Efficient Private Federated Submodel Learning

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

***Abstract*—We investigate the problem of private federated submodel learning, where a machine learning model is divided into M submodels and stored in N databases, from which a given user *privately* reads, updates and writes back an arbitrary submodel. We consider information-theoretic privacy of the updated submodel index as well as the values of the updates. We provide an efficient *private read update write* (PRUW) scheme which achieves a lower total communication cost compared to the state-of-the-art. Our scheme significantly reduces the writing cost by combining all updates into a single bit in a way that it can be privately decomposed and placed at the relevant positions at the databases. This is achieved by over-designing the system with additional random noise terms in storage, which in turn provides additional security to the submodels. The scheme is designed for arbitrary privacy and security requirements.**

I. Introduction

Processing power limitation and privacy issues that arise from the increased use of machine learning techniques in various applications is solved to a certain extent by the introduction of federated learning (FL) paradigm [1]–[4]. However, FL requires local users to download and update the entire machine learning model, which is inefficient as the local users may not have a sufficient amount of data to train the entire model. This motivates the concept of federated submodel learning (FSL) [5]–[7], where the machine learning model is divided into a number of submodels, out of which each local user downloads and updates only the submodel that can be trained using its local data. As such, FSL requires additional privacy protocols since the values of the updates as well as the indices of the submodels updated leak information on the data generated by the users. In private FSL, each user needs to privately read, update and write back an arbitrary submodel of interest. We refer to this as private read update write (PRUW). PRUW has two main phases, namely, the reading phase and the writing phase. The reading phase requires each user to read the relevant submodel without revealing its index. This is equivalent to the problem of private information retrieval (PIR), see e.g., [8]–[14]. The writing phase requires the user to write the updates to the relevant submodel without revealing the submodel index or the values of the updates.

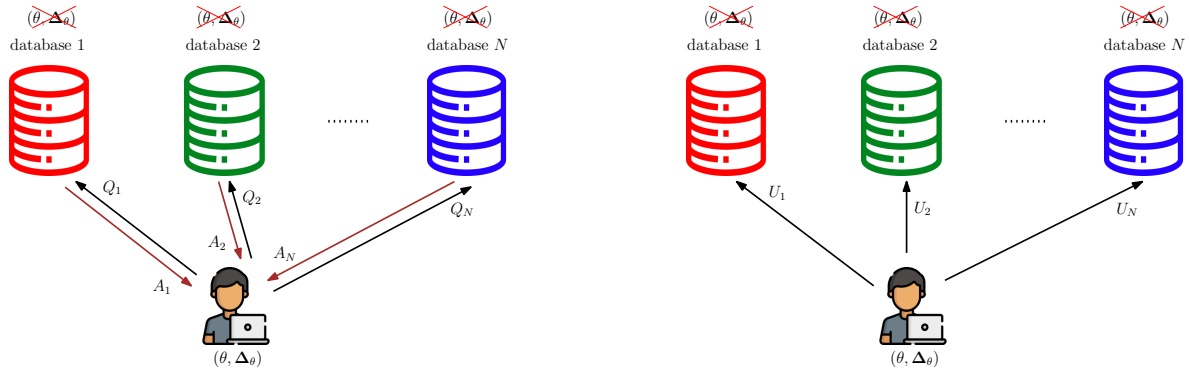
References [5]–[7], [15] use different notions of privacy to propose schemes for private FSL. [5] considers locally differential privacy in which a predetermined amount of information of the user is leaked to the databases. [6] considers information-theoretic privacy between the user and the databases. However, this scheme leaks information of the user at time t to the user at time $t+1$. [15] presents a group-wise aggregation scheme (private writing), based on local differential privacy. [7] considers information-theoretic privacy between the user and the databases, as well as among the users at different time instances. The process of PRUW is carried out in [7] by using a special PIR scheme [16] in the reading phase. The queries of this scheme are made use of again in the writing phase to privately write to the databases.

We propose a scheme that privately reads and writes to any given submodel which incurs a lower total communication cost (reading + writing cost) compared to the existing state-of-the-art in private FSL. The proposed scheme considers information-theoretic privacy of the index of the submodel updated by each user as well as the values of the updates. Both elements are kept private from the databases and other users at different time instances. In the proposed scheme, information-theoretic privacy of the submodel index and the updates is achieved by adding random noise terms to storage, queries and updates. The numbers of random noise terms added to each quantity above are calculated based on the given privacy and security requirements, with the objective of minimizing the total cost. This leads to the scheme being over-designed with extra noise terms to minimize the total cost while satisfying the required privacy and security constraints.

The proposed scheme is similar to the scheme in [7] in the reading phase. However, in the writing phase, the proposed scheme combines the updates of all parameters in a subpacket of the submodel and sends only a single bit to each database, as opposed to sending an update vector. This significantly reduces the writing cost. The updates are combined in such a way that the combined bit is in the form of a polynomial that contains random noise terms. Once the update bit is received by each of the databases, it is privately decomposed into the bit-wise incremental updates and added to the corresponding parameters in storage. This calculation introduces a few extra terms, which requires the storage to have multiple noise terms in order to guarantee privacy of the writing process. The large number of noise terms in storage increases the reading cost. However, the cost saved in the writing phase is much larger than the additional cost incurred in the reading phase resulting in a lower total cost compared to what is incurred without combining the bit-wise updates in the writing phase.

The main contributions of our paper are as follows: 1) We provide a PRUW scheme for FSL, which is an improved version of the scheme presented in [7] that results in a lower

This work was supported by ARO Grant W911NF2010142, and NSF Grants CCF 17-13977 and ECCS 18-07348.



(a) Reading phase

(b) Writing phase

Fig. 1: A user reads (downloads) a required submodel, updates it, and writes (uploads) it back to the databases.

writing cost. 2) The proposed scheme incurs the lowest total cost so far in the literature while guaranteeing information-theoretic privacy between the user and the databases as well as between different users at all time instances. 3) The proposed scheme introduces the concept of combining bit-wise updates into a single bit in such a way that the databases can privately decompose the combined update bit into multiple updates and place them at the relevant positions. 4) The proposed scheme introduced the idea of over-designing a scheme (with extra noise terms) in order to make it more cost efficient; this over-design increases the read cost but decreases the write cost much more, yielding a net reduction in the total cost.

II. Problem Formulation

Consider N databases storing M independent submodels, i.e., for all distinct $i, j \in \{1, \dots, M\}$, $I(W_i; W_j) = 0$, where W_i and W_j are the ith and jth submodels, respectively. The submodels consist of random symbols of the field $\mathbf{F}_q$. At any given time instance, a single user privately updates a single submodel of interest. The process of updating consists of two phases, namely, the reading phase where the user downloads the required submodel and the writing phase where the user uploads the incremental updates back to the databases. Any information that is communicated in both phases takes place only between a single user and the system of databases as illustrated in Fig. 1. The same process takes place independently and privately at each time instance with different users.

The user at time t has no prior information on the updating submodel, i.e., at any time instance $t \in \mathcal{Z}^+$, $I(\mathbf{Q}_n^{[t]}; W_\theta^{[t-1]}) = 0$ for all databases $n \in \{1, \dots, N\}$, where $W_\theta^{[t-1]}$ is the existing version of the required submodel and $\mathbf{Q}_n^{[t]}$ is the query sent by the user to database n. The following privacy and security constraints need to be satisfied by any given PRUW scheme in order to ensure information theoretic privacy of the user required submodel index and the values of the updates.

Privacy and security requirements (PS requirements): The PS requirements state the maximum numbers of databases that are allowed to collude and still obtain no information on the user required submodel index, values of the updates and the values of the submodel parameters. The formal definitions of privacy and security requirements are given next.

T_r-privacy of submodel index: T_r-privacy ensures that no information on the index of the submodel updated by the user is leaked to any of the databases, when upto any T_r of them are allowed to collude. I.e., for any subset of databases $\Gamma \subset \{1, \dots, N\}$ satsfying $|\Gamma| = T_r$,

$$I(\theta^{[t]}; \mathbf{Q}_\Gamma^{[t]}, \mathbf{U}_\Gamma^{[t]} | \mathbf{Q}_\Gamma^{[1:t-1]}, \mathbf{S}_\Gamma^{[1:t-1]}, \mathbf{U}_\Gamma^{[1:t-1]}) = 0, \quad (1)$$

where $\theta^{[t]}$ is the index of the submodel updated by the user at time t. $\mathbf{Q}_\Gamma$, $\mathbf{U}_\Gamma$ and $\mathbf{S}_\Gamma$ are the queries, updates and storage of databases in Γ at the corresponding time instances.

Y_r-privacy of the values of the updates: Y_r-privacy ensures that no information on the values of the updates $\mathbf{\Delta}_\theta^{[t]}$ generated by the user is leaked to any of the databases, when upto any Y_r of them are allowed to collude. I.e., for any subset of databases $\Gamma \subset \{1, \dots, N\}$ satsfying $|\Gamma| = Y_r$,

$$I(\mathbf{\Delta}_\theta^{[t]}; \mathbf{U}_\Gamma^{[t]} | \mathbf{Q}_\Gamma^{[1:t]}, \mathbf{S}_\Gamma^{[1:t-1]}, \mathbf{U}_\Gamma^{[1:t-1]}) = 0. \quad (2)$$

X_r-security of storage: X_r-security ensures that no information on the submodels is leaked to any of the databases, when upto any X_r of them are allowed to collude. I.e., for each submodel $k \in \{1, \dots, M\}$, and for any subset of databases $\Gamma \subset \{1, \dots, N\}$ satsfying $|\Gamma| = X_r$,

$$I(\mathbf{W}_k^{[t]}; \mathbf{S}_\Gamma^{[t]}) = 0. \quad (3)$$

The triple (T_r, Y_r, X_r) defines the PS-requirements of a given PRUW system. Next, we have the following correctness conditions that need to be satisfied by any PRUW scheme.

Correctness in reading phase: In the reading phase, the user must be able to correctly decode the required submodel using the queries sent and the answers received from all databases I.e., $H(\mathbf{W}_\theta^{[t-1]} | \mathbf{Q}_{[1:N]}^{[t]}, \mathbf{A}_{[1:N]}^{[t]}) = 0$, where $\mathbf{W}_\theta^{[t-1]}$ is the submodel (before updating) required by the user at time t and $A_n^{[t]}$ is the answer received from database n at time t.

Correctness in writing phase: At time t, each parameter in $\mathbf{W}_{\theta_t}$ stored in each database must be correctly updated as, $\mathbf{W}_{\theta_t}^{[t]} = \mathbf{W}_{\theta_t}^{[t-1]} + \mathbf{\Delta}_{\theta_t}$.

In this work, we provide an efficient PRUW scheme where a given user at a given time instance reads, updates and writes to a specific submodel while satisfying the above privacy, security (PS) and correctness conditions. The reading cost is

defined as $C_R = \frac{\mathcal{D}_L}{L}$, where $\mathcal{D}_L$ is the total number of bits downloaded from all databases when retrieving the required submodel and L is the size of the submodel. The writing cost is defined as $C_W = \frac{\mathcal{U}_L}{L}$, where $\mathcal{U}_L$ is the total number of bits sent to all databases in the writing phase. The total cost is defined as $C_T = C_R + C_W$.

III. Main Result

Theorem 1 *Following reading, writing and total costs are achievable in a PRUW system with N databases containing M submodels with PS-requirements given by (T_r, Y_r, X_r), whenever $N \geq \max\{X_r + T_r + 1,\ 2T_r + Y_r + 1\}$,*

$$(C_R, C_W) = \left(\frac{N}{N - T_r - X_u^*}, \frac{2N - 2X_u^* + Y_r - 1}{N - T_r - X_u^*} \right) \quad (4)$$

$$C_T = \frac{3N - 2X_u^* + Y_r - 1}{N - T_r - X_u^*}, \quad (5)$$

where $X_u^* = \max\{X_r, \lceil \frac{N+Y_r-1}{2} \rceil\}$.

The proof of Theorem 1 is presented in Section V.

Remark 1 *The proposed scheme achieves reading, writing and total costs that decrease with increasing number of databases for the general setting stated in Theorem 1. This is unique to this scheme as the existing PRUW scheme in [7] achieves a writing (hence total) cost that is increasing in N.*

Remark 2 *In the proposed scheme, the given (T_r, Y_r, X_r) PS-requirement is satisfied by adding (T_u, Y_u, X_u) random noise terms to queries, updates and storage, respectively (Shannon's one-time-pad). The optimum values of (T_u, Y_u, X_u) satisfying the given PS-requirement, on which the costs in Theorem 1 are based are $T_u^* = T_r$, $Y_u^* = Y_r$ and $X_u^* = \max\{X_r, \lceil \frac{N+Y_r-1}{2} \rceil\}$.*

Remark 3 *For a PRUW setting with non-colluding databases (i.e., $T_r = Y_r = X_r = 1$), the optimum values of (T_u, Y_u, X_u) in the proposed scheme are $T_u^* = Y_u^* = 1$ and $X_u = \lceil \frac{N}{2} \rceil$. For comparison, the scheme in [7] achieves the lowest total cost for the same setting when $T_u^* = Y_u^* = 1$ and $X_u^* \approx N - \sqrt{N} - 1$ for $N \geq 6$ and $T_u^* = Y_u^* = 1$ and $X_u^* = 2$ otherwise. The variation of the total cost of the two schemes with different number of databases is shown in Fig. 2.*

Remark 4 *For a given N and given storage (specified by the numbers of noise terms added to queries, updates and storage given by (T_u, Y_u, X_u)), the proposed scheme always performs equally or better than the scheme in [7] in terms of the total cost. To see this, consider the set of values of (N, T_u, Y_u, X_u) for which the scheme in [7] outperforms the proposed scheme given by*[1],

$$\frac{N}{N - X_u - T_u} + N - X_u + Y_u + T_u < \frac{3N - 2X_u + Y_u - 1}{N - X_u - T_u}, \quad (6)$$

[1]The expressions on the left and right are the total costs of the scheme in [7] and the proposed scheme (given in (31)) for a given storage.

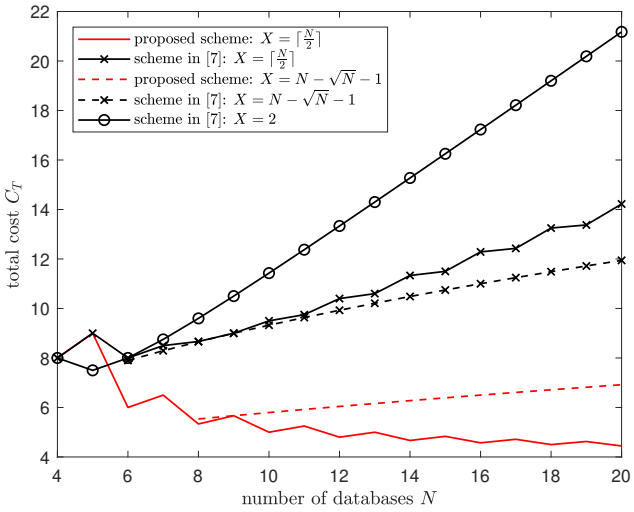


Fig. 2: Comparison of the total costs of the two schemes.

which simplifies to,

$$(N - X_u + Y_u) < 1 - T_u, \quad \text{if } N - X_u - T_u \neq 1. \quad (7)$$

Note that (7) is not satisfied by any (N, T_u, Y_u, X_u) as the right side of (7) is ≤ 0 and the left side is ≥ 2 due to the privacy and recoverability constraints. Therefore, the proposed scheme outperforms the scheme in [7] for any given setting except when $N - X_u - T_u = 1$. They both perform the same when $N - X_u - T_u = 1$.

Remark 5 *For a given PRUW setting (given N and given PS-requirement) let the optimum values of (T_u, Y_u, X_u) that minimize the total cost of the proposed scheme and the scheme in [7] be (T_r, Y_r, X_u^*) and $(T_r, Y_r, \bar{X}_u^*)$, respectively*[2]*. Then, if $C_T(\cdot)$ and $\bar{C}_T(\cdot)$ denote the total costs of the proposed scheme and the scheme in [7], respectively,*

$$C_T(T_r, Y_r, X_u^*) \leq C_T(T_r, Y_r, \bar{X}_u^*) \leq \bar{C}_T(T_r, Y_r, \bar{X}_u^*), \quad (8)$$

where the first inequality is due to the optimality of X_u^* and the second follows from Remark 4. This proves that the proposed scheme always performs equally or better than the scheme in [7], for any given PRUW setting specified by arbitrary PS-requirements and arbitrary N (even without the same storage).

IV. Representative Example

In this section, we provide a basic example in which the proposed scheme is applied to a setting where $N = 6$ and $M = 2$. This example describes the steps of the proposed scheme with explicit expressions for downloads, uploads and costs. It also shows the reduction in the total cost of the proposed scheme compared to [7]. In this example, assume that the user wants to read and write to the first submodel

[2]Both schemes have total costs that increase with increasing T_u and Y_u. Therefore, $T_u^* = T_r$ and $Y_u^* = Y_r$ in both schemes.

($\theta = 1$) and let the PS-requirement be such that $T_r = Y_r = X_r = 1$. The PS-requirements are satisfied by adding X_u, T_u and Y_u random noise terms to storage, queries and updates, respectively. Any $X_u \geq 1$, $T_u \geq 1$ and $Y_u \geq 1$ satisfies the given PS-requirement. Therefore, let $X_u = 3$, $T_u = 1$ and $Y_u = 1$. The subpacketization[3] of the scheme is $\ell = 2$ and the storage of a single subpacket of all submodels in a database is given next.

Storage: Each individual subpacket of all submodels are stored in database n, $n \in \{1, \ldots, 6\}$ as,

$$\mathbf{S}_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n) \sum_{i=0}^{2} \alpha_n^i Z_{1,i}^{[1]} \\ W_{2,1} + (f_1 - \alpha_n) \sum_{i=0}^{2} \alpha_n^i Z_{2,i}^{[1]} \end{bmatrix} \\ \begin{bmatrix} W_{1,2} + (f_2 - \alpha_n) \sum_{i=0}^{2} \alpha_n^i Z_{1,i}^{[2]} \\ W_{2,2} + (f_2 - \alpha_n) \sum_{i=0}^{2} \alpha_n^i Z_{2,i}^{[2]} \end{bmatrix} \end{bmatrix}, \tag{9}$$

where $W_{i,j}$ is the jth bit of submodel i, W_i, $Z_{i,j}^{[k]}$ is the $(j+1)$st noise term for the kth bit of W_i, and f_is and α_ns are globally known distinct constants chosen from $\mathbf{F}_q$, such that each α_n and $f_i - \alpha_n$ are coprime with q.

Reading phase: The query sent by the user to database n is given by,

$$\mathbf{Q}_n = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \tilde{\mathbf{Z}}_1 \\ \frac{1}{f_2 - \alpha_n} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \tilde{\mathbf{Z}}_2 \end{bmatrix}, \quad n \in \{1, \ldots, 6\}, \tag{10}$$

where $\tilde{\mathbf{Z}}_i$s are random noise vectors. Then, each database n, $n \in \{1, \ldots, 6\}$ sends the answer given by,

$$A_n = \mathbf{S}_n^T \mathbf{Q}_n = \frac{W_{1,1}}{f_1 - \alpha_n} + \frac{W_{1,2}}{f_2 - \alpha_n} + \sum_{i=0}^{3} \alpha_n^i R_i, \tag{11}$$

where R_is are combinations of noise terms that do not depend on n. The user retrieves the first submodel using the answers of all databases given by,

$$\begin{bmatrix} A_1 \\ \vdots \\ A_6 \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1 - \alpha_1} & \frac{1}{f_2 - \alpha_1} & 1 & \alpha_1 & \alpha_1^2 & \alpha_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1 - \alpha_6} & \frac{1}{f_2 - \alpha_6} & 1 & \alpha_6 & \alpha_6^2 & \alpha_6^3 \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{R} \end{bmatrix}, \tag{12}$$

where $\mathbf{R} = [R_0, \ldots, R_3]$. The reading cost is $C_R = \frac{N}{\ell} = 3$. With the same storage and queries, the reading phase is similar to that of the scheme in [7].

Writing phase: Let the update vector of W_1 be $\mathbf{\Delta}_1 = (\Delta_{1,1}, \Delta_{1,2})$. Then, the user sends the following combined single bit update to database n,

$$U_n = \left(\frac{f_2 - \alpha_n}{f_2 - f_1} \right) \Delta_{1,1} + \left(\frac{f_1 - \alpha_n}{f_1 - f_2} \right) \Delta_{1,2} + (f_1 - \alpha_n)(f_2 - \alpha_n) z, \tag{13}$$

where z is a random noise bit. Once database n receives U_n in (13), the incremental update is calculated by $\bar{U}_n = D_n \times U_n \times \mathbf{Q}_n$, where D_n is the constant (for database n) matrix given by $D_n = \text{diag}((f_1 - \alpha_n) I_2, (f_2 - \alpha_n) I_2)$ where I_2 is the identity matrix of size 2×2. Then, the incremental update in each database $n \in \{1, \ldots, 6\}$ is given by,

$$\bar{U}_n = D_n \times \begin{bmatrix} \frac{U_n}{f_1 - \alpha_n} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + U_n \tilde{\mathbf{Z}}_1 \\ \frac{U_n}{f_2 - \alpha_n} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + U_n \tilde{\mathbf{Z}}_2 \end{bmatrix}. \tag{14}$$

Note that $\frac{U_n}{f_1 - \alpha_n}$ is simplified to,

$$\frac{U_n}{f_1 - \alpha_n} = \frac{\Delta_{1,1}}{f_2 - f_1} \left(\frac{f_2 - \alpha_n}{f_1 - \alpha_n} \right) + \frac{\Delta_{1,2}}{f_1 - f_2} + (f_2 - \alpha_n) z \tag{15}$$

$$= \frac{\Delta_{1,1}}{f_2 - f_1} \left(\frac{f_2 - f_1 + f_1 - \alpha_n}{f_1 - \alpha_n} \right) + \frac{\Delta_{1,2}}{f_1 - f_2} + (f_2 - \alpha_n) z \tag{16}$$

$$= \frac{\Delta_{1,1}}{f_1 - \alpha_n} + s_0 + s_1 \alpha_n, \tag{17}$$

where s_0 and s_1 are constants that do not depend on n. Similarly, $\frac{U_n}{f_2 - \alpha_n} = \frac{\Delta_{1,2}}{f_2 - \alpha_n} + r_0 + r_1 \alpha_n$ for some constants r_0 and r_1. Therefore, for each database n, $n \in \{1, \ldots, 6\}$,

$$\bar{U}_n = \begin{bmatrix} \Delta_{1,1} + (f_1 - \alpha_n)(\hat{Z}_{1,0}^{[1]} + \hat{Z}_{1,2}^{[1]} \alpha_n + \hat{Z}_{1,2}^{[1]} \alpha_n^2) \\ (f_1 - \alpha_n)(\hat{Z}_{2,0}^{[1]} + \hat{Z}_{2,2}^{[1]} \alpha_n + \hat{Z}_{2,2}^{[1]} \alpha_n^2) \\ \Delta_{1,2} + (f_2 - \alpha_n)(\hat{Z}_{1,0}^{[2]} + \hat{Z}_{1,2}^{[2]} \alpha_n + \hat{Z}_{1,2}^{[2]} \alpha_n^2) \\ (f_2 - \alpha_n)(\hat{Z}_{2,0}^{[2]} + \hat{Z}_{2,2}^{[2]} \alpha_n + \hat{Z}_{2,2}^{[2]} \alpha_n^2) \end{bmatrix}, \tag{18}$$

where $\hat{Z}_{i,j}^{[k]}$ are combinations of noise terms that are fixed for all n. Note that (18) is of the same form as the storage. Therefore, the storage is updated by $S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n$ for each $n \in \{1, \ldots, 6\}$ and the writing cost is $C_W = \frac{N}{\ell} = 3$. Therefore, the total cost is $C_T = C_R + C_W = 6$. The writing cost of the scheme in [7] is $N - X_u + T_u + Y_u = 5$ and the resulting total cost is 8.

V. Proof of Theorem 1

A. Proposed Scheme

For a given PRUW system with a specific PS-requirement (T_r, Y_r, X_r), the proposed scheme can be applied to any system of $N \geq \max\{X_r + T_r + 1,\ 2T_r + Y_r + 1\}$ databases. In the proposed scheme, the given PS-requirement (T_r, Y_r, X_r) is satisfied by adding $T_u \geq T_r$, $Y_u \geq Y_r$ and $X_u \geq X_r$ independent random noise terms (within the field $\mathbf{F}_q$) to queries, updates and storage, respectively (Shannon's one-time-pad). The proposed scheme provides the optimal values of (T_u, Y_u, X_u) satisfying $T_u \geq T_r$, $Y_u \geq Y_r$ and $X_u \geq X_r$, that minimize the total cost. In other words, the proposed scheme is over-designed with extra noise terms to make the PRUW process more efficient.

We now present the proposed scheme with arbitrary (T_u, Y_u, X_u) satisfying $T_u \geq T_r$, $Y_u \geq Y_r$ and $X_u \geq X_r$. The optimum values of (T_u, Y_u, X_u) are derived towards the end of this section. Let ℓ be the subpacketization of the proposed

[3]The calculation of subpacketization and the optimal choice of (T_u, X_u, Y_u) for a given PS-requirement is explained in the description of the scheme in Section V.

scheme given by, $\ell = N - X_u - T_u$. An additional constraint given by $\frac{N+Y_u-1}{2} \leq X_u \leq N - T_u - 1$ must be satisfied by (T_u, Y_u, X_u) for a given N with given PS-requirements[4]. The storage of a single subpacket of all submodels (before updating) in database n, $n \in \{1, \ldots, N\}$ is given by,

$$\mathbf{S}_n = \begin{bmatrix} \begin{bmatrix} W_{1,1} + (f_1 - \alpha_n) \sum_{i=0}^{X_u-1} \alpha_n^i Z_{1,i}^{[1]} \\ \vdots \\ W_{M,1} + (f_1 - \alpha_n) \sum_{i=0}^{X_u-1} \alpha_n^i Z_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} W_{1,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{X_u-1} \alpha_n^i Z_{1,i}^{[\ell]} \\ \vdots \\ W_{M,\ell} + (f_\ell - \alpha_n) \sum_{i=0}^{X_u-1} \alpha_n^i Z_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix}. \quad (19)$$

Reading and writing to ℓ bits of the required submodel is explained in the rest of this section. The same procedure is followed $\frac{L}{\ell}$ times for the entire PRUW process, where L is the total length of each submodel.

1) Reading Phase: The reading phase is similar to that of the scheme in [7]. Assume that the user requires to update W_θ. Then, the user sends the following query to database n,

$$\mathbf{Q}_n = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} \mathbf{e}_M(\theta) + \sum_{i=0}^{T_u-1} \alpha_n^i \tilde{\mathbf{Z}}_{1,i} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} \mathbf{e}_M(\theta) + \sum_{i=0}^{T_u-1} \alpha_n^i \tilde{\mathbf{Z}}_{\ell,i} \end{bmatrix}, \quad (20)$$

for each $n \in \{1, \ldots, N\}$, where $\mathbf{e}_M(\theta)$ is the all zeros vector of size $M \times 1$ with a one at the θth position and $\tilde{\mathbf{Z}}_{i,j}$s are random noise vectors of size $M \times 1$. Database n then generates the answer given by,

$$A_n = \mathbf{S}_n^T \mathbf{Q}_n = \frac{W_{\theta,1}}{f_1 - \alpha_n} + \ldots + \frac{W_{\theta,\ell}}{f_\ell - \alpha_n} + \sum_{i=0}^{X_u+T_u-1} \alpha_n^i R_i, \quad (21)$$

where R_is are combinations of noise terms that do not depend on n. The answers received by the N databases in matrix form is as follows,

$$\begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \cdots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \cdots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \ldots & \alpha_N^p \end{bmatrix} \begin{bmatrix} \mathbf{W}_\theta \\ \mathbf{R} \end{bmatrix}, \quad (22)$$

where $p = X_u + T_u - 1$ and $\mathbf{R} = [R_0, \ldots, R_p]$. Since the matrix is invertible, the ℓ bits of $\mathbf{W}_\theta$ can be retrieved using (22). The reading cost is given by,

$$C_R = \frac{N}{\ell} = \frac{N}{N - X_u - T_u}. \quad (23)$$

2) Writing Phase: In the writing phase, the user sends a single bit to each database, which is a combination of the updates of the ℓ bits of $\mathbf{W}_\theta$ and Y_u random noise bits. The combined update bit is a polynomial of α_n, which allows the databases to privately decompose it into the ℓ individual update bits, with the help of the queries received in the reading phase. Finally, these incremental updates are added to the existing storage to obtain the updated storage. As explained later in this section, the above stated decomposition performed at the databases introduces a few extra terms, which are added to the X_u random noise terms in storage. The reason behind over-designing the system to have extra noise terms in storage is to have a number of noise terms that matches the number of extra terms introduced by the *decomposition* performed at the databases in the writing phase. The combined single update bit that the user sends to database n is given by,

$$U_n = \sum_{i=1}^{\ell} \tilde{\Delta}_{\theta,i} \prod_{j=1, j\neq i}^{\ell} (f_j - \alpha_n) + \prod_{j=1}^{\ell} (f_j - \alpha_n)(Z_0 + \alpha_n Z_1 + \ldots + \alpha_n^{Y_u-1} Z_{Y_u-1}), \quad (24)$$

where Z_is are random noise bits, $\tilde{\Delta}_{\theta,i} = \frac{\Delta_{\theta,i}}{\prod_{j=1,j\neq i}^{\ell}(f_j - f_i)}$ with $\Delta_{\theta,i}$ being the update of the ith bit of $\mathbf{W}_\theta$. Once database n receives U_n, it calculates the incremental update that needs to be added to the existing storage in order to obtain the new and updated storage. This calculation requires the following two definitions and lemmas.

Definition 1 *(Scaling matrix) For each database $n \in \{1, \ldots, N\}$, the scaling matrix is defined as $D_n = diag((f_1 - \alpha_n)\mathbf{1}_M, \ldots, (f_\ell - \alpha_n)\mathbf{1}_M)$, where $\mathbf{1}_M$ is a vector of M ones.*

Definition 2 *(Null shaper[5]) For each database $n \in \{1, \ldots, N\}$, the null shaper is defined as, $\Omega_n = diag\left(\left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_1)}\right)\mathbf{1}_M, \ldots, \left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_\ell)}\right)\mathbf{1}_M\right)$, where $\mathcal{F}$ is any subset of databases satisfying $|\mathcal{F}| = 2X_u - N - Y_u + 1$.*

Lemma 1 *For each $k \in \{1, \ldots, \ell\}$,*

$$\frac{U_n}{f_k - \alpha_n} = \frac{1}{f_k - \alpha_n} \Delta_{\theta,k} + P_{\alpha_n}(\ell + Y_u - 2), \quad (25)$$

where $P_{\alpha_n}(\ell + Y_u - 2)$ is a polynomial of α_n of degree $\ell + Y_u - 2$. The coefficients of α_n^is in $P_{\alpha_n}(\ell + Y_u - 2)$ are fixed for all n.

Lemma 2 *For each $k \in \{1, \ldots, \ell\}$,*

$$\left(\frac{\prod_{r\in\mathcal{F}}(\alpha_r - \alpha_n)}{\prod_{r\in\mathcal{F}}(\alpha_r - f_k)}\right) \frac{1}{f_k - \alpha_n} = \frac{1}{f_k - \alpha_n} + P_{\alpha_n}(|\mathcal{F}| - 1), \quad (26)$$

where $P_{\alpha_n}(|\mathcal{F}| - 1)$ is a polynomial of α_n of degree $|\mathcal{F}| - 1$.

[4]These conditions will be evident as the description of the scheme progresses.

[5]The purpose of the null shaper is to reduce the writing cost further by using any extra noise terms (even more than what is required for the single bit update decomposition) present in storage.

The proofs of Lemmas 1 and 2 are skipped for brevity. With the above definitions and lemmas, the incremental update is calculated by,

$$\bar{U}_n = D_n \times \Omega_n \times U_n \times \mathbf{Q}_n \tag{27}$$

$$= \begin{bmatrix} \Delta_{\theta,1}\mathbf{e}_M(\theta) \\ \Delta_{\theta,2}\mathbf{e}_M(\theta) \\ \vdots \\ \Delta_{\theta,\ell}\mathbf{e}_M(\theta) \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1-\alpha_n)\sum_{i=0}^{X_u-1}\alpha_n^i \hat{R}_{1,i}^{[1]} \\ \vdots \\ (f_1-\alpha_n)\sum_{i=0}^{X_u-1}\alpha_n^i \hat{R}_{M,i}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_\ell-\alpha_n)\sum_{i=0}^{X_u-1}\alpha_n^i \hat{R}_{1,i}^{[\ell]} \\ \vdots \\ (f_\ell-\alpha_n)\sum_{i=0}^{X_u-1}\alpha_n^i \hat{R}_{M,i}^{[\ell]} \end{bmatrix} \end{bmatrix}, \tag{28}$$

where the polynomial coefficients $\hat{R}_{i,j}^{[k]}$ are combined noise terms that do not depend on n. Note that for databases $r \in \mathcal{F}$, $\Omega_r = 0$, which makes $\bar{U}_r = 0$. Therefore, the user does not send U_n to those databases in $\mathcal{F}$ in the writing phase. For each database $n \in \{1,\dots,N\}\backslash\mathcal{F}$, the incremental update in (28) is in the same format as the storage in (19). Therefore, the updated storage is given by,

$$\mathbf{S}_n^{[t]} = \mathbf{S}_n^{[t-1]} + \bar{U}_n, \quad n \in \{1,\dots,N\}, \tag{29}$$

where $\mathbf{S}_n^{[t-1]}$ and $\mathbf{S}_n^{[t]}$ are the storages of database n before and after updating, respectively. The writing cost of this scheme is given by,

$$C_W = \frac{N-|\mathcal{F}|}{\ell} = \frac{2N-2X_u+Y_u-1}{N-X_u-T_u}. \tag{30}$$

Thus, using (23), the total communication cost is,

$$C_T = C_R + C_W = \frac{3N-2X_u+Y_u-1}{N-X_u-T_u}. \tag{31}$$

B. Optimal Values of (T_u, Y_u, X_u)

The general description and cost calculations in Section V-A are presented for arbitrary (T_u, Y_u, X_u) satisfying $T_u \geq T_r$, $T_u \geq Y_r$ and $\max\{X_r, \frac{N+Y_u-1}{2}\} \leq X_u \leq N-T_u-1$, where the last condition is derived from $|\mathcal{F}| \geq 0$ and $\ell \geq 1$. In this section, we present the optimum values of (T_u, Y_u, X_u) that minimize the total cost for a given N and PS-requirements. Since the total cost in (31) increases with T_u and Y_u, the optimum values of T_u and Y_u satisfying the privacy constraints are $T_u^* = T_r$ and $Y_u^* = Y_r$. The resulting total cost is then,

$$C_T = \frac{3N-2X_u+Y_r-1}{N-X_u-T_r}, \tag{32}$$

which is increasing in X_u since, $\frac{dC_T}{dX_u} = \frac{N+2T_r+Y_r-1}{(N-X_u-T_r)^2} > 0$. Therefore, the optimum value of X_u is given by,

$$X_u^* = \max\{X_r, \lceil \frac{N+Y_r-1}{2} \rceil\}. \tag{33}$$

The resulting optimum reading writing and total costs are given by (4) and (5), respectively.

Remark 6 *The null shaper is not used in the optimum setting when* $\lceil \frac{N+Y_r-1}{2} \rceil > X_r$ *except when* $N+Y_r-1$ *is odd due to the presence of the unavoidable additional noise term resulted by the rounding function.*

VI. Conclusion

In this paper, we proposed an efficient PRUW scheme for an arbitrary number of databases, submodels and arbitrary PS-requirements. The proposed scheme achieves a lower communication cost compared to what is achieved by the existing scheme in [7] by combining multiple parameter updates into a single bit in the writing phase. This is achieved by over-designing the system such that the storage contains more noise terms than what is required to satisfy the security constraint.

In this work, we extended [7] and provided an improved scheme, which achieves a lower total cost. After the acceptance of our paper, we became aware of independently and concurrently extended version of [7] in [17], which provided a scheme similar to ours in a broader context.

References

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

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, and M. Bennis *et al.* Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, June 2021.

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

[4] S. Ulukus, S. Avestimehr, M. Gastpar, S. A. Jafar, R. Tandon, and C. Tian. Private retrieval, computing and learning: Recent progress and future challenges. Available online at arxiv:2108.00026.

[5] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *MobiCom*, April 2020.

[6] M. Kim and J. Lee. Information-theoretic privacy in federated submodel learning. Available online at arXiv:2008.07656.

[7] Z. Jia and S. A. Jafar. X-secure T-private federated submodel learning. In *IEEE ICC*, June 2021.

[8] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, November 1998.

[9] H. Sun and S. A. Jafar. The capacity of private information retrieval. *IEEE Transactions on Information Theory*, 63(7):4075–4088, July 2017.

[10] K. Banawan and S. Ulukus. The capacity of private information retrieval from coded databases. *IEEE Transactions on Information Theory*, 64(3):1945–1956, March 2018.

[11] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, and C. Hollanti. Private information retrieval from coded storage systems with colluding, Byzantine, and unresponsive servers. *IEEE Transactions on Information Theory*, 65(6):3898–3906, June 2019.

[12] S. Kumar, H.-Y. Lin, E. Rosnes, and A. G. i Amat. Achieving maximum distance separable private information retrieval capacity with linear codes. *IEEE Trans. on Information Theory*, 65(7):4243–4273, July 2019.

[13] C. Tian, H. Sun, and J. Chen. Capacity-achieving private information retrieval codes with optimal message size and upload cost. *IEEE Transactions on Information Theory*, 65(11):7613–7627, November 2019.

[14] I. Samy, M. Attia, R. Tandon, and L. Lazos. Asymmetric leaky private information retrieval. *IEEE Transactions on Information Theory*, 67(8):5352–5369, August 2021.

[15] C. Naim, R. D'Oliveira, and S. El Rouayheb. Private multi-group aggregation. *IEEE ISIT*, July 2021.

[16] Z. Jia, H. Sun, and S. A. Jafar. Cross subspace alignment and the asymptotic capacity of X-secure T-private information retrieval. *IEEE Transactions on Information Theory*, 65(9):5783–5798, September 2019.

[17] Z. Jia and S. A. Jafar. X -secure T-private federated submodel learning with elastic dropout resilience. Available online at arXiv:2010.01059.