.

[10] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017.

[11] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends® Theory Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2013.

[12] H. Ono and T. Takahashi, "Locally private distributed reinforcement learning," 2020, *arXiv:2001.11718*.

[13] Y. Li, T.-H. Chang, and C.-Y. Chi, "Secure federated averaging algorithm with differential privacy," in *Proc. IEEE 30th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2020, pp. 1–6.

[14] N. Agarwal, A. Suresh, F. Yu, S. Kumar, and H. B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Proc. NeurIPS*, Dec. 2018.

[15] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences," in *Proc. NeurIPS*, Dec. 2018.

[16] M. A. Heikkilä, A. Koskela, K. Shimizu, S. Kaski, and A. Honkela, "Differentially private cross-silo federated learning," 2020, *arXiv:2007.05553*.

[17] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. ICLR*, May 2018.

[18] S. Asoodeh, W.-N. Chen, F. P. Calmon, and A. Özgür, "Differentially private federated learning: An information-theoretic perspective," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 344–349.

[19] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proc. ACM-SIAM Symp. Discrete Algorithm*, Jan. 2019.

[20] B. Balle, J. Bell, A. Gascon, and K. Nissim, "The privacy blanket of the shuffle model," in *Proc. CRYPTO*, Aug. 2019.

[21] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. Suresh, "Shuffled model of differential privacy in federated learning," in *Proc. AISTAT*, Apr. 2021.

[22] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," in *Proc. NeurIPS*, Dec. 2017.

[23] S. Ulukus, S. Avestimehr, M. Gastpar, S. A. Jafar, R. Tandon, and C. Tian, "Private retrieval, computing, and learning: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 729–748, Mar. 2022.

[24] C. Naim, R. G. L. D'Oliveira, and S. E. Rouayheb, "Private multi-group aggregation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1391–1396.

[25] D. Basu, D. Data, C. Karakus, and S. N. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 217–226, May 2020.

[26] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. AISTATS*, Aug. 2020.

[27] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. NeurIPS*, Dec. 2017.

[28] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Federated learning with quantization constraints," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8851–8855.

[29] C. Niu et al., "Billion-scale federated learning on mobile clients: A submodel design with tunable privacy," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2020.

[30] M. Kim and J. Lee, "Information-theoretic privacy in federated submodel learning," *ICT Exp.*, vol. 9, no. 3, pp. 415–419, Jun. 2023.

[31] C. Niu et al., "Secure federated submodel learning," 2019, *arXiv:1911.02254*.

[32] Z. Jia and S. A. Jafar, "X-secure T-private federated submodel learning with elastic dropout resilience," *IEEE Trans. Inf. Theory*, vol. 68, no. 8, pp. 5418–5439, Aug. 2022.

[33] S. Vithana and S. Ulukus, "Efficient private federated submodel learning," in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 3394–3399.

[34] Z. Jia and S. A. Jafar, "X-secure T-private federated submodel learning," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[35] S. Vithana and S. Ulukus, "Private read update write (PRUW) with storage constrained databases," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 2391–2396.

[36] S. Vithana and S. Ulukus, "Private federated submodel learning with sparsification," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2022, pp. 410–415.

[37] S. Vithana and S. Ulukus, "Rate distortion tradeoff in private read update write in federated submodel learning," in *Proc. 56th Asilomar Conf. Signals, Syst., Comput.*, Oct. 2022, pp. 210–214.

[38] J. Wangni et al., "Gradient sparsification for communication-efficient distributed optimization," in *Proc. NeurIPS*, Dec. 2018.

[39] S. Li, Q. Qi, J. Wang, H. Sun, Y. Li, and F. R. Yu, "GGS: General gradient sparsification for federated learning in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020.

[40] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 300–310.

[41] S. Shi, K. Zhao, Q. Wang, Z. Tang, and X. Chu, "A convergence analysis of distributed SGD with communication-efficient gradient sparsification," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019.

[42] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. NeurIPS*, Dec. 2018.

[43] Y. Sun, S. Zhou, Z. Niu, and D. Gündüz, "Time-correlated sparsification for efficient over-the-air model aggregation in wireless federated learning," in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 3388–3393.

[44] L. P. Barnes, H. A. Inan, B. Isik, and A. Özgür, "RTop-*k*: A statistical estimation approach to distributed SGD," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 3, pp. 897–907, Nov. 2020.

[45] E. Ozfatura, K. Ozfatura, and D. Gündüz, "Time-correlated sparsification for communication-efficient federated learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 461–466.

[46] Z. Jia, H. Sun, and S. A. Jafar, "Cross subspace alignment and the asymptotic capacity of X-secure T-private information retrieval," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5783–5798, Sep. 2019.

[47] S. Vithana and S. Ulukus, "Private read update write (PRUW) in federated submodel learning (FSL): Communication efficient schemes with and without sparsification," *IEEE Trans. Inf. Theory*, 2023.

[48] H.-Y. Lin, S. Kumar, E. Rosnes, A. G. I Amat, and E. Yaakobi, "Multi-server weakly-private information retrieval," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 1197–1219, Feb. 2022.

Sajani Vithana (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the University of Peradeniya, Sri Lanka, in 2017. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA. Her research interests include information theory, private information retrieval, distributed coded computing, and federated learning.

Sennur Ulukus (Fellow, IEEE) received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University and the Ph.D. degree in electrical and computer engineering from the Wireless Information Network Laboratory (WINLAB), Rutgers University.

She is currently a Distinguished University Professor and the Anthony Ephremides Professor in information sciences and systems with the Department of Electrical and Computer Engineering, University of Maryland (UMD), College Park, MD, USA, where she also holds a joint appointment with the Institute for Systems Research (ISR). Prior to joining UMD, she was a Senior Technical Staff Member at AT&T Labs-Research. Her research interests include information theory, wireless communications, machine learning, signal processing, and networks; with recent focus on private information retrieval, age of information, machine learning for wireless, distributed coded computing, group testing, physical layer security, energy harvesting communications, and wireless energy and information transfer.

Dr. Ulukus received the 2003 IEEE Marconi Prize Paper Award in Wireless Communications, the 2019 IEEE Communications Society Best Tutorial Paper Award, the 2020 IEEE Communications Society Women in Communications Engineering (WICE) Outstanding Achievement Award, the 2020 IEEE Communications Society Technical Committee on Green Communications and Computing (TCGCC) Distinguished Technical Achievement Recognition Award, the 2005 NSF CAREER Award, the 2011 ISR Outstanding Systems Engineering Faculty Award, and the 2012 ECE George Corcoran Outstanding Teaching Award. She is the TPC Chair of the 2021 IEEE Globecom, the 2024 IEEE Globecom, the 2024 IEEE DySPAN, the 2023 IEEE MILCOM, the 2019 IEEE ITW, the 2017 IEEE ISIT, the 2016 IEEE Globecom, the 2014 IEEE PIMRC, and the 2011 IEEE CTW. She has been a Senior Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING since 2020. She was an Area Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2019 to 2023 and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING from 2016 to 2020, an Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS-Series on Green Communications and Networking from 2015 to 2016, an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY from 2007 to 2010, and an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS from 2003 to 2007. She was a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN INFORMATION THEORY in 2021, 2022, and 2023, the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS in 2008, 2015, 2021, and 2022, the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING in 2021, *Journal of Communications and Networks* in 2012, and the IEEE TRANSACTIONS ON INFORMATION THEORY in 2011. She was a Distinguished Lecturer of the IEEE Information Theory Society from 2018 to 2019. She is a Distinguished Scholar-Teacher of the University of Maryland.