# $X$-Secure $T$-Private Federated Submodel Learning with Elastic Dropout Resilience

Zhuqing Jia, *Member, IEEE,* and Syed Ali Jafar, *Fellow, IEEE*

***Abstract*—Motivated by recent interest in federated submodel learning, this work explores the fundamental problem of privately reading from and writing to a database comprised of $K$ files (submodels) that are stored across $N$ distributed servers according to an $X$-secure threshold secret sharing scheme. One after another, various users wish to retrieve their desired file, locally process the information and then update the file in the distributed database while keeping the identity of their desired file private from any set of up to $T$ colluding servers. The availability of servers changes over time, so elastic dropout resilience is required. The main contribution of this work is an adaptive scheme, called ACSA-RW, that takes advantage of all currently available servers to reduce its communication costs, fully updates the database after each write operation even though the database is only partially accessible due to server dropouts, and ensures a memoryless operation of the network in the sense that the storage structure is preserved and future users may remain oblivious of the past history of server dropouts. The ACSA-RW construction builds upon cross-subspace alignment (CSA) codes that were originally introduced for $X$-secure $T$-private information retrieval and have been shown to be natural solutions for secure distributed batch matrix multiplication problems. ACSA-RW achieves the desired private read and write functionality with elastic dropout resilience, matches the best results for private-read from PIR literature, improves significantly upon available baselines for private-write, reveals a striking symmetry between upload and download costs, and exploits storage redundancy to accommodate arbitrary read and write dropout servers up to certain threshold values. It also answers in the affirmative an open question by Kairouz et al. for the case of partially colluding servers (i.e., tolerating collusion up to a threshold) by exploiting synergistic gains from the joint design of private read and write operations.**

***Index Terms*—Federated learning, Security, Privacy**

## I. Introduction

THE rise of machine learning is marked by fundamental tradeoffs between competing concerns. Central to this work are 1) the need for abundant training data, 2) the need for privacy, and 3) the need for low communication cost. Federated learning [1]–[4] is a distributed machine learning paradigm that aims to address the three concerns simultaneously by allowing distributed users/clients (e.g., mobile phones) to collaboratively train a shared model that is stored in a cluster of databases/servers (cloud) while keeping their training data private. The users retrieve the current model, train the model locally with their own training data, and then aggregate the modifications as focused updates. Thus, federated learning allows utilization of abundant training data while mitigating privacy concerns. Uploading focused updates (typically the same size as the model) also saves communication cost compared with uploading training data.

Communicating the full model may be unnecessary for large scale machine learning tasks where each user's local data is primarily relevant to a small part of the overall model. Federated *Submodel* Learning (FSL) [5] builds on this observation by partitioning the model into multiple submodels and allowing users to selectively train and update the submodels that are most relevant to their local view. This is the case, for example, in the binary relevance method for multi-label classification [6], [7], which *independently* trains a series of binary classifiers (viewed as submodels), one for each label. Given a sample to be predicted, the compound model predicts all labels for which the respective classifiers yield a positive result.

What makes federated submodel learning challenging is the privacy constraint. The identity of the submodel that is being retrieved and updated by a user must remain private. Prior works [5], [8]–[11] that assume centralized storage of all submodels are generally able to provide relatively weaker privacy guarantees such as plausible deniability through differential privacy mechanisms that perturb the data, and secure aggregation that relies on secure inter-user peer-to-peer communication (e.g., the computationally secure inter-user peer-to-peer communication protocol with the central server as the relay, based on Diffie–Hellman key exchange). On the other hand, it is noted recently by Kim and Lee in [12] that if the servers that store the submodels are *distributed*, then stronger information theoretic guarantees such[1] as "perfect privacy" (with partially colluding servers, i.e., tolerating collusion up to a threshold) may be attainable, without the need for user-to-user communication. Indeed, in this work we focus on this setting of distributed servers and perfect privacy. The challenge of federated submodel learning in this setting centers around three key questions.

The work of Zhuqing Jia and Syed Jafar is supported in part by funding from NSF grants CCF-1907053, ONR grant N00014-21-1-2386, and ARO grant W911NF1910344.

Z. Jia is with the School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876 China (e-mail: zhuqingj@bupt.edu.cn).

S. A. Jafar is with the Center for Pervasive Communications and Computing, Department of Electrical Engineering and Computer Science, University of California at Irvine, Irvine, CA 92697 USA (e-mail: syed@uci.edu).



This paper was presented in part at IEEE ICC 2021.

[1] By *perfect* privacy we mean that absolutely no information is leaked about the identity of a user's desired submodel to any set of colluding servers up to a target threshold.

**Q1 Private Read:** How can a user efficiently retrieve the desired submodel from the distributed servers without revealing which submodel is being retrieved?

**Q2 Private Write:** How can a user efficiently update the desired submodel to the distributed servers without revealing which submodel is being updated?

**Q3 Synergy of Private Read-Write:** Are there synergistic gains in the *joint* design of retrieval and update operations, and if so, then how to exploit these synergies?

The significance of these fundamental questions goes well beyond federated submodel learning. As recognized by [5] the private read question (Q1) by itself is equivalent to the problem of Private Information Retrieval (PIR) [13], [14], which has recently been studied extensively from an information theoretic perspective [15]–[50]. Much less is known about Q2 and Q3, i.e., the fundamental limits of private-write, and joint read-write solutions from the information theoretic perspective. Notably, Q3 has also been highlighted previously as an open problem by Kairouz et al in [2].

The problem of privately reading and writing data from a distributed memory falls under the larger umbrella of Distributed Oblivious RAM (DORAM) [51] primitives in theoretical computer science and cryptography. With a few limited exceptions (e.g., a specialized 4-server construction in [52] that allows information theoretic privacy), prior studies of DORAM generally take a cryptographic perspective, e.g., privacy is guaranteed subject to computational hardness assumptions, and the number of memory blocks is assumed to be much larger than the size of each block. In contrast, the focus of this work is on Q2 and Q3 under the stronger notion of information theoretic privacy. Furthermore, because our motivation comes from federated submodel learning, the size of a submodel is assumed to be significantly larger than the number of submodels (see motivating examples in [5] and Section III-A10). Indeed, this is a pervasive assumption in the growing body of literature on information theoretic PIR [15]–[49]. In a broad sense, our problem formulation in this paper is motivated by applications of (information theoretic) PIR that also require private writes. There is no shortage of such applications, e.g., a distributed database of medical records that not only allows a physician to privately download the desired record (private read) but also to update the record with new information (private write), or a banking service that would similarly allow private reads and writes of financial records from authorized entities. Essentially, while FSL serves as our nominal application of interest based on prior works that motivated this effort, our problem formulation is broad enough to capture various distributed file systems that enable the users to read and write files without revealing the identity of the target file. The files are viewed as submodels, and the assumption that the size of the file is significantly larger than the number of the files captures the nature of file systems that are most relevant to this work.

*Overview:* We consider the federated submodel learning setting where the global model is partitioned into $K$ submodels, and stored among $N$ distributed servers according to an $X$-secure threshold secret sharing scheme, i.e., any set of up to $X$ colluding servers can learn nothing about the stored models, while the full model can be recovered from the data stored by any $X+K_c$ servers. One at a time, users update the submodel most relevant to their local training data. The updates must be $T$-private, i.e., any set of up to $T$ colluding servers must not learn anything about which submodel is being updated. The contents of the updates must be $X_\Delta$-secure, i.e., any set of up to $X_\Delta$ colluding servers must learn nothing about the contents of the submodel updates. The size of a submodel is assumed to be significantly larger than the number of submodels, which is significantly larger than 1, i.e., $L \gg K \gg 1$ where $L$ is the size of a submodel and $K$ is the number of submodels. Due to uncertainties of the servers' I/O states, link states, etc., an important concern in distributed systems is to allow resilience against servers that may temporarily drop out [53]–[57]. To this end, we assume that at each time $t, t \in \mathbb{N}$, a subset of servers may be unavailable. These unavailable servers are referred to as read-dropout servers or write-dropout servers depending on whether the user intends to perform the private read or the private write operation. Since the set of dropout servers changes over time, and is assumed to be known to the user, the private read and write schemes must *adapt* to the set of currently available servers. Note that this is different from the problem of stragglers in massive distributed computing applications where the set of responsive servers is not known in advance, because servers may become unavailable *during* the lengthy time interval required for their local computations. Since our focus is not on massive computing applications, the server side processing needed for private read and write is not as time-consuming (at most linear in the size of the global model) compared with the server side computing required for massive computing (e.g., massive matrix multiplication) tasks (polynomial running time typically). So the availabilities, which are determined in advance by the user before initiating the read or write operation, e.g., by pinging the servers, are not expected to change during the read or write operation. We do allow the server availabilities to change between the read and write operations due to the delay introduced by the intermediate processing that is needed at the user to generate his updated submodel. A somewhat surprising aspect of private write with unavailable servers is that even though the data at the unavailable servers cannot be updated, the collective storage at all servers (including the unavailable ones) must represent the updated models. The redundancy in coded storage and the $X$-security constraints which require that the stored information at any $X$ servers is independent of the data, are essential in this regard.

Since the private-read problem (Q1) is essentially a form of PIR, our starting point is the $X$-secure $T$-private information retrieval scheme (XSTPIR) of [30]. In particular, we build on the idea of cross-subspace alignment (CSA) from [30], and introduce a new private read-write scheme, called Adaptive CSA-RW (ACSA-RW) as an answer to Q1 and Q2. ACSA-RW is a federated submodel learning scheme that guarantees information-theoretically *perfect* privacy (with partially colluding servers), achieves the (conjectured) asymptotic optimal download cost, and is at least orderwise optimal in its upload cost (see Section III-A3, III-A8 and III-A9 for details). ACSA-RW also answers Q3 in the affirmative for the case of

partially-colluding servers as it exploits query structure from the private-read operation to reduce the communication cost for the private-write operation. The evidence of synergistic gain in ACSA-RW from a joint design of submodel retrieval and submodel aggregation addresses the corresponding open problem highlighted in Section 4.4.4 of [2]. The observation that the ACSA-RW scheme takes advantage of storage redundancy for private read and private write is indicative of fundamental tradeoffs between download cost, upload cost, data security level, and storage redundancy for security and recoverability. In particular, the storage redundancy for $X$-security is exploited by private write, while the storage redundancy for robust recoverability is used for private read (see Theorem 1 and Section III-A1 for details). It is also remarkable that the ACSA-RW scheme requires absolutely no user-user communication, even though the server states change over time and the read-write operations are adaptive. In other words, a user is not required to be aware of the history of previous updates and the previous availability states of the servers (see Section III-A7 for details). The download cost and the upload cost achieved by each user is an increasing function of the number of unavailable servers at the time. When more servers are available, the download cost and the upload costs are reduced, which provides elastic dropout resilience (see Theorem 1). To this end, the ACSA-RW scheme uses adaptive MDS-coded answer strings. This idea originates from CSA code constructions for the problem of coded distributed batch computation [58]. In terms of comparisons against available baselines, we note (see Section III-A8) that ACSA-RW improves significantly in both the communication efficiency and the level of privacy compared to [12]. In fact, ACSA-RW achieves asymptotically optimal download cost when $X \geq X_\Delta + T$, and is order-wise optimal in terms of the upload cost. Compared with the 4 server construction of information theoretic DORAM in [52], (where $X = 1, T = 1, X_\Delta = 0, N = 4$) ACSA-RW has better communication efficiency (the assumption of $L \gg K$ is important in this regard). For example, as the ratio $L/K$ approaches infinity, ACSA-RW achieves total communication cost (i.e., the summation of the download cost and the upload cost, normalized by the submodel size) of 6, versus the communication cost of 8 achieved by the construction in [52].

*Notation:* Bold symbols are used to denote vectors and matrices, while calligraphic symbols denote sets. By convention, let the empty product be the multiplicative identity, and the empty sum be the additive identity. For two positive integers $M, N$ such that $M \leq N$, $[M : N]$ denotes the set $\{M, M+1, \cdots, N\}$. We use the shorthand notation $[N]$ for $[1 : N]$. $\mathbb{N}$ denotes the set of positive integers $\{1, 2, 3, \cdots\}$, and $\mathbb{Z}^*$ denotes the set $\mathbb{N} \cup \{0\}$. For a subset of integers $\mathcal{N} \subset \mathbb{N}$, $\mathcal{N}(i), i \in [|\mathcal{N}|]$ denotes its $i^{th}$ element, sorted in ascending order. The notation $\text{diag}(\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_n)$ denotes the block diagonal matrix, i.e., the main-diagonal blocks are square matrices $(\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_n)$ and all off-diagonal blocks are zero matrices. For a positive integer $K$, $\mathbf{I}_K$ denotes the $K \times K$ identity matrix. For two positive integers $k, K$ such that $k \leq K$, $\mathbf{e}_K(k)$ denotes the $k^{th}$ column of the $K \times K$ identity matrix. The notation $\widetilde{\mathcal{O}}(a \log^2 b)$ suppresses[2] polylog terms. It may be replaced with $\mathcal{O}(a \log^2 b)$ if the field supports the Fast Fourier Transform (FFT)[3], and with $\mathcal{O}(a \log^2 b \log\log(b))$ if it does not[4].

## II. Problem Statement: Robust XSTPFSL

Consider $K$ initial submodels $\left(\mathbf{W}_1^{(0)}, \mathbf{W}_2^{(0)}, \cdots, \mathbf{W}_K^{(0)}\right)$, each of which consists of $L$ uniformly i.i.d.[5] random symbols from a finite field[6] $\mathbb{F}_q$. In particular, we have

$$H\left(\mathbf{W}_1^{(0)}, \mathbf{W}_2^{(0)}, \cdots, \mathbf{W}_K^{(0)}\right) = KL, \tag{1}$$

in $q$-ary units. Indeed, finite field symbols are used as a representation of the submodels, we refer readers to Section III-A2 for detailed explanation. Time slots are associated with users and their corresponding submodel updates, i.e., at time slot $t, t \in \mathbb{N}$, User $t$ wishes to perform the $t^{th}$ submodel update. At time $t, t \in \mathbb{Z}^*$, the $K$ submodels are denoted as $\left(\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)}, \cdots, \mathbf{W}_K^{(t)}\right)$. The submodels are represented as vectors, i.e., for all $t \in \mathbb{Z}^*$, $k \in [K]$,

$$\mathbf{W}_k^{(t)} = \left[W_k^{(t)}(1), W_k^{(t)}(2), \cdots, W_k^{(t)}(L)\right]^\mathsf{T}. \tag{2}$$

The $K$ submodels are distributively stored among the $N$ servers. The storage at server $n, n \in [N]$ at time $t, t \in \mathbb{Z}^*$ is denoted as $\mathbf{S}_n^{(t)}$. Note that $\mathbf{S}_n^{(0)}$ represents the initial storage.

A full cycle of robust XSTPFSL is comprised of two phases — the read phase and the write phase. At the beginning of the cycle, User $t$ privately generates the desired index $\theta_t$, uniformly from $[K]$, and the user-side randomness $\mathcal{Z}_U^{(t)}$, which is intended to protect the user's privacy and security. In the read phase, User $t$ wishes to retrieve the submodel $\mathbf{W}_{\theta_t}^{(t-1)}$. At all times, nothing must be revealed about any (current or past) desired indices $(\theta_1, \theta_2, \cdots, \theta_t)$ to any set of up to $T$ colluding servers. To this end, User $t$ generates the read-queries $\left(Q_1^{(t,\theta_t)}, Q_2^{(t,\theta_t)}, \cdots, Q_N^{(t,\theta_t)}\right)$, where $Q_n^{(t,\theta_t)}$ is intended for the $n^{th}$ server, such that,

$$H\left(Q_n^{(t,\theta_t)} \,\middle|\, \theta_t, \mathcal{Z}_U^{(t)}\right) = 0, \qquad \forall n \in [N]. \tag{3}$$

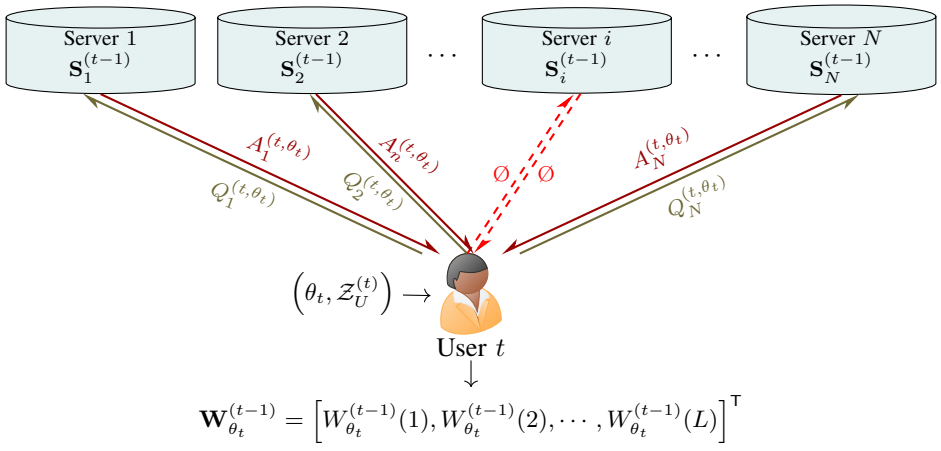
[2]There is another standard definition of the notation $\widetilde{\mathcal{O}}$ which fully suppresses polylog terms, i.e, $\mathcal{O}(a\,\text{polylog}(b))$ is represented by $\widetilde{\mathcal{O}}(a)$, regardless of the exact form of $\text{polylog}(b)$. The definition used in this paper emphasizes the dominant factor in the polylog term.

[3]FFT is used in fast encoding/decoding algorithms, where structured matrix algebras are involved and convolutions are required.

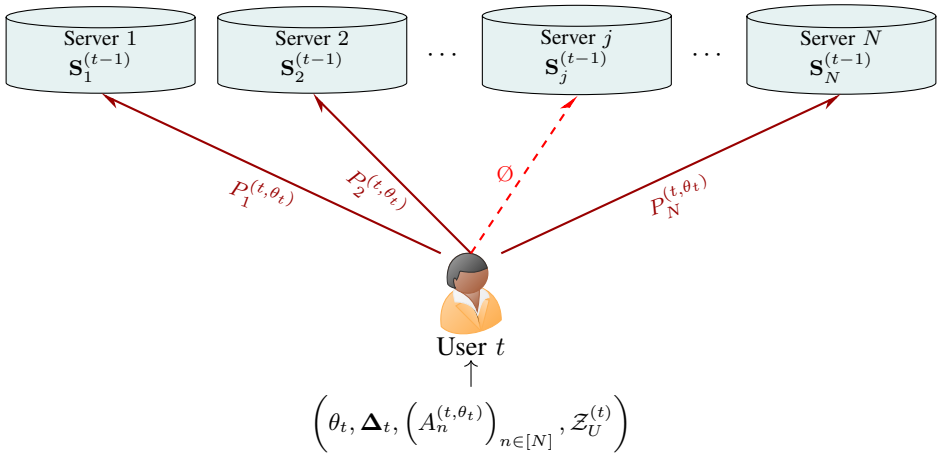
[4]If the FFT is not supported by the field, Schönhage–Strassen algorithm [59] can be used for fast algorithms that require convolutions, with an extra factor of $\log\log b$ in the complexity.

[5]Note that the proposed ACSA-RW scheme does not require the assumption of uniformly i.i.d. model data and uniformly i.i.d. desired index to be correct, secure and private. The assumption is mainly used for converse arguments and communication cost metrics. However, we note that the uniformly i.i.d. model data assumption is in fact *not* very strong because submodel learning is performed locally, and it may be possible to achieve (nearly) uniformly i.i.d. model data by exploiting entropy encoding.

[6]Note that the size of the finite field required is not too large. By our ACSA-RW scheme (see Theorem 1), it is sufficient to choose a finite field with $q \geq 2N$.




(a) The private-read phase of robust XSTPFSL. The $i^{th}$ server is unavailable, $i \in \mathcal{S}_r^{(t)}$.




(b) The private-write phase of robust XSTPFSL. The $j^{th}$ server is unavailable, $j \in \mathcal{S}_w^{(t)}$. Note that Server $i$ and Server $j$ may be different servers.

Fig. 1: The two phases of robust $X$-Secure $T$-Private Federated Submodel Learning (XSTPFSL) with arbitrary realizations of unavailable servers.

Each of the currently available servers $n, n \in [N] \setminus \mathcal{S}_r^{(t)}$ is sent the query $Q_n^{(t,\theta_t)}$ by the user, and responds to the user with an answer $A_n^{(t,\theta_t)}$, such that,

$$H\left(A_n^{(t,\theta_t)} \,\middle|\, \mathbf{S}_n^{(t-1)}, Q_n^{(t,\theta_t)}\right) = 0, \quad \forall n \in [N] \setminus \mathcal{S}_r^{(t)}. \tag{4}$$

From the answers returned by the servers $n, n \in [N]\setminus\mathcal{S}_r^{(t)}$, the user must be able to reconstruct the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$.

[Correctness]

$$H\left(\mathbf{W}_{\theta_t}^{(t-1)} \,\middle|\, \left(A_n^{(t,\theta_t)}\right)_{n\in[N]\setminus\mathcal{S}_r^{(t)}}, \left(Q_n^{(t,\theta_t)}\right)_{n\in[N]}, \theta_t\right) = 0. \tag{5}$$

This is the end of the read phase.

Upon finishing the local submodel training, User $t$ privately generates an increment[7] for the $\theta_t^{th}$ submodel. The increment is represented as a vector, $\mathbf{\Delta}_t = [\Delta_1^{(t)}, \Delta_2^{(t)}, \cdots, \Delta_L^{(t)}]^\mathsf{T}$, which consists of $L$ i.i.d. uniformly distributed symbols from the finite field $\mathbb{F}_q$, i.e., $H(\mathbf{\Delta}_t) = L$ in $q$-ary units. The increment $\mathbf{\Delta}_t$ is intended to update the $\theta_t^{th}$ submodel $\mathbf{W}_{\theta_t}^{(t-1)}$, such that the next user who wishes to make an update, User $t+1$, is able to retrieve the submodel $\mathbf{W}_{\theta_t}^{(t)} = \mathbf{W}_{\theta_t}^{(t-1)} + \mathbf{\Delta}_t$ if $\theta_{t+1} = \theta_t$, and the submodel $\mathbf{W}_{\theta_t'}^{(t)} = \mathbf{W}_{\theta_t'}^{(t-1)}$ if $\theta_{t+1} = \theta_t' \neq \theta_t$. In other words, for all $t \in \mathbb{N}$, $k \in [K]$, the submodel $\mathbf{W}_k^{(t)}$ is defined recursively as follows.

$$\mathbf{W}_k^{(t)} = \begin{cases} \mathbf{W}_k^{(t-1)} + \mathbf{\Delta}_t & k = \theta_t, \\ \mathbf{W}_k^{(t-1)} & k \neq \theta_t. \end{cases} \tag{6}$$

User $t$ initializes the write phase by generating the write-queries $\left(P_1^{(t,\theta_t)}, P_2^{(t,\theta_t)}, \cdots, P_N^{(t,\theta_t)}\right)$. For ease of notation, let the write-queries be nulls for the write-dropout servers, i.e., $P_n^{(t,\theta_t)} = \emptyset$ for $n \in \mathcal{S}_w^{(t)}$. For all $n \in [N]$,

$$H\left(P_n^{(t,\theta_t)} \,\middle|\, \theta_t, \mathcal{Z}_U^{(t)}, \left(A_n^{(t,\theta_t)}\right)_{n\in[N]}, \mathbf{\Delta}_t\right) = 0. \tag{7}$$

[7] In general, the increment is the difference between the new submodel and the old submodel. We note that no generality is lost by assuming additive increments because the submodel training is performed locally.

The user sends the write-query $P_n^{(t,\theta_t)}$ to the $n^{th}$ server, $n \in [N] \setminus \mathcal{S}_w^{(t)}$, if the server was available in the read-phase and therefore already received the read-query. Otherwise, if the server was not available during the read phase, then the user sends[8] both read and write queries $(Q_n^{(t,\theta_t)}, P_n^{(t,\theta_t)})$. Still, any set of up to $T$ colluding servers must learn nothing about the desired indices $(\theta_1, \theta_2, \cdots, \theta_t)$.

Upon receiving the write-queries, each of the servers $n, n \in [N] \setminus \mathcal{S}_w^{(t)}$ updates its storage based on the existing storage $\mathbf{S}_n^{(t-1)}$ and the queries for the two phases $\left(P_n^{(t,\theta_t)}, Q_n^{(t,\theta_t)}\right)$, i.e.,

$$H\left(\mathbf{S}_n^{(t)} \;\middle|\; \mathbf{S}_n^{(t-1)}, P_n^{(t,\theta_t)}, Q_n^{(t,\theta_t)}\right) = 0. \tag{8}$$

On the other hand, the write-dropout servers are unable to perform any storage update.

$$\mathbf{S}_n^{(t)} = \mathbf{S}_n^{(t-1)}, \qquad \forall n \in \mathcal{S}_w^{(t)}. \tag{9}$$

Next, let us formalize the security and privacy constraints. $T$-privacy guarantees that at any time, any set of up to $T$ colluding servers learn nothing about the indices $(\theta_1, \theta_2, \cdots, \theta_t)$ from all the read and write queries and storage states.

[$T$-Privacy]

$$I\left((\theta_\tau)_{\tau\in[t]}; \left(P_n^{(t,\theta_t)}, Q_n^{(t,\theta_t)}\right)_{n\in\mathcal{T}} \;\middle|\; \left(\mathbf{S}_n^{(\tau-1)}, P_n^{(\tau-1,\theta_{\tau-1})}, Q_n^{(\tau-1,\theta_{\tau-1})}\right)_{\tau\in[t],n\in\mathcal{T}}\right) = 0, \forall \mathcal{T} \in [N], |\mathcal{T}| = T, t \in \mathbb{N}, \tag{10}$$

where for all $n \in [N]$, we define $P_n^{(0,\theta_0)} = Q_n^{(0,\theta_0)} = \emptyset$.

Similarly, any set of up to $X_\Delta$ colluding servers must learn nothing about the increments $(\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_t)$.

[$X_\Delta$-Security]

$$I\left((\mathbf{\Delta}_\tau)_{\tau\in[t]}; \left(P_n^{(t,\theta_t)}\right)_{n\in\mathcal{X}} \;\middle|\; \left(\mathbf{S}_n^{(\tau-1)}, P_n^{(\tau-1,\theta_{\tau-1})}, Q_n^{(\tau,\theta_\tau)}\right)_{\tau\in[t],n\in\mathcal{X}}\right) = 0, \forall \mathcal{X} \subset [N], |\mathcal{X}| = X_\Delta, t \in \mathbb{N}. \tag{11}$$

The storage at the $N$ servers is formalized according to a threshold secret sharing scheme. Specifically, the storage at any set of up to $X$ colluding servers must reveal nothing about the submodels. Formally,

$$H\left(\mathbf{S}_n^{(0)} \;\middle|\; \left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}, \mathcal{Z}_S\right) = 0, \forall n \in [N], \tag{12}$$

[$X$-Security]

$$I\left(\left(\mathbf{W}_k^{(t)}\right)_{k\in[K]}; \left(\mathbf{S}_n^{(t)}\right)_{n\in\mathcal{X}}\right) = 0, \quad \forall \mathcal{X} \subset [N], |\mathcal{X}| = X, t \in \mathbb{Z}^*, \tag{13}$$

[8]For ease of exposition we will assume in the description of the scheme that during the read phase, the read queries are sent to *all* servers that are available during the read phase, or will become available later during the write phase, so that only the write-queries need to be sent during the write phase.

where for all $n \in [N]$, we define $\mathbf{S}_n^{-1} = \emptyset$. Note that $\mathcal{Z}_S$ is the private randomness used by the secret sharing scheme that implements $X$-secure storage across the $N$ servers.

There is a subtle difference in the security constraint that we impose on the storage, and the previously specified security and privacy constraints on updates and queries. To appreciate this difference let us make a distinction between the notions of an internal adversary and an external adversary. We say that a set of colluding servers forms an internal adversary if those colluding servers have access to not only their current storage, but also their entire history of previous stored values and queries. Essentially the internal adversary setting represents a greater security threat because the servers themselves are dishonest and surreptitiously keep records of all their history in an attempt to learn from it. In contrast, we say that a set of colluding servers forms an external adversary if those colluding servers have access to only their current storage, but not to other historical information. Essentially, this represents an external adversary who is able to steal the current information from honest servers who do not keep records of their historical information. Clearly, an external adversary is weaker than an internal adversary. Now let us note that while the $T$-private queries and the $X_\Delta$-secure updates are protected against *internal* adversaries, the $X$-secure storage is only protected against *external* adversaries. This is mainly because we will generally assume $X > \max(X_\Delta, T)$, i.e., a higher security threshold for storage, than for updates and queries. Note that once the number of compromised servers exceeds $\max(X_\Delta, T)$, the security of updates and the privacy of queries is no longer guaranteed. In such settings the security of storage is still guaranteed, albeit in a weaker sense (against external adversaries). On the other hand if the number of compromised servers is small enough, then indeed secure storage may be guaranteed in a stronger sense, even against internal adversaries. We refer the reader to Remark 6 for further insight into this aspect.

The independence among various quantities of interest is specified for all $t \in \mathbb{N}$ as follows.

$$\begin{aligned} &H\left(\left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}, (\mathbf{\Delta}_\tau)_{\tau\in[t]}, (\theta_\tau)_{\tau\in[t]}, \left(\mathcal{Z}_U^{(\tau)}\right)_{\tau\in[t]}, \mathcal{Z}_S\right) \\ &= H\left(\left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}\right) + H\left((\mathbf{\Delta}_\tau)_{\tau\in[t]}\right) + H\left((\theta_\tau)_{\tau\in[t]}\right) \\ &\quad + H\left(\left(\mathcal{Z}_U^{(\tau)}\right)_{\tau\in[t]}\right) + H(\mathcal{Z}_S). \end{aligned} \tag{14}$$

To evaluate the performance of a robust XSTPFSL scheme, we consider the following metrics. The first two metrics focus on communication cost. For $t \in \mathbb{N}$, the download cost $D_t$ is the expected (over all realizations of queries) number of $q$-ary symbols downloaded by User $t$, normalized by $L$. The upload cost $U_t$ is the expected number of $q$-ary symbols uploaded by User $t$, also normalized by $L$. The next metric focuses on storage efficiency. For $t \in \mathbb{Z}^*$, the storage efficiency is defined as the ratio of the total data content to the total storage resources consumed by a scheme, i.e., $\eta^{(t)} = \frac{KL}{\sum_{n\in[N]} H\left(S_n^{(t)}\right)}$. If $\eta^{(t)}$ takes the same value for all $t \in \mathbb{Z}^*$, we use the compact notation $\eta$ instead.

## III. Main Result: The ACSA-RW Scheme for Private Read/Write

The main contribution of this work is the ACSA-RW scheme, which allows private read and private write from $N$ distributed servers according to the problem statement provided in Section II. The scheme achieves storage efficiency $K_c/N$, i.e., it uses a total storage of $(KLN/K_c)\log_2 q$ bits across $N$ servers in order to store the $KL\log_2 q$ bits of actual data, where $K_c \in \mathbb{N}$, and allows arbitrary read-dropouts and write-dropouts as long as the number of dropout servers is less than the corresponding threshold values. The thresholds are defined below and their relationship to redundant storage dimensions is explained in Section III-A1.

$$\textbf{Read-dropout threshold: } S_r^{\text{thresh}} \triangleq N-(K_c+X+T-1). \tag{15}$$

$$\textbf{Write-dropout threshold: } S_w^{\text{thresh}} \triangleq X-(X_\Delta+T-1). \tag{16}$$

It is worth emphasizing that the ACSA-RW scheme does not just tolerate dropout servers, it adapts (hence the *elastic* resilience) to the number of available servers so as to reduce its communication cost. The elasticity would be straightforward if the only concern was the private-read operation because known replicated-storage based PIR schemes can be adapted to the number of available servers. This is because when the subset of servers that is unable to respond is known in advance, the problem simply reduces to replicated-storage based PIR with fewer servers. What makes the elasticity requirement non-trivial is that the scheme must accommodate both private read and private write. The private-write requirements are particularly intriguing, almost paradoxical in that the coded storage across all $N$ servers needs to be updated to be consistent with the new submodels, even though some of those servers (the write-dropout servers) are unavailable, so their stored information cannot be changed. Furthermore, as server states continue to change over time, future updates need no knowledge of prior dropout histories. Also of interest are the tradeoffs between storage redundancy and the resources needed for private-read and private-write functionalities. Because many of these aspects become more intuitively transparent when $L \gg K \gg 1$, the asymptotic setting is used to present the main result in Theorem 1. In particular, by suppressing minor terms (which can be found in the full description of the scheme), the asymptotic setting reveals an elegant symmetry between the upload and download costs. The remainder of this section is devoted to stating and then understanding the implications of Theorem 1. The scheme itself is presented in the form of the proof of Theorem 1 in Section IV.

**Theorem 1.** *In the limit $L/K \to \infty$, for all $t \in \mathbb{N}$ the ACSA-RW scheme achieves the following download, upload cost pair $(D_t, U_t)$ and storage efficiency $\eta$:*

$$(D_t,U_t)=\left(\frac{N-|\mathcal{S}_r^{(t)}|}{S_r^{\text{thresh}}-|\mathcal{S}_r^{(t)}|},\ \frac{N-|\mathcal{S}_w^{(t)}|}{S_w^{\text{thresh}}-|\mathcal{S}_w^{(t)}|}\right),\quad \eta=\frac{K_c}{N}, \tag{17}$$

*for any $K_c \in \mathbb{N}$ such that $|\mathcal{S}_r^{(t)}| < S_r^{\text{thresh}}$, $|\mathcal{S}_w^{(t)}| < S_w^{\text{thresh}}$, and the field size $q \geq N + \max\left\{S_r^{\text{thresh}}, S_w^{\text{thresh}}, K_c\right\}$.*

**Remark 1.** *For non-asymptotic region, the ACSA-RW scheme achieves the same download cost, and the upload cost achieved is shown in the formal presentation of the scheme, (102).*

### A. Observations

*1) Storage Redundancy and Private Read/Write Thresholds:* The relationship between redundant storage dimensions and the private read/write thresholds is conceptually illustrated in Figure 2.

The total storage utilized by the ACSA-RW scheme across $N$ servers is represented as the overall $N$ dimensional space in Figure 2. Out of this, the actual data occupies only $K_c$ dimensions, which is why the storage efficiency of ACSA-RW is $K_c/N$. Because the storage must be $X$-secure, i.e., any set of up to $X$ colluding servers cannot learn anything about the data, it follows that out of the $N$ dimensions of storage space, $X$ dimensions are occupied by information that is independent of the actual data. The storage redundancy represented by these $X$ dimensions will be essential to enable the private-write functionality. But first let us consider the private-read operation for which we have a number of prior results on PIR as baselines for validating our intuition. From Figure 2 we note that outside the $X$ dimensions of redundancy that was introduced due to data-security, the $T$-privacy constraint adds a storage redundancy of another $T-1$ dimensions that is shown in red. To understand this, compare the asymptotic (large $K$) capacity of MDS-PIR [60]: $C_{\text{MDS-PIR}} = 1 - K_c/N$ with the (conjectured) asymptotic capacity of MDS-TPIR [25], [26]: $C_{\text{MDS-TPIR}} = 1-(K_c+T-1)/N$. To achieve a non-zero value of the asymptotic capacity, the former requires $N > K_c$, but the latter requires $N > K_c + T - 1$. Equivalently, the former allows a read-dropout threshold of $N - K_c$ while the latter allows a read-dropout threshold of $N-(K_c+T-1)$. In fact, going further to the (conjectured) asymptotic capacity of MDS-XSTPIR [27], which also includes the $X$-security constraint, we note that a non-zero value of capacity requires $N > K_c + X + T - 1$. Intuitively, we may interpret this as: the $X$-security constraint increases the demands on storage redundancy by $X$ dimensions and the $T$-privacy constraint increases the demands on storage redundancy by another $T-1$ dimensions. This is what is represented in Figure 2. Aside from the $K_c$ dimensions occupied by data, the $X$ dimensions of redundancy added by the security constraint, and the $T-1$ dimensions of redundancy added by the $T$-privacy constraint, the remaining dimensions at the right end of the figure are used to accommodate read-dropouts. Indeed, this is what determines the read-dropout threshold, as we note from Figure 2 that $N-(X+K_c+(T-1)) = S_r^{\text{thresh}}$. Now consider the private-write operation which is novel and thus lacks comparative baselines. What is remarkable is the synergistic aspect of private-write, that it does not add further redundancy beyond the $X$ dimensions of storage redundancy already added by the $X$-security constraint. Instead, it operates within these $X$ dimensions to create further sub-partitions. Within these $X$

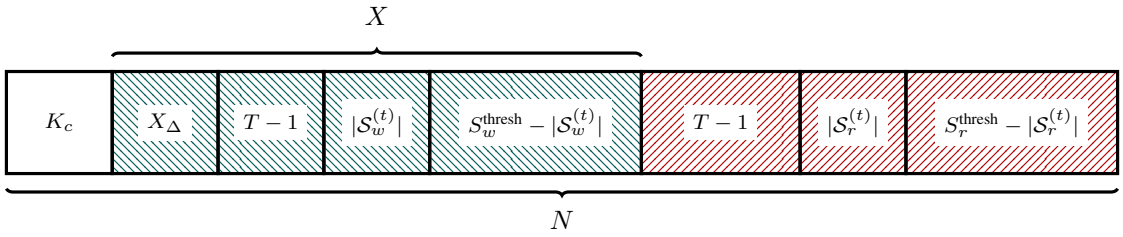



Fig. 2: Conceptual partitioning of total server storage space ($N$ dimensions) into data content ($K_c$ dimensions), storage redundancy that is exploited by private-write ($X$ dimensions), and storage redundancy that is exploited by private-read ($N - K_c - X$ dimensions).

dimensions, a total of $X_\Delta + T - 1$ dimensions are used to achieve $X_\Delta$-secure updates that also preserve $T$-privacy, and the remaining dimensions are used to accommodate write-dropout servers, giving us the write-dropout threshold as $X - (X_\Delta + T - 1) = S_w^{\text{thresh}}$. Remarkably in Figure 2, the $(K_c, X_\Delta, T-1, S_w^{\text{thresh}})$ partition structure for private-write replicates at a finer level the original $(K_c, X, T-1, S_r^{\text{thresh}})$ partition structure of private-read. Also remarkable is a new constraint introduced by the private-write operation that is not encountered in prior works on PIR — the feasibility of ACSA-RW requires $X \geq T$. Whether or not this constraint is fundamental in the asymptotic setting of large $K$ is an open problem for future work.

*2) On Finite Field Representation of Submodels:* Recall that we assume finite field symbol representation of the submodels. It is crucial to note that this allows *arbitrary* mapping (including quantization and possibly compression) of the real-valued submodels to the discrete alphabet (finite field vectors in this case), and does not require any form of "homomorphism" to be satisfied by the mapping. For example, initially, the $N$ distributed servers collectively store $\left(\mathbf{W}_k^{(0)}\right)_{k\in[K]}$ in an $X$-secured form where for all $k \in [K]$, $\mathbf{W}_k^{(0)} = f(\Omega_k^{(0)})$ and $f(\cdot)$ is an arbitrary mapping of the real-valued submodel to a finite field vector. Upon retrieving the desired submodel $\mathbf{W}_\theta^{(0)}$, the user is able to recover the finite field representation $\mathbf{W}_\theta^{(0)}$ of the desired submodel, from which there must be some mapping to a (quantized) real-valued submodel, say $g(\cdot)$. The user applies $g(\cdot)$ to obtain $\Omega_\theta^{'(0)}$ as the (quantized) old submodel, and trains locally to obtain the new real-valued submodel, $\Omega_\theta^{(1)}$. Applying the mapping $f(\cdot)$ on the new real-valued submodel, the user obtains its finite field representation $\mathbf{W}_\theta^{(1)} = f(\Omega_\theta^{(1)})$. Now the user computes the increment $\mathbf{\Delta}_1 = \mathbf{W}_\theta^{(1)} - \mathbf{W}_\theta^{(0)}$, and updates the desired submodel in the distributed database depending on which of the servers are available in the private write phase. Note that this requires no assumption of homomorphism in the mappings $f(\cdot)$ and $g(\cdot)$, which can be arbitrary. As a side remark, while our focus is only on communication costs, it is worthwhile to point out that computational complexity concerns may also be important in the choice of mappings $f(\cdot)$ and $g(\cdot)$.

*3) Optimality:* Asymptotic (large $K$) optimality of ACSA-RW remains an open question in general. Any attempt to resolve this question runs into other prominent open problems in the information-theoretic PIR literature, such as the asymptotic capacity of MDS-TPIR [25], [26] and MDS-XSTPIR [27] that also remain open. Nevertheless, it is worth noting that Theorem 1 matches or improves upon the best known results in all cases where such results are available. In particular, the private-read phase of ACSA-RW scheme recovers a universally robust $X$-secure $T$-private information retrieval [30], [61] scheme. When $K_c = 1, X \geq X_\Delta + T$, it achieves the asymptotic capacity; and when $K_c > 1, X \geq X_\Delta + T$, it achieves the conjectured asymptotic capacity [27]. While much less is known about optimal private-write schemes, it is clear that ACSA-RW significantly improves upon previous work as explained in Section III-A8. Notably, both upload and download costs are $O(1)$ in $K$[9], i.e., they do not scale with $K$. Thus, at the very least the costs are orderwise optimal. Another interesting point of reference is the best case scenario, where we have no dropout servers, $|\mathcal{S}_r^{(t)}| = 0, |\mathcal{S}_w^{(t)}| = 0$. The total communication cost of ACSA-RW in this case, i.e., the sum of upload and download costs, is $D_t + U_t = N\left(\frac{1}{S_r^{\text{thresh}}} + \frac{1}{S_w^{\text{thresh}}}\right)$, which is minimized when $K_c = 1$, $S_r^{\text{thresh}} = S_w^{\text{thresh}}$, and $X = (N + X_\Delta - 1)/2$. For large $N$, we have $D_t + U_t \approx 2 + 2 = 4$, thus in the best case scenario, ACSA-RW is optimal within a factor of 2. Finally, on a speculative note, perhaps the most striking aspect of Theorem 1 is the symmetry between upload and download costs, which (if not coincidental) bodes well for their fundamental significance and information theoretic optimality.

*4) The Choice of Parameter $K_c$:* The choice of the parameter $K_c$ in ACSA-RW determines the storage efficiency of the scheme, $\eta = K_c/N$. At one extreme, we have the smallest possible value of $K_c$, i.e., $K_c = 1$, which is the least efficient storage setting, indeed the storage efficiency is analogous to replicated storage, each server uses as much storage space as the size of all data ($KL\log_2 q$ bits). This setting yields the best (smallest) download costs. The other extreme corresponds to the maximum possible value of $K_c$, which is obtained as $K_c = N - (X + T)$ because the dropout thresholds cannot be smaller than 1. At this extreme, storage is the most efficient, but there is no storage redundancy left to accommodate any read dropouts, and the download cost of ACSA-RW takes its maximal value, equal to $N$. Remarkably, the upload cost of the private-write operation does not depend on $K_c$. However, $K_c$ is significant for another reason; it determines the access complexity (see Remark 7) of both private read and write

[9]Note that this is true even if $L$ is of the same order as $K$.

operations, i.e., the number of bits that are read from or written to by each available server. In particular, the access complexity of each available server in the private read or write phases is at most $(KL/K_c)\log_2 q$, so for example, increasing $K_c$ from 1 to 2 can reduce the access complexity in half, while simultaneously doubling the storage efficiency.

*5) Tradeoff between Upload and Download Costs:* The trade-off between the upload cost and the download cost of the ACSA-RW scheme is illustrated via two examples in Figure 3, where we have $N = 10, X_\Delta = T = 1$ for the blue solid curve, and $N = 10, X_\Delta = 1, T = 2$ for the red solid curve. For both examples, we set $K_c = 1$ and assume that there are no dropout servers. The trade-off is achieved with various choices of $X$. For the example shown in the blue solid curve, we set $X = (2, 3, 4, 5, 6, 7, 8)$. For the example in the red solid curve, we set $X = (3, 4, 5, 6, 7)$. Note that the most balanced trade-off point is achieved when $X = N/2$.

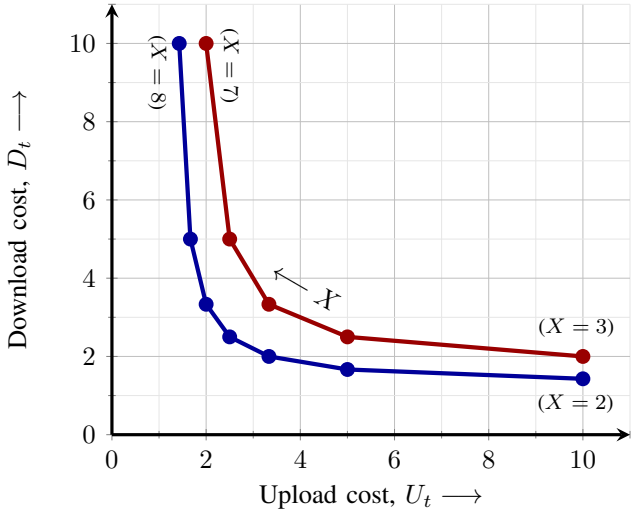



Fig. 3: Upload, download costs pairs $(U_t, D_t)$ of the ACSA-RW scheme in the asymptotic setting $L \gg K \gg 1$, for $N = 10, X_\Delta = T = 1$ (the blue curve) and $N = 10, X_\Delta = 1, T = 2$ (the red curve), with various choices of $X$. Both examples assume that $K_c = 1$ and there are no dropout servers.

*6) Synergistic Gains from Joint Design of Private Read and Write:* A notable aspect of the ACSA-RW scheme, which answers in the affirmative an open question raised in Section 4.4.4 of [2] for the case of partially-colluding servers, is the synergistic gain from the joint design of the read phase and the write phase. While the details of the scheme are non-trivial and can be found in Section IV, let us provide an intuitive explanation here by ignoring some of the details. As a simplification, let us ignore security constraints and consider the database represented by a vector $\mathbf{W} = [W_1, W_2, \cdots, W_K]^\mathsf{T}$ that consists of symbols from the $K$ submodels. Suppose the user is interested in $W_{\theta_t}$ for some index $\theta_t \in [K]$ that must be kept private. Note that $W_{\theta_t} = \mathbf{W}^\mathsf{T}\mathbf{e}_K(\theta_t)$, where $\mathbf{e}_K(\theta_t)$ is the standard basis vector, i.e., the desired symbol is obtained as an inner product of the database vector and the basis vector $\mathbf{e}_K(\theta_t)$. To do this privately, the basis vector is treated by the user as a secret and a linear threshold secret-sharing scheme is used to generate shares that are sent to the servers. The servers return the inner products of the stored data and the secret-shared basis vector, which effectively form the secret shares of the desired inner product. Once the user collects sufficiently many secret shares, (s)he is able to retrieve the desired inner product, and therefore the desired symbol $W_{\theta_t}$. This is the key to the private read-operation. Now during the write phase, the user wishes to update the database to the new state: $\mathbf{W}' = [W_1, \cdots, W_{\theta_t-1}, W_{\theta_t} + \Delta_t, W_{\theta_t+1} \cdots, W_K]$, which can be expressed as $\mathbf{W}' = \mathbf{W} + \Delta_t \mathbf{e}_K(\theta_t)$. This can be accomplished by sending the secret-shares of $\Delta_t \mathbf{e}_K(\theta_t)$ to the $N$ servers. The key observation here is the following: since the servers (those that were available during the read phase) have already received secret shares of $\mathbf{e}_K(\theta_t)$, the cost of sending the secret-shares of $\Delta_t \mathbf{e}_K(\theta_t)$ is significantly reduced. Essentially, it suffices to send secret shares of $\Delta_t$ which can be multiplied with the secret shares of $\mathbf{e}_K(\theta_t)$ to generate secret shares of $\Delta_t \mathbf{e}_K(\theta_t)$ at the servers. This is much more efficient because $\Delta_t \mathbf{e}_K(\theta_t)$ is a $K \times 1$ vector, while the dimension of $\Delta_t$ is 1 (scalar), and $K \gg 1$. This is the intuition behind the synergistic gains from the joint design of private read and write operations that are exploited by ACSA-RW. Note that the servers operate directly on secret shares, as in homomorphic encryption [62], and that these operations (inner products and scalar multiplications) are special cases of secure distributed *batch* matrix multiplications. The reason for the need of *batch* matrix multiplication in FSL is that each submodel consists of multiple symbols, and as such multiple symbols, instead of just one, are retrieved/updated during the read/write cycle of FSL. Since CSA codes, which take advantage of *batch* processing and vectorized codewords to gain additional communication efficiencies, have been shown to be natural solutions for secure distributed batch matrix multiplications [50], [58], it is intuitively to be expected that CSA schemes should lead to communication-efficient solutions for the private read-write implementation as described above.

*7) How to Fully Update a Distributed Database that is only Partially Accessible:* A seemingly paradoxical aspect of the write phase of ACSA-RW is that it is able to force the distributed database across all $N$ servers to be fully consistent with the updated data, even though the database is only partially accessible due to write-dropout servers. Let us explain the intuition behind this with a toy example[10] where we have $N = 2$ servers and $X = 1$ security level is required. For simplicity we have only one file/submodel, i.e., $K = 1$, so there are no privacy concerns. The storage at the two servers is $S_1 = W + Z$, $S_2 = Z$, where $W$ is the data (submodel) symbol, and $Z$ is the random noise symbol used to guarantee the $X = 1$ security level, i.e., the storage at each server individually reveals nothing about the data $W$. The storage can be expressed in the following form.

$$\begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} W \\ Z \end{bmatrix}, \tag{18}$$

[10]The toy example is not strictly a special case of the ACSA-RW scheme, because the number of servers $N = 2$ is too small to guarantee any privacy. However, the example serves to demonstrate the key idea.

so that $\mathbf{G}$ is the coding function, and the data can be recovered as $S_1 - S_2 = W$.

Now suppose the data $W$ needs to be updated to the new value $W' = W + \Delta$. The updated storage $S_1', S_2'$ should be such that the coding function is unchanged (still the same $\mathbf{G}$ matrix) and the updated data can similarly be recovered as $S_1' - S_2' = W'$. Remarkably this can be done even if one of the two servers drops out and is therefore inaccessible. For example, if Server 1 drops out, then we can update the storage only at Server 2 to end up with $S_1' = S_1 = W + Z$, and $S_2' = Z - \Delta$, such that indeed $S_1' - S_2' = W'$ and the coding function is unchanged.

$$\begin{bmatrix} S_1' \\ S_2' \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W + \Delta \\ Z - \Delta \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W' \\ Z' \end{bmatrix}. \quad (19)$$

Similarly, if Server 2 drops out, then we can update the storage only at Server 1 to end up with $S_1' = W + \Delta + Z$, and $S_2' = S_2 = Z$, such that we still have $S_1' - S_2' = W'$ and the coding function is unchanged.

$$\begin{bmatrix} S_1' \\ S_2' \end{bmatrix} = \begin{bmatrix} S_1' \\ S_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W + \Delta \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W' \\ Z \end{bmatrix}. \quad (20)$$

This, intuitively, is how the paradox is resolved, and a distributed database is fully updated even when it is only partially accessible. Note that the realization of the "noise" symbol is different for various realizations of the dropout servers, $Z' \neq Z$.

While the toy example conveys a key idea, the generalization of this idea to the ACSA-RW scheme is rather non-trivial. Let us shed some light on this generalization, which is made possible by the construction of an "*ACSA null-shaper*", see Definition 8. Specifically, by carefully placing nulls of the CSA code polynomial in the update equation, the storage of the write-dropout servers in $\mathcal{S}_w^{(t)}$ is left unmodified. It is important to point out that the storage structure (i.e., ACSA storage, see Definition 3) is preserved, just as the coding function is left unchanged in the toy example above. Let us demonstrate the idea with a minimal example where $X = 2, T = 1, X_\Delta = 0, N = 4$. Let us define the following functions.

$$\mathbf{S}(\alpha) = \mathbf{W} + \alpha \mathbf{Z}_1 + \alpha^2 \mathbf{Z}_2, \quad (21)$$

$$\mathbf{Q}(\alpha) = \mathbf{e}_K(\theta_t) + \alpha \mathbf{Z}', \quad (22)$$

where $\mathbf{W} = [W_1, W_2, \cdots, W_K]$ is a $K \times 1$ vector of the $K$ data (submodel) symbols, $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}'$ are uniformly and independently distributed noise vectors that are used to protect data security and the user's privacy, respectively. Let $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ be 4 distinct non-zero elements from a finite field $\mathbb{F}_q$. The storage at the 4 servers is $\mathbf{S}(\alpha_1)$, $\mathbf{S}(\alpha_2)$, $\mathbf{S}(\alpha_3)$, $\mathbf{S}(\alpha_4)$, respectively. Similarly, the read-queries for the 4 servers are $\mathbf{Q}(\alpha_1)$, $\mathbf{Q}(\alpha_2)$, $\mathbf{Q}(\alpha_3)$, $\mathbf{Q}(\alpha_4)$, respectively. We note that the storage vectors and the query vectors can be viewed as secret sharings of $\mathbf{W}$ and $\mathbf{e}_K(\theta_t)$ vectors with threshold of $X = 2$ and $T = 1$, respectively. Now let us assume that $\mathcal{S}_w^{(t)} = \{1\}$ for some $t \in \mathbb{N}$, i.e., Server 1 drops out. Let us define the function $\Omega(\alpha) = (\alpha_1 - \alpha)/\alpha_1$, which is referred to as *ACSA null-shaper*, and consider the following update equation.

$$\mathbf{S}'(\alpha) = \mathbf{S}(\alpha) + \Omega(\alpha) \Delta_t \mathbf{Q}(\alpha). \quad (23)$$

Inspecting the second term on the RHS, we note that

$$\begin{aligned} &\Omega(\alpha)\Delta_t \mathbf{Q}(\alpha) \\ &= \frac{1}{\alpha_1}(\alpha_1 - \alpha)\Delta_t(\mathbf{e}_K(\theta_t) + \alpha \mathbf{Z}') && (24) \\ &= \frac{1}{\alpha_1}\Delta_t \left(\alpha_1 \mathbf{e}_K(\theta_t) + \alpha(\alpha_1 \mathbf{Z}' - \mathbf{e}_K(\theta_t)) - \alpha^2 \mathbf{Z}'\right) && (25) \\ &= \Delta_t \left(\mathbf{e}_K(\theta_t) + \alpha(\mathbf{Z}' - \alpha_1^{-1}\mathbf{e}_K(\theta_t)) - \alpha^2 \alpha_1^{-1}\mathbf{Z}'\right), && (26) \\ &= \Delta_t \left(\mathbf{e}_K(\theta_t) + \alpha \dot{\mathbf{I}}_1 - \alpha^2 \dot{\mathbf{I}}_2\right), && (27) \end{aligned}$$

where $\dot{\mathbf{I}}_1 = \mathbf{Z}' - \alpha_1^{-1}\mathbf{e}_K(\theta_t)$ and $\dot{\mathbf{I}}_2 = \alpha_1^{-1}\mathbf{Z}'$. Evidently, by the update equation, the user is able to update the symbol of the $\theta_t^{th}$ message with the increment $\Delta_t$, while maintaining the storage as a secret sharing of threshold 2, i.e.,

$$\begin{aligned} &\mathbf{S}'(\alpha) \\ &= (\mathbf{W} + \Delta_t \mathbf{e}_k(\theta_t)) + \alpha(\mathbf{Z}_1 + \Delta_t \dot{\mathbf{I}}_1) + \alpha^2(\mathbf{Z}_2 + \Delta_t \dot{\mathbf{I}}_2). \end{aligned} \quad (28)$$

However, by the definition of ACSA null-shaper, we have $\Omega(\alpha_1) = 0$. Thus $\mathbf{S}'(\alpha_1) = \mathbf{S}(\alpha_1)$, and we do not have to update the storage at the Server 1. In other words, $\dot{\mathbf{I}}_1$ and $\dot{\mathbf{I}}_2$ are *artificially correlated interference symbols* such that the codeword $\Omega(\alpha_1)\mathbf{Q}(\alpha_1)$ is zero, and accordingly, the storage of Server 1 is left unmodified. Note that ACSA null-shaper does not affect the storage structure because $X = 2 > T = 1$. The idea illustrated in this minimal example indeed generalizes to the full ACSA-RW scheme, see Section IV for details.

*8) Comparison with [12]:* Let us compare our ACSA-RW solution with that in [12]. The setting in [12] corresponds to $X = 0, T = 1, X_\Delta = 0$, and $|\mathcal{S}_w^{(t)}| = |\mathcal{S}_r^{(t)}| = 0$ for all $t \in \mathbb{N}$. Note that our ACSA-RW scheme for $X = 1, T = 1, X_\Delta = 0$ and $K_c = 1$ applies to the setting of [12] ($X = 1$ security automatically satisfies $X = 0$ security). To make the comparison more transparent, let us briefly review the construction in [12], where at any time $t, t \in \mathbb{Z}^*$, each of the $N$ servers stores the $K$ submodels in the following coded form.

$$\mathbf{W}_k^{(t-1)} + z_k^{(t)} \mathbf{\Delta}_t, \forall k \in [K]. \quad (29)$$

For all $t \in \mathbb{N}$, $\left(z_k^{(t)}\right)_{k \in [K]}$ are distinct random scalars generated by User $t$ and we set $z_{\theta_t}^{(t)} = 1$. For completeness we define $\mathbf{\Delta}_0 = \mathbf{0}, z_k^{(0)} = 0, \mathbf{W}_k^{(-1)} = \mathbf{W}_k^{(0)}, \forall k \in [K]$. In addition, the $N$ servers store the random scalars $\left(z_k^{(t)}\right)_{k \in [K]}$, as well as the increment $\mathbf{\Delta}_t$ according to a secret sharing scheme of threshold 1. In the retrieval phase, User $t$ retrieves the coded desired submodel $\mathbf{W}_{\theta_t}^{(t-2)} + z_{\theta_t}^{(t-1)} \mathbf{\Delta}_{t-1}$ privately according to a capacity-achieving replicated storage based PIR scheme, e.g., [14]. Besides, User $t$ also downloads the secret shared random scalars $\left(z_k^{(t-1)}\right)_{k \in [K]}$ and the increment $\mathbf{\Delta}_{t-1}$ to correctly recover the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$. In the update phase, User $t$ uploads to each of the $N$ servers the following update vectors $\mathbf{P}_k^{(t)}$ for all $k \in [K]$.

$$\mathbf{P}_k^{(t)} = \begin{cases} z_k^{(t)} \mathbf{\Delta}_t - z_k^{(t-1)} \mathbf{\Delta}_{t-1}, & k \neq \theta_{t-1}, \\ z_k^{(t)} \mathbf{\Delta}_t, & k = \theta_{t-1}. \end{cases} \quad (30)$$

Also, User $t$ uploads the secret shared random scalars $\left(z_k^{(t)}\right)_{k\in[K]}$ and the increment $\mathbf{\Delta}_t$ to the $N$ servers. To perform an update, each of the $N$ servers updates all of the submodels $k \in [K]$ according to the following equation.

$$\left(\mathbf{W}_k^{(t-2)} + z_k^{(t-1)}\mathbf{\Delta}_{t-1}\right) + \mathbf{P}_k^{(t)} = \mathbf{W}_k^{(t-1)} + z_k^{(t)}\mathbf{\Delta}_t. \quad (31)$$

Perhaps the most significant difference between our ACSA-RW scheme and the construction in [12] is that the latter does not guarantee the privacy of successive updates, i.e., by monitoring the storage at multiple time slots, the servers are eventually able to learn about the submodel indices from past updates[11]. On the other hand, our construction guarantees information theoretic privacy for an unlimited number of updates, without extra storage overhead. Furthermore, we note that the normalized download cost achieved by the construction in [12] cannot be less than 2, whereas ACSA-RW achieves download cost of less than 2 with large enough $N$. For the asymptotic setting $L/K \to \infty$, the upload cost[12] achieved by [12] is at least $2N+1$, while ACSA-RW achieves the upload cost of at most $N$. The lower bound of upload cost of XSTPFSL is characterized in [12] as $NK$. However, our construction of ACSA-RW shows that it is possible to do better.[13] In particular, for the asymptotic setting $K \to \infty, L/K \to \infty$, the upload cost of less than $N$ is achievable by the ACSA-RW scheme.

*9) Comparison with [52]:* Let us also briefly review the 4-server information-theoretic DORAM construction in [52] to see how our ACSA-RW scheme improves upon it. Note that the setting considered in [52] is a special case of our problem where $X = 1, T = 1, X_\Delta = 0, K_c = 1$ and $|\mathcal{S}_w^{(t)}| = |\mathcal{S}_r^{(t)}| = 0$ for all $t \in \mathbb{N}$. First, we note that in the asymptotic setting $L/K \to \infty$, the upload cost achieved by the ACSA-RW is the same as that in [52]. Therefore, for this comparison we focus on the read phase and the download cost. Specifically, the information-theoretic DORAM construction in [52] partitions the four servers into two groups, each of which consists of 2 servers. For the retrieval phase, the first group emulates a 2-server PIR, storing the $K$ submodels secured with additive random noise, i.e., $\mathbf{W}+\mathbf{Z}$. The second group emulates another 2-server PIR storing the random noise $\mathbf{Z}$. To retrieve the desired submodel privately, the user exploits a PIR scheme to retrieve the desired secured submodel, as well as the corresponding random noise. Therefore, with capacity-achieving PIR schemes, the download cost is 4 (for large $K$). On the other hand, our ACSA-RW scheme avoids the partitioning of the servers and improves the download cost by jointly exploiting all 4 servers. Remarkably, with the idea of cross-subspace alignment, out of the 4 downloaded symbols, the interference symbols align within 2 dimensions, leaving 2 dimensions interference-free for the desired symbols, and consequently, the asymptotically optimal download cost of 2 is achievable. Lastly, the ACSA-RW scheme also generalizes efficiently to arbitrary numbers of servers.

*10) On the Assumption $L \gg K \gg 1$ for FSL:* Finally, let us briefly explore the practical relevance of the asymptotic limits $K \to \infty, L/K \to \infty$, with an example. Suppose we have $N = 6$ distributed servers, we require security and privacy levels of $X_\Delta = T = 1, X = 3$. Let us set $K_c = 1$, and we operate over $\mathbb{F}_8$. Consider an e-commerce recommendation application similar to what is studied in [5], where a global model with a total of $3,500,000$ symbols (from $\mathbb{F}_8$) is partitioned into $K = 50$ submodels. Each of the submodels comprises $L = 70,000$ symbols. Note that $L \gg K \gg 1$. Let us assume that there are no dropout servers for some $t \in \mathbb{N}$. Now according to Lemma 2, the normalized upload cost achieved by the ACSA-RW scheme is $U_t = \frac{6\times 35000+6\times 2\times 50}{70{,}000} \approx 3.00857$. On the other hand, the normalized download cost achieved is $D_t = 6/2 = 3$. Evidently, the asymptotic limits $K \to \infty, L/K \to \infty$ are fairly accurate for this non-asymptotic setting. For this particular example, the upload cost is increased by only $0.29\%$ compared to the asymptotic limit. On the other hand, the download cost is increased by only $1.4 \times 10^{-22}\%$ compared to the lower bound (evaluated for $K = 50$) from [30].

## IV. Proof of Theorem 1

### A. Motivating Example: $N = 9, K_c = 2, X = 4, T = 1, X_\Delta = 1$

To make the presentation of the general scheme (in Section IV-B) more accessible, in this section let us consider a motivating example where $N = 9, K_c = 2, X = 4, T = 1, X_\Delta = 1$. In particular, following the observation in Section III-A1, the total server storage space of the $N = 9$ servers is represented as an $N = 9$ dimensional space, where $K_c = 2$ dimensions are occupied with model data, $X = 4$ dimensions are occupied with noise that is independent of the model data, and accordingly, the remaining $N - K_c - X = 3$ dimensions represent the storage-redundancy that will be exploited by the private-read phase of the ACSA-RW scheme. Within these 3 dimensions, $T - 1 = 0$ dimensions are required to enable $T = 1$ private-read, thus leaving us 3 dimensions for elastic read-dropout resilience, i.e., the read-dropout threshold is 3. On the other hand, private-write functionality leverages the $X = 4$ dimensions where $X_\Delta$ and $T$-privacy constraints requires a total of $X_\Delta + T - 1 = 1$ dimensions, and the remaining $X - 1 = 3$ dimensions determine the write-dropout threshold. The conceptual partitioning of the total server storage space is illustrated in Figure 4.

With these conceptual ideas in mind, let us construct the ACSA-RW scheme for this setting step by step. Let us set $L = 12$, i.e., each submodel consists of $L = 12$ symbols

[11] This is because the update vectors (30) for the $K$ submodels at any time $t$ can be viewed as $\mathcal{P}_t = \text{span}\{\mathbf{\Delta}_t, \mathbf{\Delta}_{t-1}\}$. For any two consecutive time slots $t$ and $t+1$, it is possible to determine $\text{span}\{\mathbf{\Delta}_t\} = \text{span}\{\mathbf{\Delta}_t, \mathbf{\Delta}_{t-1}\} \cap \text{span}\{\mathbf{\Delta}_{t+1}, \mathbf{\Delta}_t\} = \mathcal{P}_t \cap \mathcal{P}_{t+1}$. Due to the fact that for User $t$, the update vector for the $\theta_{t-1}^{th}$ submodel only lies in $\text{span}\{\mathbf{\Delta}_t\}$, any curious server is able to obtain information about $\theta_{t-1}$ from $\mathcal{P}_t$ if $\mathbf{\Delta}_t$ is linearly independent of $\mathbf{\Delta}_{t-1}$.

[12] In [12] the achieved upload cost is $NLK + L + K$. However, in terms of average upload compression, e.g., by entropy encoding, allows lower upload cost. For example, in the asymptotic setting $L/K \to \infty$, the upload cost of $(2N + 1 + 1/(N-1))$ may be achievable.

[13] It is assumed in [12] that for the desired submodel, the user uploads $L$ symbols for the update, while for other submodels, the user should also upload $(K-1)L$ symbols to guarantee the privacy. However, it turns out that the uploaded symbols for the update of the desired submodel and the symbols for the purpose of guaranteeing the privacy do not have to be independent.

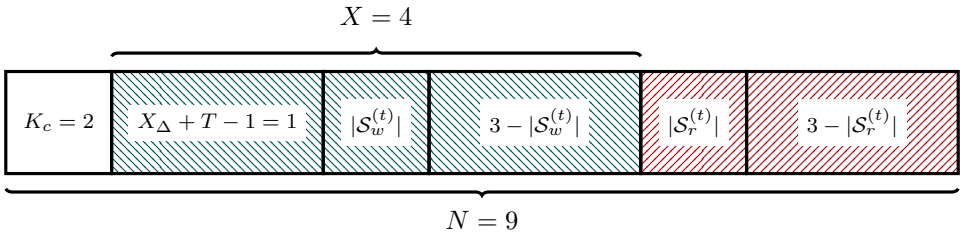



Fig. 4: Conceptual partitioning of total server storage space for the motivating example.

from the finite field $\mathbb{F}_q$. The reason for choosing $L = 12$ is explained as follows. By the construction of our ACSA-RW scheme, private-read and private-write cycles can be viewed as consisting of several sub-operations where in each sub-operation, $3 - |\mathcal{S}_r^{(t)}|$ and $3 - |\mathcal{S}_w^{(t)}|$ symbols of the desired submodel are retrieved and updated, respectively. Since $(3 - |\mathcal{S}_r^{(t)}|) \in \{1, 2, 3\}, (3 - |\mathcal{S}_w^{(t)}|) \in \{1, 2, 3\}$, the size of the submodels must be multiples of $\text{lcm}(\{1, 2, 3\}) = 6$ to guarantee the functionality of the sub-operations under all circumstances of read-dropouts and write-dropouts. Besides, since we set $K_c = 2$ in the motivating setting, by our ACSA-RW scheme, the number of $q$-ary symbols in each submodel is set to be $L = 2 \times 6 = 12$. Note that we choose $L = 12$ mainly for simplicity. In our general scheme, the size of the submodels can be any multiple of 12, thus it can be arbitrarily large. Let $(\alpha_1, \alpha_2, \cdots, \alpha_9), (f_1, f_2, f_3)$ be a total of 12 distinct elements from the finite field $\mathbb{F}_q, q \geq 12$. For all $t \in \mathbb{Z}^*, j \in [6], k \in [K], i \in [2]$, let us define

$$W_k^{(t)}(j, i) = W_k^{(t)}(i + 2(j - 1)), \tag{32}$$

i.e., the $L = 12$ symbols of each of the $K$ messages are reshaped into a $6 \times 2$ matrix. Similarly, for all $t \in \mathbb{N}, j \in [6], i \in [2]$, we define

$$\Delta_{j,i}^{(t)} = \Delta_t(i + 2(j - 1)). \tag{33}$$

For all $t \in \mathbb{Z}^*, j \in [6], i \in [2]$, let us define the following vectors.

$$\dot{\mathbf{W}}_{j,i}^{(t)} = \left[W_1^{(t)}(j, i), W_2^{(t)}(j, i), \cdots, W_K^{(t)}(j, i)\right]^\mathsf{T}. \tag{34}$$

In other words, for all $t \in \mathbb{Z}^*, j \in [6], i \in [2]$, the vector $\dot{\mathbf{W}}_{j,i}^{(t)}$ consists of the $(i+2(j-1))^{th}$ symbol of all submodels at time $t$. Let $\left(\dot{\mathbf{Z}}_{l,x}^{(0)}\right)_{l \in [12], x \in [4]}$ be uniformly i.i.d. column vectors from $\mathbb{F}_q^K$. Now we are ready to construct the initial (i.e., $t = 0$) ACSA storage at the $N = 9$ servers. In particular, the storage at Server $n$, $n \in [9]$, is as follows.

$$\mathbf{S}_n^{(0)} = \begin{bmatrix} \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{1,1}^{(0)} + \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{1,2}^{(0)} + \sum_{x \in [4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{1,x}^{(0)} \\ \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{2,1}^{(0)} + \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{2,2}^{(0)} + \sum_{x \in [4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{2,x}^{(0)} \\ \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{3,1}^{(0)} + \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{3,2}^{(0)} + \sum_{x \in [4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{3,x}^{(0)} \\ \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{4,1}^{(0)} + \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{4,2}^{(0)} + \sum_{x \in [4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{4,x}^{(0)} \\ \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{5,1}^{(0)} + \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{5,2}^{(0)} + \sum_{x \in [4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{5,x}^{(0)} \\ \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{6,1}^{(0)} + \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{6,2}^{(0)} + \sum_{x \in [4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{6,x}^{(0)} \end{bmatrix}. \tag{35}$$

Note that in each row of (35), $K_c = 2$ model data terms as well as $X = 4$ i.i.d. uniform random noise terms are coded according to the Cauchy-Vandermonde structure [58], where the Cauchy terms carry data symbols and Vandermonde terms carry noise terms. This guarantees $X = 4$ secure storage because of the MDS coded noise terms. Note that in each row, the constants $f_*$ in the denominator of Cauchy terms are distinct so that these terms are linearly independent. Now, consider the first term (and the second term) in each row of (35). If viewed column-wise, the constants $f_*$ in the denominator of Cauchy terms are distinct for at most any 3 consecutive rows. This fact is important for elastic dropout resilience in private-read and private-write mechanisms because our ACSA-RW private-read and private-write functionality consists of a series of sub-operations, and in each sub-operation, several (at most 3 in this setting) symbols in the column-wise direction (e.g., $W_{1,1}^{(t)}, W_{2,1}^{(t)}$ and $W_{3,1}^{(t)}$) of the desired submodel are retrieved/updated. This fact guarantees that the desired submodel symbols to be retrieved/updated are linearly separable due to distinct $f_*$ constants.

Now let us assume that User 1 experiences $|\mathcal{S}_r^{(1)}| = 1, |\mathcal{S}_w^{(1)}| = 1$, i.e., there is 1 dropout server in the read phase and 1 dropout server in the write phase. Note that the two sets can be arbitrarily realized. Let us denote $\overline{\mathcal{S}}_r^{(1)} = [N] \setminus \mathcal{S}_r^{(1)}$, $\overline{\mathcal{S}}_w^{(1)} = [N] \setminus \mathcal{S}_w^{(1)}$. Now, let us construct the private-read scheme to retrieve the $L$ symbols of the desired submodel $\theta_1$. To this end, User 1 generates uniformly i.i.d. column vectors $\left(\widetilde{\mathbf{Z}}_{i,1}^{(1)}\right)_{i \in [3]}$ from $\mathbb{F}_q^K$ and sends the ACSA-RW query to the $n^{th}$ server, $n \in \overline{\mathcal{S}}_r^{(1)}$, as follows.

$$Q_n^{(1,\theta_1)} = \left( \underbrace{\begin{bmatrix} \mathbf{e}_K(\theta_1) + (\alpha_n - f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)} \end{bmatrix}}_{\mathbf{Q}_{n,1}^{(1,\theta_1)}}, \underbrace{\begin{bmatrix} \mathbf{e}_K(\theta_1) + (\alpha_n - f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)} \\ \mathbf{e}_K(\theta_1) + (\alpha_n - f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)} \end{bmatrix}}_{\mathbf{Q}_{n,2}^{(1,\theta_1)}} \right). \tag{36}$$

Note that by the cyclic structure of $Q_n^{(1,\theta_1)}$, it suffices

to upload $\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_i)\widetilde{\mathbf{Z}}_{i,1}^{(1)}\right)_{i\in[3]}$ to the $n^{th}$ server, $n\in\overline{\mathcal{S}}_r^{(1)}$. This fact is important in terms of reducing the upload cost. Since the standard basis vector containing the information of the desired index $\mathbf{e}_K(\theta_1)$ is protected by a uniformly i.i.d. random noise vector, $T=1$ privacy is guaranteed. Upon receiving the query sent by the user, the $n^{th}$ server, $n\in\overline{\mathcal{S}}_r^{(1)}$ constructs the following 6 vectors according to the query, denoted as $\widehat{\mathbf{Q}}_{n,1}^{(1,\theta_1)},\widehat{\mathbf{Q}}_{n,2}^{(1,\theta_1)},\cdots,\widehat{\mathbf{Q}}_{n,6}^{(1,\theta_1)}$ respectively.

$$
\begin{bmatrix}
\frac{\alpha_n-f_3}{f_1-f_3}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right)\\
\frac{\alpha_n-f_1}{f_2-f_1}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right)\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}
\end{bmatrix},
\begin{bmatrix}
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\frac{\alpha_n-f_2}{f_3-f_2}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right)\\
\frac{\alpha_n-f_3}{f_1-f_3}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right)\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}
\end{bmatrix},
\begin{bmatrix}
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\frac{\alpha_n-f_1}{f_2-f_1}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right)\\
\frac{\alpha_n-f_2}{f_3-f_2}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right)
\end{bmatrix},
\begin{bmatrix}
\frac{\alpha_n-f_1}{f_3-f_1}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right)\\
\frac{\alpha_n-f_2}{f_1-f_2}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right)\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}
\end{bmatrix},
\begin{bmatrix}
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\frac{\alpha_n-f_3}{f_2-f_3}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right)\\
\frac{\alpha_n-f_1}{f_3-f_1}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right)\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}
\end{bmatrix},
\begin{bmatrix}
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\mathbf{0}_{K\times1}\\
\frac{\alpha_n-f_2}{f_1-f_2}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right)\\
\frac{\alpha_n-f_3}{f_2-f_3}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right)
\end{bmatrix},
\tag{37}
$$

where $\mathbf{0}_{K\times1}$ is $K\times1$ zero vector. As mentioned before, in our ACSA-RW scheme, the private-read functionality consists of a total of 6 sub-operations, in each sub-operation, 2 symbols of the desired submodel are retrieved. Indeed, the 6 vectors $\widehat{\mathbf{Q}}_{n,1}^{(1,\theta_1)},\widehat{\mathbf{Q}}_{n,2}^{(1,\theta_1)},\cdots,\widehat{\mathbf{Q}}_{n,6}^{(1,\theta_1)}$ will be used in each of the sub-operations, and they can be viewed as scaled version of the query sent by the user. Such scaling, in our formal presentation of the general scheme, is made possible by the element named *ACSA-Packer*, as defined in Definition 6. Now the $n^{th}$ server, $n\in\overline{\mathcal{S}}_r^{(1)}$, responds to the user with the answer defined as $\left(\left(\mathbf{S}_n^{(0)}\right)^{\mathsf{T}}\widehat{\mathbf{Q}}_{n,i}^{(1,\theta_1)}\right)_{i\in[6]}$. To see the correctness of the private-read scheme, for example, let us consider $\left(\mathbf{S}_n^{(0)}\right)^{\mathsf{T}}\widehat{\mathbf{Q}}_{n,1}^{(1,\theta_1)}$.

$$
\begin{aligned}
&\left(\mathbf{S}_n^{(0)}\right)^{\mathsf{T}}\widehat{\mathbf{Q}}_{n,1}^{(1,\theta_1)}\\
&=\frac{\alpha_n-f_3}{f_1-f_3}\left(\frac{1}{\alpha_n-f_1}\dot{\mathbf{W}}_{1,1}^{(0)}+\frac{1}{\alpha_n-f_3}\dot{\mathbf{W}}_{1,2}^{(0)}\right.\\
&\qquad\left.+\sum_{x\in[4]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{1,x}^{(0)}\right)^{\mathsf{T}}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right)\\
&+\frac{\alpha_n-f_1}{f_2-f_1}\left(\frac{1}{\alpha_n-f_2}\dot{\mathbf{W}}_{2,1}^{(0)}+\frac{1}{\alpha_n-f_1}\dot{\mathbf{W}}_{2,2}^{(0)}\right.\\
&\qquad\left.+\sum_{x\in[4]}\alpha_n^{x-1}\dot{\mathbf{Z}}_{2,x}^{(0)}\right)^{\mathsf{T}}\left(\mathbf{e}_K(\theta_1)+(\alpha_n-f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right)
\end{aligned}
\tag{38}
$$

$$
\begin{aligned}
&=\frac{1}{\alpha_n-f_1}\left(\dot{\mathbf{W}}_{1,1}^{(0)}\right)^{\mathsf{T}}\mathbf{e}_K(\theta_1)+\frac{1}{\alpha_n-f_2}\left(\dot{\mathbf{W}}_{2,1}^{(0)}\right)^{\mathsf{T}}\mathbf{e}_K(\theta_1)\\
&\quad+\sum_{m\in[6]}\alpha_n^{m-1}\ddot{I}_{1,m}^{(1)}
\end{aligned}
\tag{39}
$$

$$
\begin{aligned}
&=\frac{1}{\alpha_n-f_1}W_{\theta_1}^{(0)}(1,1)+\frac{1}{\alpha_n-f_2}W_{\theta_1}^{(0)}(2,1)\\
&\quad+\sum_{m\in[6]}\alpha_n^{m-1}\ddot{I}_{1,m}^{(1)},
\end{aligned}
\tag{40}
$$

where $\left(\ddot{I}_{1,m}^{(1)}\right)_{m\in[6]}$ are various interference symbols, whose exact forms are not relevant. Note that the terms in (38) are viewed as rational functions with respect to $\alpha_n$. Thus (39) follows from some manipulations on the rational functions and the fact that the denominators of the terms $\frac{\alpha_n-f_3}{f_1-f_3}$ and $\frac{\alpha_n-f_1}{f_2-f_1}$ are used for normalization, which are the remainders of the polynomial division $(\alpha_n-f_3)/(\alpha_n-f_1)$ and $(\alpha_n-f_1)/(\alpha_n-f_2)$, respectively. Note that $\left(\left(\mathbf{S}_n^{(0)}\right)^{\mathsf{T}}\widehat{\mathbf{Q}}_{n,1}^{(1,\theta_1)}\right)_{n\in\overline{\mathcal{S}}_r^{(1)}}$ can be viewed as a linear system with a Cauchy-Vandermonde structure, therefore, the desired symbols $W_{\theta_1}^{(0)}(1,1)$ and $W_{\theta_1}^{(0)}(2,1)$ can be recovered by inverting the following decoding matrix $\mathbf{C}$ [63].

$$
\mathbf{C}=\begin{bmatrix}
\frac{1}{f_1-\alpha_{\overline{\mathcal{S}}_r^{(1)}(1)}} & \frac{1}{f_2-\alpha_{\overline{\mathcal{S}}_r^{(1)}(1)}} & 1 & \cdots & \alpha^5_{\overline{\mathcal{S}}_r^{(1)}(1)}\\
\frac{1}{f_1-\alpha_{\overline{\mathcal{S}}_r^{(1)}(2)}} & \frac{1}{f_2-\alpha_{\overline{\mathcal{S}}_r^{(1)}(2)}} & 1 & \cdots & \alpha^5_{\overline{\mathcal{S}}_r^{(1)}(2)}\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
\frac{1}{f_1-\alpha_{\overline{\mathcal{S}}_r^{(1)}(8)}} & \frac{1}{f_2-\alpha_{\overline{\mathcal{S}}_r^{(1)}(8)}} & 1 & \cdots & \alpha^5_{\overline{\mathcal{S}}_r^{(1)}(8)}
\end{bmatrix},
\tag{41}
$$

where $\left\{\overline{\mathcal{S}}_r^{(1)}(1),\overline{\mathcal{S}}_r^{(1)}(2),\cdots,\overline{\mathcal{S}}_r^{(1)}(8)\right\}=\overline{\mathcal{S}}_r^{(1)}$. Following a similar argument, it can be readily checked that other symbols

of the desired submodel are recoverable by the user. This completes the private-read cycle at $t = 1$.

After User 1 completes the local training of the submodel and generates the increment $\mathbf{\Delta}_1$, (s)he updates the desired submodel according to the private-write mechanism of our ACSA-RW scheme. Specifically, the user generate i.i.d. uniform symbols $\left(\ddot{Z}_{i,1}^{(1)}\right)_{i\in[6]}$ from the finite field $\mathbb{F}_q$. For all $n \in \overline{\mathcal{S}}_w^{(1)}$, the user encodes the $L = 12$ symbols of the increment as follows.

$$\widetilde{\Delta}_1^{(1)} = \frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)} + \frac{1}{\alpha_n - f_2}\Delta_{2,1}^{(1)} + \ddot{Z}_{1,1}^{(1)}, \tag{42}$$

$$\widetilde{\Delta}_2^{(1)} = \frac{1}{\alpha_n - f_3}\Delta_{3,1}^{(1)} + \frac{1}{\alpha_n - f_1}\Delta_{4,1}^{(1)} + \ddot{Z}_{2,1}^{(1)}, \tag{43}$$

$$\widetilde{\Delta}_3^{(1)} = \frac{1}{\alpha_n - f_2}\Delta_{5,1}^{(1)} + \frac{1}{\alpha_n - f_3}\Delta_{6,1}^{(1)} + \ddot{Z}_{3,1}^{(1)}, \tag{44}$$

$$\widetilde{\Delta}_4^{(1)} = \frac{1}{\alpha_n - f_3}\Delta_{1,2}^{(1)} + \frac{1}{\alpha_n - f_1}\Delta_{2,2}^{(1)} + \ddot{Z}_{4,1}^{(1)}, \tag{45}$$

$$\widetilde{\Delta}_5^{(1)} = \frac{1}{\alpha_n - f_2}\Delta_{3,2}^{(1)} + \frac{1}{\alpha_n - f_3}\Delta_{4,2}^{(1)} + \ddot{Z}_{5,1}^{(1)}, \tag{46}$$

$$\widetilde{\Delta}_6^{(1)} = \frac{1}{\alpha_n - f_1}\Delta_{5,2}^{(1)} + \frac{1}{\alpha_n - f_2}\Delta_{6,2}^{(1)} + \ddot{Z}_{6,1}^{(1)}, \tag{47}$$

and $P_n^{(1,\theta_1)} = \big(\mathrm{diag}(\widetilde{\Delta}_1^{(1)}\mathbf{I}_{2K}, \widetilde{\Delta}_2^{(1)}\mathbf{I}_{2K}, \widetilde{\Delta}_3^{(1)}\mathbf{I}_{2K}),$ $\mathrm{diag}(\widetilde{\Delta}_4^{(1)}\mathbf{I}_{2K}, \widetilde{\Delta}_5^{(1)}\mathbf{I}_{2K}, \widetilde{\Delta}_6^{(1)}\mathbf{I}_{2K})\big)$ is sent to the $n^{th}$ server. $X_\Delta = 1$ security is guaranteed because of the uniformly i.i.d. random noise terms. Note that it suffices to upload $\left(\widetilde{\Delta}_i^{(1)}\right)_{i\in[6]}$ to the $n^{th}$ server, $n \in \overline{\mathcal{S}}_w^{(1)}$. The construction of $P_n^{(1,\theta_1)}$ can be done by server-side processing. Once the encoded ACSA-RW increment is available at the servers $n, n \in \overline{\mathcal{S}}_w^{(1)}$, they construct the following two diagonal matrices.

$$\begin{aligned}\widehat{\mathbf{P}}_{n,1}^{(1,\theta_1)} = \mathrm{diag}\Bigg(&\frac{\alpha_n - f_2}{f_1 - f_2}\widetilde{\Delta}_1^{(1)}\mathbf{I}_K, \frac{\alpha_n - f_1}{f_2 - f_1}\widetilde{\Delta}_1^{(1)}\mathbf{I}_K,\\ &\frac{\alpha_n - f_1}{f_3 - f_1}\widetilde{\Delta}_2^{(1)}\mathbf{I}_K, \frac{\alpha_n - f_3}{f_1 - f_3}\widetilde{\Delta}_2^{(1)}\mathbf{I}_K,\\ &\frac{\alpha_n - f_3}{f_2 - f_3}\widetilde{\Delta}_3^{(1)}\mathbf{I}_K, \frac{\alpha_n - f_2}{f_3 - f_2}\widetilde{\Delta}_3^{(1)}\mathbf{I}_K\Bigg),\end{aligned} \tag{48}$$

$$\begin{aligned}\widehat{\mathbf{P}}_{n,2}^{(1,\theta_1)} = \mathrm{diag}\Bigg(&\frac{\alpha_n - f_1}{f_3 - f_1}\widetilde{\Delta}_4^{(1)}\mathbf{I}_K, \frac{\alpha_n - f_3}{f_1 - f_3}\widetilde{\Delta}_4^{(1)}\mathbf{I}_K,\\ &\frac{\alpha_n - f_3}{f_2 - f_3}\widetilde{\Delta}_5^{(1)}\mathbf{I}_K, \frac{\alpha_n - f_2}{f_3 - f_2}\widetilde{\Delta}_5^{(1)}\mathbf{I}_K,\\ &\frac{\alpha_n - f_2}{f_1 - f_2}\widetilde{\Delta}_6^{(1)}\mathbf{I}_K, \frac{\alpha_n - f_1}{f_2 - f_1}\widetilde{\Delta}_6^{(1)}\mathbf{I}_K\Bigg).\end{aligned} \tag{49}$$

Recall that the private-write consists of a total of 6 sub-operations, and in each sub-operation, 2 symbols of the desired submodel are updated, which correspond to the 6 encoded increment symbols $\left(\widetilde{\Delta}_i^{(1)}\right)_{i\in[6]}$. To ensure that the structure of the encoded increment symbols is in accordance with the storage structure, $\left(\widehat{\mathbf{P}}_{n,1}^{(1,\theta_1)}, \widehat{\mathbf{P}}_{n,2}^{(1,\theta_1)}\right)$ are constructed, which can be viewed as scaled versions of $P_n^{(1,\theta_1)}$. In our formal presentation of the general scheme, this scaling is done by the element named *ACSA Unpacker*, as defined in Definition 7. To understand this more clearly, let us consider the term $\frac{\alpha_n - f_2}{f_1 - f_2}\widetilde{\Delta}_1^{(1)}$ as an example.

$$\begin{aligned}&\frac{\alpha_n - f_2}{f_1 - f_2}\widetilde{\Delta}_1^{(1)}\\ &= \frac{\alpha_n - f_2}{f_1 - f_2}\left(\frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)} + \frac{1}{\alpha_n - f_2}\Delta_{2,1}^{(1)} + \ddot{Z}_{1,1}^{(1)}\right) &(50)\\ &= \frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)} + \sum_{m\in[2]}\alpha_n^{i-1}\,\dddot{I}_{1,1,m}^{(1)}, &(51)\end{aligned}$$

where $\dddot{I}_{1,1,1}^{(1)}, \dddot{I}_{1,1,2}^{(1)}$ are various interference symbols, whose exact forms are not important. (51) follows by regarding (50) as a rational function with respect to $\alpha_n$ and noting that the denominator of $\frac{\alpha_n - f_2}{f_1 - f_2}$ is used for normalization. The other terms in $\left(\widehat{\mathbf{P}}_{n,1}^{(1,\theta_1)}, \widehat{\mathbf{P}}_{n,2}^{(1,\theta_1)}\right)$ can be similarly manipulated.

Recall that we assumed that User 1 experiences $|\mathcal{S}_w^{(1)}| = 1$, without loss of generality, let us denote the dropout server as Server $s_0$, i.e., $\mathcal{S}_w^{(1)} = \{s_0\}$. Server $n$, $n \in \overline{\mathcal{S}}_w^{(1)}$, scales $\left(\widehat{\mathbf{P}}_{n,1}^{(1,\theta_1)}, \widehat{\mathbf{P}}_{n,2}^{(1,\theta_1)}\right)$ with the following two diagonal matrices respectively.

$$\begin{aligned}\mathbf{\Omega}_{n,1}^{(1)} = \mathrm{diag}\Bigg(&\frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_2 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_3 - \alpha_{s_0}}\mathbf{I}_K,\\ &\frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_2 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_3 - \alpha_{s_0}}\mathbf{I}_K\Bigg),\end{aligned} \tag{52}$$

$$\begin{aligned}\mathbf{\Omega}_{n,2}^{(1)} = \mathrm{diag}\Bigg(&\frac{\alpha_n - \alpha_{s_0}}{f_3 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_2 - \alpha_{s_0}}\mathbf{I}_K,\\ &\frac{\alpha_n - \alpha_{s_0}}{f_3 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\mathbf{I}_K, \frac{\alpha_n - \alpha_{s_0}}{f_2 - \alpha_{s_0}}\mathbf{I}_K\Bigg).\end{aligned} \tag{53}$$

As described in the observation section, this idea is referred to as *ACSA Null-shaper*, and is formally defined as Definition 8 in the general scheme. To see why storage consistency is preserved, for example, let us consider the term $\frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\frac{\alpha_n - f_2}{f_1 - f_2}\widetilde{\Delta}_1^{(1)}$.

$$\begin{aligned}&\frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\frac{\alpha_n - f_2}{f_1 - f_2}\widetilde{\Delta}_1^{(1)}\\ &= \frac{\alpha_n - \alpha_{s_0}}{f_1 - \alpha_{s_0}}\left(\frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)} + \sum_{m\in[2]}\alpha_n^{i-1}\,\dddot{I}_{1,m}^{(1)}\right) &(54)\\ &= \frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)} + \sum_{m\in[3]}\alpha_n^{m-1}\,\ddddot{I}_{1,1,m}^{(1)}, &(55)\end{aligned}$$

where for all $m \in [3]$, $\ddddot{I}_{1,1,m}^{(1)}$ are various interference symbols. The other terms can be similarly manipulated so that the storage structure is preserved. Now, the $n^{th}$ server, $n \in \overline{\mathcal{S}}_w^{(1)}$, updates its storage according to the following equation.

$$\mathbf{S}_n^{(1)} = \mathbf{S}_n^{(0)} + \widehat{\mathbf{P}}_{n,1}^{(1,\theta_1)}\mathbf{\Omega}_{n,1}^{(1)}\mathbf{Q}_{n,1}^{(1,\theta_1)} + \widehat{\mathbf{P}}_{n,2}^{(1,\theta_1)}\mathbf{\Omega}_{n,2}^{(1)}\mathbf{Q}_{n,2}^{(1,\theta_1)}. \tag{56}$$

To see the correctness of the private-write mechanism of our ACSA-RW scheme, recall that

$$\widehat{\mathbf{P}}_{n,1}^{(1,\theta_1)}\mathbf{\Omega}_{n,1}^{(1)}\mathbf{Q}_{n,1}^{(1,\theta_1)} \tag{57}$$

$$= \begin{bmatrix} \left(\frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)} + \sum_{m\in[3]} \alpha_n^{i-1} \dddot{I}_{1,1,m}^{(1)}\right) \\ \left(\mathbf{e}_K(\theta_1) + (\alpha_n - f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right) \\ \hdashline \left(\frac{1}{\alpha_n - f_2}\Delta_{2,1}^{(1)} + \sum_{m\in[3]} \alpha_n^{i-1} \dddot{I}_{2,1,m}^{(1)}\right) \\ \left(\mathbf{e}_K(\theta_1) + (\alpha_n - f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right) \\ \hdashline \left(\frac{1}{\alpha_n - f_3}\Delta_{3,1}^{(1)} + \sum_{m\in[3]} \alpha_n^{i-1} \dddot{I}_{3,1,m}^{(1)}\right) \\ \left(\mathbf{e}_K(\theta_1) + (\alpha_n - f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right) \\ \hdashline \left(\frac{1}{\alpha_n - f_1}\Delta_{4,1}^{(1)} + \sum_{m\in[3]} \alpha_n^{i-1} \dddot{I}_{4,1,m}^{(1)}\right) \\ \left(\mathbf{e}_K(\theta_1) + (\alpha_n - f_1)\widetilde{\mathbf{Z}}_{1,1}^{(1)}\right) \\ \hdashline \left(\frac{1}{\alpha_n - f_2}\Delta_{5,1}^{(1)} + \sum_{m\in[3]} \alpha_n^{i-1} \dddot{I}_{5,1,m}^{(1)}\right) \\ \left(\mathbf{e}_K(\theta_1) + (\alpha_n - f_2)\widetilde{\mathbf{Z}}_{2,1}^{(1)}\right) \\ \hdashline \left(\frac{1}{\alpha_n - f_3}\Delta_{6,1}^{(1)} + \sum_{m\in[3]} \alpha_n^{i-1} \dddot{I}_{6,1,m}^{(1)}\right) \\ \left(\mathbf{e}_K(\theta_1) + (\alpha_n - f_3)\widetilde{\mathbf{Z}}_{3,1}^{(1)}\right) \end{bmatrix} \tag{58}$$

$$= \begin{bmatrix} \frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{1,1,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{2,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{2,1,m} \\ \frac{1}{\alpha_n - f_3}\Delta_{3,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{3,1,m} \\ \frac{1}{\alpha_n - f_1}\Delta_{4,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{4,1,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{5,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{5,1,m} \\ \frac{1}{\alpha_n - f_3}\Delta_{6,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{6,1,m} \end{bmatrix}, \tag{59}$$

and that

$$\widehat{\mathbf{P}}_{n,2}^{(1,\theta_1)}\mathbf{\Omega}_{n,2}^{(1)}\mathbf{Q}_{n,2}^{(1,\theta_1)} \tag{60}$$

$$= \begin{bmatrix} \frac{1}{\alpha_n - f_3}\Delta_{1,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{1,2,m} \\ \frac{1}{\alpha_n - f_1}\Delta_{2,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{2,2,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{3,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{3,2,m} \\ \frac{1}{\alpha_n - f_3}\Delta_{4,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{4,2,m} \\ \frac{1}{\alpha_n - f_1}\Delta_{5,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{5,2,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{6,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{6,2,m} \end{bmatrix}, \tag{61}$$

where for all $j \in [6], i \in [2], m \in [4]$, $\dot{\mathbf{I}}_{j,i,m}$ are various interference symbols. Therefore, by the update equation, we have (62)–(64) (shown at the top of the next page), which preserves the storage structure and updates the desired submodel correctly. Crucially, by the constructions of $\mathbf{\Omega}_{n,1}^{(1)}$ and $\mathbf{\Omega}_{n,2}^{(1)}$, it is guaranteed that $\mathbf{\Omega}_{s_0,1}^{(1)} = \mathbf{\Omega}_{s_0,2}^{(1)} = \mathbf{0}$, therefore the update equation does not modify the storage at the dropout server $s_0$, which achieves the desired write-dropout resilience functionality. This completes the write cycle at $t = 1$. Since the storage structure is preserved, User 2 may perform the ACSA-RW scheme even though the user is oblivious of the prior history of server dropouts. The general scheme, which is presented in the next section, builds upon the motivating example and can be constructed for general settings and server dropouts.

### B. The General Scheme

For all $t \in \mathbb{N}$, we require that the number of read and write dropout servers is less than the corresponding threshold values, $0 \le |\mathcal{S}_r^{(t)}| < S_r^{\text{thresh}}$ and $0 \le |\mathcal{S}_w^{(t)}| < S_w^{\text{thresh}}$. Since the read and write dropout thresholds cannot be less than 1, we require that $X \ge X_\Delta + T$, and $N \ge K_c + X + T$ for a positive integer $K_c$. Let us define $J = \xi \cdot \text{lcm}\left([S_r^{\text{thresh}}] \cup [S_w^{\text{thresh}}]\right)$ and we set $L = JK_c$, where $\xi$ is a positive[14] integer. In other words, $i \mid J$ for all $i \in [S_r^{\text{thresh}}] \cup [S_w^{\text{thresh}}]$. For ease of reference, let us define $R_r^{(t)} \triangleq S_r^{\text{thresh}} - |\mathcal{S}_r^{(t)}|$, $U_r^{(t)} \triangleq J/R_r^{(t)}$, $R_w^{(t)} \triangleq S_w^{\text{thresh}} - |\mathcal{S}_w^{(t)}|$ and $U_w^{(t)} \triangleq J/R_w^{(t)}$ for all $t \in \mathbb{N}$. Note that it is guaranteed by the choice of $J$ that $U_r^{(t)}, U_w^{(t)}$ are positive integers for all $t \in \mathbb{N}$. Indeed in the ACSA-RW scheme, private read and write operations can be viewed as operations that consist of $K_c U_r^{(t)}$ and $K_c U_w^{(t)}$ sub-operations, and in each sub-operation, $R_r^{(t)}$ and $R_w^{(t)}$ symbols of the desired submodel are retrieved and updated, respectively. The choice of $J$ guarantees that the number of sub-operations is always an integer, regardless of $|\mathcal{S}_r^{(t)}|$ and $|\mathcal{S}_w^{(t)}|$. The parameter $\xi$ guarantees that $L$ is still a free parameter so that $L/K \to \infty$ is well-defined. In other words, $L$ can be any multiple of $K_c \cdot \text{lcm}\left([S_r^{\text{thresh}}] \cup [S_w^{\text{thresh}}]\right)$. Let us define $\mu \triangleq \max(S_r^{\text{thresh}}, S_w^{\text{thresh}})$. We will need a total of $N + \max(\mu, K_c)$ distinct elements from the finite field $\mathbb{F}_q, q \ge N + \max(\mu, K_c)$, denoted as $(\alpha_1, \alpha_2, \cdots, \alpha_N)$, $(\widetilde{f}_1, \widetilde{f}_2, \cdots, \widetilde{f}_{\max(\mu,K_c)})$. Let us define the set $\overline{\mathcal{S}}_w^{(t)} = [N] \setminus \mathcal{S}_w^{(t)}$, and the set $\overline{\mathcal{S}}_r^{(t)} = [N] \setminus \mathcal{S}_r^{(t)}$. For all $t \in \mathbb{Z}^*, j \in [J], k \in [K], i \in [K_c]$, let us define

$$W_k^{(t)}(j,i) = W_k^{(t)}(i + K_c(j-1)), \tag{65}$$

i.e., the $L = JK_c$ symbols of each of the $K$ messages are reshaped into a $J \times K_c$ matrix. Similarly, for all $t \in \mathbb{N}, j \in [J], i \in [K_c]$, we define

$$\Delta_{j,i}^{(t)} = \Delta_t(i + K_c(j-1)). \tag{66}$$

For all $t \in \mathbb{Z}^*, j \in [J], i \in [K_c]$, let us define the following vectors.

$$\dot{\mathbf{W}}_{j,i}^{(t)} = \left[W_1^{(t)}(j,i), W_2^{(t)}(j,i), \cdots, W_K^{(t)}(j,i)\right]^\mathsf{T}. \tag{67}$$

Further, let us set $\mathcal{Z}_s = \left\{\dot{\mathbf{Z}}_{j,x}^{(0)}\right\}_{j\in[J],x\in[X]}$, where $\dot{\mathbf{Z}}_{j,x}^{(0)}$ are i.i.d. uniform column vectors from $\mathbb{F}_q^K, \forall j \in [J], x \in [X]$. For all $t \in \mathbb{N}, j \in [J], x \in [X]$, let $\dot{\mathbf{Z}}_{j,x}^{(t)}$ be $K \times 1$ column vectors from the finite field $\mathbb{F}_q$. For all $t \in \mathbb{N}, i \in [\max(\mu, K_c)], s \in [T]$, let $\widetilde{\mathbf{Z}}_{i,s}^{(t)}$ be i.i.d. uniform column vectors from $\mathbb{F}_q^K$, and let us set $\mathcal{Z}_U^{(t)} = \left\{\widetilde{\mathbf{Z}}_{i,s}^{(t)}\right\}_{i\in[\max(\mu,K_c)],s\in[T]} \cup \left\{\dddot{Z}_{\ell,i,x}^{(t)}\right\}_{\ell\in[U_w^{(t)}],i\in[K_c],x\in[X_\Delta]}$, where for all $t \in \mathbb{N}, \ell \in [U_w^{(t)}], i \in [K_c], x \in [X_\Delta]$, $\dddot{Z}_{\ell,i,x}^{(t)}$ are i.i.d. uniform scalars from the finite field $\mathbb{F}_q$.

The ACSA-RW scheme (Definition 9) is built upon the elements introduced in Definitions 1–8. The ACSA-RW scheme is given in the form of the construction of the answers returned by the servers and storage update equations, with Definitions 1–8 as building elements. Therefore, it is straightforward to implement the ACSA-RW scheme if the elements as defined in Definitions 1–8 are properly constructed.

[14] The purpose of $\xi$ is primarily to allow the scheme to scale to larger values of $L$, one could assume $\xi = 1$ for simplicity.

$$\mathbf{S}_n^{(0)} + \begin{bmatrix} \frac{1}{\alpha_n - f_1}\Delta_{1,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{1,1,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{2,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{2,1,m} \\ \frac{1}{\alpha_n - f_3}\Delta_{3,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{3,1,m} \\ \frac{1}{\alpha_n - f_1}\Delta_{4,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{4,1,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{5,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{5,1,m} \\ \frac{1}{\alpha_n - f_3}\Delta_{6,1}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{6,1,m} \end{bmatrix} + \begin{bmatrix} \frac{1}{\alpha_n - f_3}\Delta_{1,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{1,2,m} \\ \frac{1}{\alpha_n - f_1}\Delta_{2,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{2,2,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{3,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{3,2,m} \\ \frac{1}{\alpha_n - f_3}\Delta_{4,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{4,2,m} \\ \frac{1}{\alpha_n - f_1}\Delta_{5,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{5,2,m} \\ \frac{1}{\alpha_n - f_2}\Delta_{6,2}^{(1)}\mathbf{e}_K(\theta_1) + \sum_{m\in[4]} \alpha_n^{m-1}\dot{\mathbf{I}}_{6,2,m} \end{bmatrix} \tag{62}$$

$$= \begin{bmatrix} \frac{1}{\alpha_n - f_1}\left(\dot{\mathbf{W}}_{1,1}^{(0)} + \Delta_{1,1}^{(1)}\mathbf{e}_K(\theta_1)\right) + \frac{1}{\alpha_n - f_3}\left(\dot{\mathbf{W}}_{1,2}^{(0)} + \Delta_{1,2}^{(1)}\mathbf{e}_K(\theta_1)\right) + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{1,x}^{(1)} \\ \frac{1}{\alpha_n - f_2}\left(\dot{\mathbf{W}}_{2,1}^{(0)} + \Delta_{2,1}^{(1)}\mathbf{e}_K(\theta_1)\right) + \frac{1}{\alpha_n - f_1}\left(\dot{\mathbf{W}}_{2,2}^{(0)} + \Delta_{2,2}^{(1)}\mathbf{e}_K(\theta_1)\right) + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{2,x}^{(1)} \\ \frac{1}{\alpha_n - f_3}\left(\dot{\mathbf{W}}_{3,1}^{(0)} + \Delta_{3,1}^{(1)}\mathbf{e}_K(\theta_1)\right) + \frac{1}{\alpha_n - f_2}\left(\dot{\mathbf{W}}_{3,2}^{(0)} + \Delta_{3,2}^{(1)}\mathbf{e}_K(\theta_1)\right) + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{3,x}^{(1)} \\ \frac{1}{\alpha_n - f_1}\left(\dot{\mathbf{W}}_{4,1}^{(0)} + \Delta_{4,1}^{(1)}\mathbf{e}_K(\theta_1)\right) + \frac{1}{\alpha_n - f_3}\left(\dot{\mathbf{W}}_{4,2}^{(0)} + \Delta_{4,2}^{(1)}\mathbf{e}_K(\theta_1)\right) + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{4,x}^{(1)} \\ \frac{1}{\alpha_n - f_2}\left(\dot{\mathbf{W}}_{5,1}^{(0)} + \Delta_{5,1}^{(1)}\mathbf{e}_K(\theta_1)\right) + \frac{1}{\alpha_n - f_1}\left(\dot{\mathbf{W}}_{5,2}^{(0)} + \Delta_{5,2}^{(1)}\mathbf{e}_K(\theta_1)\right) + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{5,x}^{(1)} \\ \frac{1}{\alpha_n - f_3}\left(\dot{\mathbf{W}}_{6,1}^{(0)} + \Delta_{6,1}^{(1)}\mathbf{e}_K(\theta_1)\right) + \frac{1}{\alpha_n - f_2}\left(\dot{\mathbf{W}}_{6,2}^{(0)} + \Delta_{6,2}^{(1)}\mathbf{e}_K(\theta_1)\right) + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{6,x}^{(1)} \end{bmatrix} \tag{63}$$

$$= \begin{bmatrix} \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{1,1}^{(1)} + \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{1,2}^{(1)} + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{1,x}^{(1)} \\ \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{2,1}^{(1)} + \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{2,2}^{(1)} + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{2,x}^{(1)} \\ \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{3,1}^{(1)} + \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{3,2}^{(1)} + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{3,x}^{(1)} \\ \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{4,1}^{(1)} + \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{4,2}^{(1)} + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{4,x}^{(1)} \\ \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{5,1}^{(1)} + \frac{1}{\alpha_n - f_1}\dot{\mathbf{W}}_{5,2}^{(1)} + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{5,x}^{(1)} \\ \frac{1}{\alpha_n - f_3}\dot{\mathbf{W}}_{6,1}^{(1)} + \frac{1}{\alpha_n - f_2}\dot{\mathbf{W}}_{6,2}^{(1)} + \sum_{x\in[4]} \alpha_n^{x-1}\dot{\mathbf{Z}}_{6,x}^{(1)} \end{bmatrix} = \mathbf{S}_n^{(1)}, \tag{64}$$

**Definition 1. (Pole Assignment)** *If $\mu \geq K_c$, let us define the $\mu \times K_c$ matrix $\mathbf{F}$ to be the first $K_c$ columns of the following $\mu \times \mu$ matrix.*

$$\begin{bmatrix} \widetilde{f}_1 & \widetilde{f}_\mu & \dots & \widetilde{f}_3 & \widetilde{f}_2 \\ \widetilde{f}_2 & \widetilde{f}_1 & \widetilde{f}_\mu & & \widetilde{f}_3 \\ \vdots & \widetilde{f}_2 & \widetilde{f}_1 & \ddots & \vdots \\ \widetilde{f}_{\mu-1} & & \ddots & \ddots & \widetilde{f}_\mu \\ \widetilde{f}_\mu & \widetilde{f}_{\mu-1} & \dots & \widetilde{f}_2 & \widetilde{f}_1 \end{bmatrix}. \tag{68}$$

*On the other hand, if $\mu < K_c$, let us define the $\mu \times K_c$ matrix $\mathbf{F}$ to be the first $\mu$ rows of the following $K_c \times K_c$ matrix.*

$$\begin{bmatrix} \widetilde{f}_1 & \widetilde{f}_2 & \dots & \widetilde{f}_{K_c-1} & \widetilde{f}_{K_c} \\ \widetilde{f}_{K_c} & \widetilde{f}_1 & \widetilde{f}_2 & & \widetilde{f}_{K_c-1} \\ \vdots & \widetilde{f}_{K_c} & \widetilde{f}_1 & \ddots & \vdots \\ \widetilde{f}_3 & & \ddots & \ddots & \widetilde{f}_2 \\ \widetilde{f}_2 & \widetilde{f}_3 & \dots & \widetilde{f}_{K_c} & \widetilde{f}_1 \end{bmatrix}. \tag{69}$$

*Let $(f_{j,i})_{j\in[J],i\in[K_c]}$ be a total of $JK_c$ elements from the finite field $\mathbb{F}_q$, which are defined as follows.*

$$\begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,K_c} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,K_c} \\ \vdots & \vdots & \vdots & \vdots \\ f_{J,1} & f_{J,2} & \cdots & f_{J,K_c} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{F} \\ \vdots \\ \mathbf{F} \end{bmatrix}. \tag{70}$$

According to the definition, we have the following two propositions immediately.

**Proposition 1.** *For all $i \in [K_c], m, n \in [J]$ such that $m \leq n$, $|[m:n]| \leq \mu$, the constants $(f_{j,i})_{j\in[m:n]}$ are distinct.*

**Proposition 2.** *For all $j \in [J]$, the constants $(f_{j,i})_{i\in[K_c]}$ are distinct.*

**Definition 2. (Noise Assignment)** *If $\mu \geq K_c$, for all $t \in \mathbb{N}, s \in [T]$, let us define the $\mu \times K_c$ matrix $\widetilde{\mathbf{Z}}_s^{(t)}$ to be the first $K_c$ columns of the following $\mu \times \mu$ matrix.*

$$\begin{bmatrix} \widetilde{\mathbf{Z}}_{1,s}^{(t)} & \widetilde{\mathbf{Z}}_{\mu,s}^{(t)} & \dots & \widetilde{\mathbf{Z}}_{3,s}^{(t)} & \widetilde{\mathbf{Z}}_{2,s}^{(t)} \\ \widetilde{\mathbf{Z}}_{2,s}^{(t)} & \widetilde{\mathbf{Z}}_{1,s}^{(t)} & \widetilde{\mathbf{Z}}_{\mu,s}^{(t)} & & \widetilde{\mathbf{Z}}_{3,s}^{(t)} \\ \vdots & \widetilde{\mathbf{Z}}_{2,s}^{(t)} & \widetilde{\mathbf{Z}}_{1,s}^{(t)} & \ddots & \vdots \\ \widetilde{\mathbf{Z}}_{\mu-1,s}^{(t)} & & \ddots & \ddots & \widetilde{\mathbf{Z}}_{\mu,s}^{(t)} \\ \widetilde{\mathbf{Z}}_{\mu,s}^{(t)} & \widetilde{\mathbf{Z}}_{\mu-1,s}^{(t)} & \dots & \widetilde{\mathbf{Z}}_{2,s}^{(t)} & \widetilde{\mathbf{Z}}_{1,s}^{(t)} \end{bmatrix}. \tag{71}$$

*On the other hand, if $\mu < K_c$, for all $t \in \mathbb{N}, s \in [T]$, let us define the $\mu \times K_c$ matrix $\widetilde{\mathbf{Z}}_s^{(t)}$ to be the first $\mu$ rows of the following $K_c \times K_c$ matrix.*

$$\begin{bmatrix} \widetilde{\mathbf{Z}}_{1,s}^{(t)} & \widetilde{\mathbf{Z}}_{2,s}^{(t)} & \dots & \widetilde{\mathbf{Z}}_{K_c-1,s}^{(t)} & \widetilde{\mathbf{Z}}_{K_c,s}^{(t)} \\ \widetilde{\mathbf{Z}}_{K_c,s}^{(t)} & \widetilde{\mathbf{Z}}_{1,s}^{(t)} & \widetilde{\mathbf{Z}}_{2,s}^{(t)} & & \widetilde{\mathbf{Z}}_{K_c-1,s}^{(t)} \\ \vdots & \widetilde{\mathbf{Z}}_{K_c,s}^{(t)} & \widetilde{\mathbf{Z}}_{1,s}^{(t)} & \ddots & \vdots \\ \widetilde{\mathbf{Z}}_{3,s}^{(t)} & & \ddots & \ddots & \widetilde{\mathbf{Z}}_{2,s}^{(t)} \\ \widetilde{\mathbf{Z}}_{2,s}^{(t)} & \widetilde{\mathbf{Z}}_{3,s}^{(t)} & \dots & \widetilde{\mathbf{Z}}_{K_c,s}^{(t)} & \widetilde{\mathbf{Z}}_{1,s}^{(t)} \end{bmatrix}. \tag{72}$$

*For all $t \in \mathbb{N}, s \in [T]$, let $\left(\ddot{\mathbf{Z}}_{u,i,s}^{(t)}\right)_{u\in[J],i\in[K_c]}$ be column vectors from $\mathbb{F}_q^K$, which are defined as follows.*

$$\begin{bmatrix} \ddot{\mathbf{Z}}_{1,1,s}^{(t)} & \ddot{\mathbf{Z}}_{1,2,s}^{(t)} & \cdots & \ddot{\mathbf{Z}}_{1,K_c,s}^{(t)} \\ \ddot{\mathbf{Z}}_{2,1,s}^{(t)} & \ddot{\mathbf{Z}}_{2,2,s}^{(t)} & \cdots & \ddot{\mathbf{Z}}_{2,K_c,s}^{(t)} \\ \vdots & \vdots & \vdots & \vdots \\ \ddot{\mathbf{Z}}_{J,1,s}^{(t)} & \ddot{\mathbf{Z}}_{J,2,s}^{(t)} & \cdots & \ddot{\mathbf{Z}}_{J,K_c,s}^{(t)} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{Z}}_{s}^{(t)} \\ \widetilde{\mathbf{Z}}_{s}^{(t)} \\ \vdots \\ \widetilde{\mathbf{Z}}_{s}^{(t)} \end{bmatrix}. \tag{73}$$

**Definition 3. (ACSA Storage)** *For any $t \in \mathbb{Z}^*$, the storage at the $N$ servers is said to form an ACSA storage if for all $n \in [N]$, $\mathbf{S}_n^{(t)}$ has the following form.*

$$\mathbf{S}_n^{(t)} = \begin{bmatrix} \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{1,i}} \dot{\mathbf{W}}_{1,i}^{(t)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{1,x}^{(t)} \\ \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{2,i}} \dot{\mathbf{W}}_{2,i}^{(t)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{2,x}^{(t)} \\ \vdots \\ \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{J,i}} \dot{\mathbf{W}}_{J,i}^{(t)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{J,x}^{(t)} \end{bmatrix}. \tag{74}$$

*Note that $X$ i.i.d. uniform random noise terms are MDS coded to guarantee the $X$-security.*

**Definition 4. (ACSA Query)** *For any $t \in \mathbb{N}$, the read-queries $\left(Q_n^{(t,\theta_t)}\right)_{n\in[N]}$ by User $t$ for the $N$ servers are said to form an ACSA query if for all $n \in [N]$, we have*

$$Q_n^{(t,\theta_t)} = \left(\mathbf{Q}_{n,1}^{(t,\theta_t)}, \mathbf{Q}_{n,2}^{(t,\theta_t)}, \cdots, \mathbf{Q}_{n,K_c}^{(t,\theta_t)}\right), \tag{75}$$

*where for all $i \in [K_c]$*

$$\mathbf{Q}_{n,i}^{(t,\theta_t)} = \begin{bmatrix} \mathbf{e}_K(\theta_t) + (\alpha_n - f_{1,i}) \sum_{s\in[T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{1,i,s}^{(t)} \\ \mathbf{e}_K(\theta_t) + (\alpha_n - f_{2,i}) \sum_{s\in[T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{2,i,s}^{(t)} \\ \vdots \\ \mathbf{e}_K(\theta_t) + (\alpha_n - f_{J,i}) \sum_{s\in[T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{J,i,s}^{(t)} \end{bmatrix}. \tag{76}$$

*Note that for all $t \in \mathbb{N}, n \in [N]$, $H\left(Q_n^{(t,\theta_t)}\right) = K\max(\mu, K_c)$ in $q$-ary units, because according to Definition 1 and Definition 2, $Q_n^{(t,\theta_t)}$ is uniquely determined by its $K\max(\mu, K_c)$ entries (the rest are replicas). Also note that $T$ i.i.d. uniform random noise terms are MDS coded to guarantee the $T$-privacy.*

**Remark 2.** *Indeed, the fact that $H\left(Q_n^{(t,\theta_t)}\right) = K\max(\mu, K_c)$ in $q$-ary units is jointly guaranteed by the cyclic structures of pole assignment and noise assignment, i.e., Definition 1 and Definition 2. These cyclic structures are important in terms of minimizing the entropy of ACSA queries, so that it does not scale with the number of symbols $L$.*

**Definition 5. (ACSA Increment)** *For any $t \in \mathbb{N}$, the write-queries $\left(P_n^{(t,\theta_t)}\right)_{n\in[N]}$ by User $t$ for the $N$ servers are said to form an ACSA increment if for all $n \in [N]$, we have*

$$P_n^{(t,\theta_t)} = \left(\mathbf{P}_{n,1}^{(t,\theta_t)}, \mathbf{P}_{n,2}^{(t,\theta_t)}, \cdots, \mathbf{P}_{n,K_c}^{(t,\theta_t)}\right), \tag{77}$$

*where for all $i \in [K_c]$,*

$$\mathbf{P}_{n,i}^{(t,\theta_t)} = \mathrm{diag}\left(\widetilde{\Delta}_{n,1,i}^{(t)} \mathbf{I}_{KR_w^{(t)}}, \widetilde{\Delta}_{n,2,i}^{(t)} \mathbf{I}_{KR_w^{(t)}}, \cdots, \widetilde{\Delta}_{n,U_w^{(t)},i}^{(t)} \mathbf{I}_{KR_w^{(t)}}\right), \tag{78}$$

*and for all $\ell \in [U_w^{(t)}]$,*

$$\widetilde{\Delta}_{n,\ell,i}^{(t)} = \sum_{j\in[(\ell-1)R_w^{(t)}+1:\ell R_w^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} \Delta_{j,i}^{(t)} + \sum_{x\in[X_\Delta]} \alpha_n^{x-1} \ddot{Z}_{\ell,i,x}^{(t)}. \tag{79}$$

*Note that for all $t \in \mathbb{N}, n \in [N]$, $H\left(P_n^{(t,\theta_t)}\right) = U_w^{(t)} K_c$ in $q$-ary units. This is because for all $i \in [K_c]$, $\mathbf{P}_{n,i}^{(t,\theta_t)}$ is uniquely determined by $\left(\widetilde{\Delta}_{n,\ell,i}^{(t)}\right)_{\ell\in[U_w^{(t)}]}$. Also note that $X_\Delta$ MDS coded i.i.d. uniform noise terms are used to guarantee the $X_\Delta$-security.*

**Definition 6. (ACSA Packer)** *For all $t \in \mathbb{N}, n \in [N]$, the ACSA Packer is defined as follows.*

$$\Xi_n^{(t)} = \left(\mathbf{\Xi}_{n,\ell,i}^{(t)}\right)_{\ell\in[U_r^{(t)}], i\in[K_c]}, \tag{80}$$

*where for all $\ell \in [U_r^{(t)}]$, $i \in [K_c]$,*

$$\mathbf{\Xi}_{n,\ell,i}^{(t)} = \mathrm{diag}\left(\frac{\prod_{i'\in[K_c]\setminus\{i\}}(\alpha_n - f_{1,i'})}{\prod_{i'\in[K_c]\setminus\{i\}}(f_{1,i} - f_{1,i'})}\mathbf{I}_K, \cdots, \frac{\prod_{i'\in[K_c]\setminus\{i\}}(\alpha_n - f_{J,i'})}{\prod_{i'\in[K_c]\setminus\{i\}}(f_{J,i} - f_{J,i'})}\mathbf{I}_K\right) \times \mathrm{diag}(\underbrace{0\mathbf{I}_K, \cdots, 0\mathbf{I}_K}_{R_r^{(t)}(\ell-1)\ 0\mathbf{I}_K\text{'s}}, \underbrace{1\mathbf{I}_K, \cdots, 1\mathbf{I}_K}_{R_r^{(t)}\ 1\mathbf{I}_K\text{'s}}, \underbrace{0\mathbf{I}_K, \cdots, 0\mathbf{I}_K}_{(J-\ell R_r^{(t)})\ 0\mathbf{I}_K\text{'s}}). \tag{81}$$

*Note that the ACSA packer is a* **constant** *since $|\mathcal{S}_r^{(t)}|$ is globally known.*

**Definition 7. (ACSA Unpacker)** *For all $t \in \mathbb{N}, n \in [N]$, the ACSA Unpacker is defined as follows.*

$$\Upsilon_n^{(t)} = \left(\mathbf{\Upsilon}_{n,1}^{(t)}, \mathbf{\Upsilon}_{n,2}^{(t)}, \cdots, \mathbf{\Upsilon}_{n,K_c}^{(t)}\right), \tag{82}$$

*where for all $i \in [K_c]$,*

$$\mathbf{\Upsilon}_{n,i}^{(t)} = \mathrm{diag}\left(\left(\frac{\prod_{j\in\mathcal{F}_1^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_1^{(t)}}(f_{1,i} - f_{j,i})}\right)\mathbf{I}_K, \cdots, \left(\frac{\prod_{j\in\mathcal{F}_J^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_J^{(t)}}(f_{J,i} - f_{j,i})}\right)\mathbf{I}_K\right), \tag{83}$$

*where for all $j \in [J]$, we define $\mathcal{F}_j = \left[\left(\left\lceil j/R_w^{(t)}\right\rceil - 1\right)R_w^{(t)} + 1 : \left\lceil j/R_w^{(t)}\right\rceil R_w^{(t)}\right] \setminus \{j\}$. Note that the ACSA unpacker is a* **constant** *since $|\mathcal{S}_w^{(t)}|$ is globally known.*

**Remark 3.** *As noted at the beginning of this section, as well as in the motivating example, private read and write operations consists of $K_c U_r^{(t)}$ and $K_c U_w^{(t)}$ sub-operations, and for each sub-operation, $R_r^{(t)}$ and $R_w^{(t)}$ symbols of the desired submodel are retrieved and updated respectively. This is respectively made possible by the constructions of ACSA Packer and Unpacker. For private read, by ACSA Packer,*

*in each sub-operation $R_r^{(t)}$ symbols of the desired submodel are "packed" into a Cauchy-Vandermonde structured answer string for best download efficiency, where Cauchy terms carry desired symbols and interference symbols are aligned along Vandermonde terms. Recoverability of the desired symbols is guaranteed by the invertibility of Cauchy-Vandermonde matrices (see, e.g., [63]). On the other hand, ACSA Increment (Definition 5) can be viewed as a bunch of Cauchy-Vandermonde structured codewords, and in each codeword, a total of $R_w^{(t)}$ symbols are "packed" together for best upload efficiency. To correctly perform private write, each of the codewords is "unpacked" by the ACSA Unpacker to produce $R_w^{(t)}$ codewords that individually carry different increment symbols to preserve the ACSA Storage structure (Definition 3).*

**Definition 8. (ACSA Null-shaper)** *For all $t \in \mathbb{N}, n \in [N]$, the ACSA null-shaper is defined as follows.*

$$\Omega_n^{(t)} = \left(\mathbf{\Omega}_{n,1}^{(t)}, \mathbf{\Omega}_{n,2}^{(t)}, \cdots, \mathbf{\Omega}_{n,K_c}^{(t)}\right), \tag{84}$$

*where for all $i \in [K_c]$,*

$$\mathbf{\Omega}_{n,i}^{(t)} = \mathrm{diag}\left(\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{1,i}-\alpha_m)}\right)\mathbf{I}_K, \cdots, \left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{J,i}-\alpha_m)}\right)\mathbf{I}_K\right). \tag{85}$$

*Note that for all $n \in \mathcal{S}_w^{(t)}$, we have $\mathbf{\Omega}_n = \mathbf{0}$. Besides, the ACSA null-shaper is a* **constant** *since $\mathcal{S}_w^{(t)}$ is globally known.*

**Definition 9. (ACSA-RW Scheme)** *The initial storage at the $N$ servers is the ACSA storage at time 0, i.e., $\left(\mathbf{S}_n^{(0)}\right)_{n\in[N]}$. At time $t, t \in \mathbb{N}$, in the read phase, the user uploads the ACSA query for the $N$ servers and retrieves the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ from the answers returned by the servers $n, n \in \overline{\mathcal{S}}_r^{(t)}$, which are constructed as follows.*

$$A_n^{(t,\theta_t)} = \left(\left(\mathbf{S}_n^{(t-1)}\right)^{\mathsf{T}} \mathbf{\Xi}_{n,\ell,i}^{(t)} \mathbf{Q}_{n,i}^{(t,\theta_t)}\right)_{\ell\in[U_r^{(t)}], i\in[K_c]}, n \in \overline{\mathcal{S}}_r^{(t)}. \tag{86}$$

*In the update phase, the user uploads the ACSA increment for the servers $n, n \in \overline{\mathcal{S}}_w^{(t)}$, and each of the servers updates its storage according to the following equation.*

$$\mathbf{S}_n^{(t)} = \mathbf{S}_n^{(t-1)} + \sum_{i\in[K_c]} \mathbf{\Omega}_{n,i}^{(t)} \mathbf{\Upsilon}_{n,i}^{(t)} \mathbf{P}_{n,i}^{(t,\theta_t)} \mathbf{Q}_{n,i}^{(t,\theta_t)}, n \in \overline{\mathcal{S}}_w^{(t)}. \tag{87}$$

Now let us prove the correctness, privacy and security of the ACSA-RW scheme. To proceed, we need the following lemmas.

**Lemma 1.** *At any time $t, t \in \mathbb{N}$, User $t$ retrieves the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ from the answers returned by the servers $n, n \in \overline{\mathcal{S}}_r^{(t)}$ according to the ACSA-RW scheme while guaranteeing $T$-privacy.*

*Proof.* Let us consider the answers returned by the servers $n, n \in \overline{\mathcal{S}}_r^{(t)}$ in (86). Note that for all $\ell \in [U_r^{(t)}]$, $i \in [K_c]$, we have (88)–(92), shown at the top of the next page. For all $j \in [J], m \in [X+T+K_c-1]$, $\dot{I}_{j,i,m}^{(t)}$ are various linear combinations of inner products of $\left(\dot{\mathbf{W}}_{j,i}^{(t-1)}\right)_{i\in[K_c]}, \mathbf{e}_K(\theta_t), \left(\dot{\mathbf{Z}}_{j,x}^{(t-1)}\right)_{x\in[X]}$ and $\left(\ddot{\mathbf{Z}}_{j,i,s}^{(t)}\right)_{i\in[K_c],s\in[T]}$, whose exact forms are irrelevant. Besides, $\ddot{I}_{\ell,i,m}^{(t)} = \sum_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \dot{I}_{j,i,m}^{(t)}, \forall \ell \in [U_r^{(t)}], i \in [K_c]$. Note that for all $j \in [J], i \in [K_c]$, $\prod_{i'\in[K_c]\setminus\{i\}}(f_{j,i}-f_{j,i'})$ is the remainder of the polynomial division (with respect to $\alpha_n$) $\left(\prod_{i'\in[K_c]\setminus\{i\}}(\alpha_n-f_{j,i'})\right)/(\alpha_n-f_{j,i})$, which is for normalization. The existence of multiplicative inverse of $\prod_{i'\in[K_c]\setminus\{i\}}(f_{j,i}-f_{j,i'})$ is guaranteed by Proposition 2, i.e., $\prod_{i'\in[K_c]\setminus\{i\}}(f_{j,i}-f_{j,i'}) \neq 0$. Therefore, due to the fact that the desired symbols are carried by the Cauchy terms (i.e., the first term in (92)) and the interference symbols (i.e., the undesired symbols, second term in (92)) are aligned along the Vandermonde terms, the desired symbols $\left(W_{\theta_t}^{(t-1)}(j,i)\right)_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]}$ of the submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ are resolvable by inverting the Cauchy-Vandermonde matrix in (93), shown at the top of the next page. It is remarkable that the non-singularity of the matrix $\mathbf{C}$ follows from the determinant of Cauchy-Vandermonde matrix (see, e.g., [63]) and the fact that according to Proposition 1, the constants $\left((f_{j,i})_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]}, (\alpha_n)_{n\in\overline{\mathcal{S}}_r^{(t)}}\right)$ are distinct for all $\ell \in [U_r^{(t)}]$, regardless of the realizations of $\overline{\mathcal{S}}_r^{(t)}$ and $\overline{\mathcal{S}}_w^{(t)}$. Recall that $\overline{\mathcal{S}}_r^{(t)}(1)$, $\overline{\mathcal{S}}_r^{(t)}(2)$, etc., refer to distinct elements of $\overline{\mathcal{S}}_r^{(t)}$ (arranged in ascending order). Therefore, the user is able to reconstruct the desired submodel due to the fact that $\mathbf{W}_{\theta_t}^{(t-1)} = \left(\left(W_{\theta_t}^{(t-1)}(j,i)\right)_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]}\right)_{\ell\in[U_r^{(t)}],i\in[K_c]}$. To see why the $T$-privacy holds, we note that by the construction of the query $\left(Q_n^{(t,\theta_t)}\right)_{n\in[N]}$, the vector $\mathbf{e}_K(\theta_t)$, which carries the information of desired index $\theta_t$, is protected by the MDS$(N,T)$ coded uniform i.i.d. random noise vectors. Thus the queries for the $N$ servers form a secret sharing of threshold $T$, and are independent of the queries and the increments of all prior users $\tau, \tau \in [t-1]$. Thus $T$-privacy is guaranteed. To calculate the download cost, we note that a total of $L = JK_c$ symbols of the desired submodel are retrieved out of the $K_c U_r^{(t)} |\overline{\mathcal{S}}_r^{(t)}|$ downloaded symbols. Therefore we have $D_t = \left(K_c U_r^{(t)} |\overline{\mathcal{S}}_r^{(t)}|\right)/(JK_c) = \left(U_r^{(t)}\left(N - |\mathcal{S}_r^{(t)}|\right)\right)/J = \left(N - |\mathcal{S}_r^{(t)}|\right)/R_r^{(t)} = \left(N - |\mathcal{S}_r^{(t)}|\right)/\left(S_r^{\text{thresh}} - |\mathcal{S}_r^{(t)}|\right)$. This completes the proof of Lemma 1. □

**Lemma 2.** *At any time $t, t \in \mathbb{N}$, User $t$ correctly updates the desired submodel $\mathbf{W}_{\theta_t}^{(t-1)}$ and achieves ACSA storage by uploading the ACSA increments to the servers $n, n \in \overline{\mathcal{S}}_w^{(t)}$ and exploiting the update equation (87) according to the ACSA-RW scheme while guaranteeing $T$-privacy and $X_\Delta$-security.*

*Proof.* Let us first inspect the second term on the RHS of (87). Note that for any $t \in \mathbb{N}$, for all $n \in [N], i \in [K_c]$, we can write

$$\mathbf{\Omega}_{n,i}^{(t)} \mathbf{\Upsilon}_{n,i}^{(t)} \mathbf{P}_{n,i}^{(t,\theta_t)} \mathbf{Q}_{n,i}^{(t,\theta_t)}$$

$$
\begin{aligned}
&\left(\mathbf{S}_n^{(t-1)}\right)^{\mathsf{T}} \boldsymbol{\Xi}_{n,\ell,i}^{(t)} \mathbf{Q}_{n,i}^{(t,\theta_t)} \\
&= \sum_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \left( \sum_{i'\in[K_c]} \frac{1}{\alpha_n - f_{j,i'}} \dot{\mathbf{W}}_{j,i'}^{(t-1)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{j,x}^{(t-1)} \right)^{\mathsf{T}} \\
&\qquad \left( \frac{\prod_{i'\in[K_c]\setminus\{i\}} (\alpha_n - f_{j,i'})}{\prod_{i'\in[K_c]\setminus\{i\}} (f_{j,i} - f_{j,i'})} \right) \left( \mathbf{e}_K(\theta_t) + (\alpha_n - f_{j,i}) \sum_{s\in[T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{j,i,s}^{(t)} \right) \qquad (88) \\
&= \sum_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \left( \sum_{i'\in[K_c]} \frac{1}{\alpha_n - f_{j,i'}} \dot{\mathbf{W}}_{j,i'}^{(t-1)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{j,x}^{(t-1)} \right)^{\mathsf{T}} \qquad (89) \\
&\qquad \left( \left( \frac{\prod_{i'\in[K_c]\setminus\{i\}} (\alpha_n - f_{j,i'})}{\prod_{i'\in[K_c]\setminus\{i\}} (f_{j,i} - f_{j,i'})} \right) \mathbf{e}_K(\theta_t) + \frac{\prod_{i'\in[K_c]} (\alpha_n - f_{j,i'})}{\prod_{i'\in[K_c]\setminus\{i\}} (f_{j,i} - f_{j,i'})} \sum_{s\in[T]} \alpha_n^{s-1} \ddot{\mathbf{Z}}_{j,i,s}^{(t)} \right) \qquad (90) \\
&= \sum_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \left( \frac{1}{\alpha_n - f_{j,i}} \dot{\mathbf{W}}_{j,i}^{(t-1)} \mathbf{e}_K(\theta_t) + \sum_{m\in[X+T+K_c-1]} \alpha_n^{m-1} \dot{I}_{j,i,m}^{(t)} \right) \qquad (91) \\
&= \sum_{j\in[(\ell-1)R_r^{(t)}+1:\ell R_r^{(t)}]} \frac{1}{\alpha_n - f_{j,i}} W_{\theta_t}^{(t-1)}(j,i) + \sum_{m\in[X+T+K_c-1]} \alpha_n^{m-1} \ddot{I}_{\ell,i,m}^{(t)}. \qquad (92)
\end{aligned}
$$

$$
\mathbf{C} = \begin{bmatrix}
\frac{1}{f_{(\ell-1)R_r^{(t)}+1,i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(1)}} & \cdots & \frac{1}{f_{\ell R_r^{(t)},i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(1)}} & 1 & \cdots & \alpha_{\overline{\mathcal{S}}_r^{(t)}(1)}^{X+T+K_c-1} \\
\frac{1}{f_{(\ell-1)R_r^{(t)}+1,i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(2)}} & \cdots & \frac{1}{f_{\ell R_r^{(t)},i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(2)}} & 1 & \cdots & \alpha_{\overline{\mathcal{S}}_r^{(t)}(2)}^{X+T+K_c-1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{1}{f_{(\ell-1)R_r^{(t)}+1,i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(|\overline{\mathcal{S}}_r^{(t)}|)}} & \cdots & \frac{1}{f_{\ell R_r^{(t)},i} - \alpha_{\overline{\mathcal{S}}_r^{(t)}(|\overline{\mathcal{S}}_r^{(t)}|)}} & 1 & \cdots & \alpha_{\overline{\mathcal{S}}_r^{(t)}(|\overline{\mathcal{S}}_r^{(t)}|)}^{X+T+K_c-1}
\end{bmatrix}. \qquad (93)
$$

$$
= \left[ \left(\boldsymbol{\Gamma}_{n,1,i}^{(t)}\right)^{\mathsf{T}}, \left(\boldsymbol{\Gamma}_{n,2,i}^{(t)}\right)^{\mathsf{T}}, \cdots, \left(\boldsymbol{\Gamma}_{n,U_w^{(t)},i}^{(t)}\right)^{\mathsf{T}} \right]^{\mathsf{T}}, \qquad (94)
$$

where for all $\ell \in [U_w^{(t)}]$, we define $\phi_\ell = (\ell-1)R_w^{(t)}$, and $\boldsymbol{\Gamma}_{n,\ell,i}^{(t)}$ is written as (95)–(98) (shown in the next page), where for all $v \in [R_w^{(t)}]$, $\left(\ddot{I}_{\phi_\ell+v,m}\right)_{m\in[X_\Delta+R_w^{(t)}-1]}$, $\left(\dddot{I}_{\phi_\ell+v,m}\right)_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}$ and $\left(\dot{\mathbf{I}}_{\phi_\ell+v,m}\right)_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}$ are various interference symbols, whose exact forms are not important. Note that in (96), we multiply the first two terms in each row of (95). It can be justified from the fact that the constants $(f_{j,i})_{j\in[\phi_\ell+1:\phi_\ell+R_w^{(t)}]}$ are distinct according to Proposition 1. Besides, for all $v \in [R_w^{(t)}]$, according to the definition of $\mathcal{F}_{\phi_\ell+v}^{(t)}$, we can equivalently write $\mathcal{F}_{\phi_\ell+v}^{(t)} = [\phi_\ell+1 : \phi_\ell+R_w^{(t)}] \setminus \{\phi_\ell+v\}$, thus $\left|\mathcal{F}_{\phi_\ell+v}^{(t)}\right| = R_w^{(t)} - 1$ and $(\phi_\ell+v) \notin \mathcal{F}_{\phi_\ell+v}$. Note that the denominator in the first term in each row of (95) is for normalization, which is the remainder of the polynomial division (with respect to $\alpha_n$) $\left(\prod_{j\in\mathcal{F}_{\phi_\ell+v}^{(t)}} (\alpha_n - f_{j,i})\right) / (\alpha_n - f_{\phi_\ell+v,i})$. In (97), we multiply the first two terms in each row of (96), and it can be justified by noting that the denominator in the first term in each row of (96) is the remainder of the polynomial division (with respect to $\alpha_n$) $\left(\prod_{m\in\mathcal{S}_w^{(t)}} (\alpha_n - \alpha_m)\right) / (\alpha_n - f_{\phi_\ell+v,i})$. And finally in (98), we multiply the two terms in each row of (97). Therefore, for all $n \in [N], i \in [K_c]$, the term $\boldsymbol{\Omega}_{n,i}^{(t)} \boldsymbol{\Upsilon}_{n,i}^{(t)} \mathbf{P}_{n,i}^{(t,\theta_t)} \mathbf{Q}_{n,i}^{(t,\theta_t)}$ can be written as follows.

$$
\begin{aligned}
&\boldsymbol{\Omega}_{n,i}^{(t)} \boldsymbol{\Upsilon}_{n,i}^{(t)} \mathbf{P}_{n,i}^{(t,\theta_t)} \mathbf{Q}_{n,i}^{(t,\theta_t)} \\
&= \begin{bmatrix}
\frac{1}{\alpha_n - f_{1,i}} \Delta_{1,i}^{(t)} \mathbf{e}_K(\theta_t) \\
\qquad + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]} \alpha_n^{m-1} \dot{\mathbf{I}}_{1,i,m}^{(t)} \\
\hdashline
\frac{1}{\alpha_n - f_{2,i}} \Delta_{2,i}^{(t)} \mathbf{e}_K(\theta_t) \\
\qquad + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]} \alpha_n^{m-1} \dot{\mathbf{I}}_{2,i,m}^{(t)} \\
\hdashline
\vdots \\
\hdashline
\frac{1}{\alpha_n - f_{J,i}} \Delta_{J,i}^{(t)} \mathbf{e}_K(\theta_t) \\
\qquad + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]} \alpha_n^{m-1} \dot{\mathbf{I}}_{J,i,m}^{(t)}
\end{bmatrix}. \qquad (99)
\end{aligned}
$$

Note that $X = X_\Delta + R_w^{(t)} + |\mathcal{S}_w^{(t)}| + T - 1$, the update equation (87) is thus correct because

$$
\mathbf{S}_n^{(t-1)} + \sum_{i\in[K_c]} \boldsymbol{\Omega}_{n,i}^{(t)} \boldsymbol{\Upsilon}_{n,i}^{(t)} \mathbf{P}_{n,i}^{(t,\theta_t)} \mathbf{Q}_{n,i}^{(t,\theta_t)}
$$

$$
\boldsymbol{\Gamma}_{n,\ell,i}^{(t)} = \begin{bmatrix}
\left(\frac{\prod_{j\in\mathcal{F}_{\phi_\ell+1}^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_{\phi_\ell+1}^{(t)}}(f_{\phi_\ell+1,i}-f_{j,i})}\right)\left(\sum_{j\in[\phi_\ell+1:\phi_\ell+R_w^{(t)}]}\frac{1}{\alpha_n - f_{j,i}}\Delta_{j,i}^{(t)} + \sum_{x\in[X_\Delta]}\alpha_n^{x-1}\dddot{Z}_{\ell,i,x}^{(t)}\right) \\
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+1,i}-\alpha_m)}\right)\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+1,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+1,i,s}^{(t)}\right) \\
\hdashline
\left(\frac{\prod_{j\in\mathcal{F}_{\phi_\ell+2}^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_{\phi_\ell+2}^{(t)}}(f_{\phi_\ell+2,i}-f_{j,i})}\right)\left(\sum_{j\in[\phi_\ell+1:\phi_\ell+R_w^{(t)}]}\frac{1}{\alpha_n - f_{j,i}}\Delta_{j,i}^{(t)} + \sum_{x\in[X_\Delta]}\alpha_n^{x-1}\dddot{Z}_{\ell,i,x}^{(t)}\right) \\
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+2,i}-\alpha_m)}\right)\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+2,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+2,i,s}^{(t)}\right) \\
\hdashline
\vdots \\
\hdashline
\left(\frac{\prod_{j\in\mathcal{F}_{\phi_\ell+R_w^{(t)}}^{(t)}}(\alpha_n - f_{j,i})}{\prod_{j\in\mathcal{F}_{\phi_\ell+R_w^{(t)}}^{(t)}}(f_{\phi_\ell+R_w^{(t)},i}-f_{j,i})}\right)\left(\sum_{j\in[\phi_\ell+1:\phi_\ell+R_w^{(t)}]}\frac{1}{\alpha_n - f_{j,i}}\Delta_{j,i}^{(t)} + \sum_{x\in[X_\Delta]}\alpha_n^{x-1}\dddot{Z}_{\ell,i,x}^{(t)}\right) \\
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+R_w^{(t)},i}-\alpha_m)}\right)\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+R_w^{(t)},i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+R_w^{(t)},i,s}^{(t)}\right)
\end{bmatrix} \tag{95}
$$

$$
= \begin{bmatrix}
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+1,i}-\alpha_m)}\right)\left(\frac{1}{\alpha_n - f_{\phi_\ell+1,i}}\Delta_{\phi_\ell+1,i}^{(t)} + \sum_{m\in[X_\Delta+R_w^{(t)}-1]}\alpha_n^{m-1}\dddot{I}_{\phi_\ell+1,i,m}^{(t)}\right) \\
\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+1,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+1,i,s}^{(t)}\right) \\
\hdashline
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+2,i}-\alpha_m)}\right)\left(\frac{1}{\alpha_n - f_{\phi_\ell+2,i}}\Delta_{\phi_\ell+2,i}^{(t)} + \sum_{m\in[X_\Delta+R_w^{(t)}-1]}\alpha_n^{m-1}\dddot{I}_{\phi_\ell+2,i,m}^{(t)}\right) \\
\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+2,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+2,i,s}^{(t)}\right) \\
\hdashline
\vdots \\
\hdashline
\left(\frac{\prod_{m\in\mathcal{S}_w^{(t)}}(\alpha_n-\alpha_m)}{\prod_{m\in\mathcal{S}_w^{(t)}}(f_{\phi_\ell+R_w^{(t)},i}-\alpha_m)}\right)\left(\frac{1}{\alpha_n - f_{\phi_\ell+R_w^{(t)},i}}\Delta_{\phi_\ell+R_w^{(t)},i}^{(t)} + \sum_{m\in[X_\Delta+R_w^{(t)}-1]}\alpha_n^{m-1}\dddot{I}_{\phi_\ell+R_w^{(t)},i,m}^{(t)}\right) \\
\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+R_w^{(t)},m})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+R_w^{(t)},i,s}^{(t)}\right)
\end{bmatrix} \tag{96}
$$

$$
= \begin{bmatrix}
\left(\frac{1}{\alpha_n - f_{\phi_\ell+1,i}}\Delta_{\phi_\ell+1,i}^{(t)} + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}\alpha_n^{m-1}\dddot{I}_{\phi_\ell+1,i,m}^{(t)}\right) \\
\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+1,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+1,i,s}^{(t)}\right) \\
\hdashline
\left(\frac{1}{\alpha_n - f_{\phi_\ell+2,i}}\Delta_{\phi_\ell+2,i}^{(t)} + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}\alpha_n^{m-1}\dddot{I}_{\phi_\ell+2,i,m}^{(t)}\right) \\
\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+2,i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+2,i,s}^{(t)}\right) \\
\hdashline
\vdots \\
\hdashline
\left(\frac{1}{\alpha_n - f_{\phi_\ell+R_w^{(t)},i}}\Delta_{\phi_\ell+R_w^{(t)},i}^{(t)} + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|-1]}\alpha_n^{m-1}\dddot{I}_{\phi_\ell+R_w^{(t)},i,m}^{(t)}\right) \\
\left(\mathbf{e}_K(\theta_t) + (\alpha_n - f_{\phi_\ell+R_w^{(t)},i})\sum_{s\in[T]}\alpha_n^{s-1}\ddot{\mathbf{Z}}_{\phi_\ell+R_w^{(t)},i,s}^{(t)}\right)
\end{bmatrix} \tag{97}
$$

$$
= \begin{bmatrix}
\left(\frac{1}{\alpha_n - f_{\phi_\ell+1,i}}\Delta_{\phi_\ell+1,i}^{(t)}\mathbf{e}_K(\theta_t) + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}_{\phi_\ell+1,i,m}^{(t)}\right) \\
\hdashline
\left(\frac{1}{\alpha_n - f_{\phi_\ell+2,i}}\Delta_{\phi_\ell+2,i}^{(t)}\mathbf{e}_K(\theta_t) + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}_{\phi_\ell+2,i,m}^{(t)}\right) \\
\hdashline
\vdots \\
\hdashline
\left(\frac{1}{\alpha_n - f_{\phi_\ell+R_w^{(t)},i}}\Delta_{\phi_\ell+R_w^{(t)},i}^{(t)}\mathbf{e}_K(\theta_t) + \sum_{m\in[X_\Delta+R_w^{(t)}+|\mathcal{S}_w^{(t)}|+T-1]}\alpha_n^{m-1}\dot{\mathbf{I}}_{\phi_\ell+R_w^{(t)},i,m}^{(t)}\right)
\end{bmatrix}, \tag{98}
$$

$$= \begin{bmatrix} \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{1,i}} \dot{\mathbf{W}}_{1,i}^{(t-1)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{1,x}^{(t-1)} \\ \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{2,i}} \dot{\mathbf{W}}_{2,i}^{(t-1)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{2,x}^{(t-1)} \\ \vdots \\ \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{J,i}} \dot{\mathbf{W}}_{J,i}^{(t-1)} + \sum_{x\in[X]} \alpha_n^{x-1} \dot{\mathbf{Z}}_{J,x}^{(t-1)} \end{bmatrix}$$
$$+ \begin{bmatrix} \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{1,i}} \Delta_{1,i}^{(t)} \mathbf{e}_K(\theta_t) + \sum_{x\in[X]} \alpha_n^{x-1} \ddot{\mathbf{I}}_{1,x}^{(t)} \\ \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{2,i}} \Delta_{2,i}^{(t)} \mathbf{e}_K(\theta_t) + \sum_{x\in[X]} \alpha_n^{x-1} \ddot{\mathbf{I}}_{2,x}^{(t)} \\ \vdots \\ \sum_{i\in[K_c]} \frac{1}{\alpha_n - f_{J,i}} \Delta_{J,i}^{(t)} \mathbf{e}_K(\theta_t) + \sum_{x\in[X]} \alpha_n^{x-1} \ddot{\mathbf{I}}_{J,x}^{(t)} \end{bmatrix} \tag{100}$$

$$= \mathbf{S}_n^{(t)}, \tag{101}$$

where for all $t \in \mathbb{N}, j \in [J], x \in [X]$, $\dot{\mathbf{Z}}_{j,x}^{(t)} = \dot{\mathbf{Z}}_{j,x}^{(t-1)} + \ddot{\mathbf{I}}_{j,x}^{(t)}$, and $\ddot{\mathbf{I}}_{j,x}^{(t)} = \sum_{i\in[K_c]} \dot{\mathbf{I}}_{j,i,x}^{(t)}$. Note that for all $n \in \mathcal{S}_w^{(t)}$, $i \in [K_c]$, we have $\mathbf{\Omega}_{n,i}^{(t)} = \mathbf{0}$. Therefore, for all $n \in \mathcal{S}_w^{(t)}$, it holds that $\mathbf{S}_n^{(t)} = \mathbf{S}_n^{(t-1)}$. In other words, the update equation (87) correctly updates the desired submodel and achieves the ACSA storage at time $t+1$ by updating the storage of server $n, n \in \overline{\mathcal{S}}_w^{(t)}$. The proof of $T$-privacy follows from that in Lemma 1, so we do not repeat it here. The proof of $X_\Delta$-security follows from the fact that by the definition of ACSA increment $\left(P_n^{(t,\theta_t)}\right)_{n\in[N]}$, the symbols of the increment $\mathbf{\Delta}_t$ are protected by the MDS$(N, X_\Delta)$ coded uniform i.i.d. random noise symbols. Thus the ACSA increment for the $N$ servers form a secret sharing of threshold $X_\Delta$, and it is independent of the write-queries by the users $\tau, \tau \in [t-1]$ and the read-queries by the users $\tau, \tau \in [t]$. Finally let us calculate the upload cost. The upload cost consists of two parts, i.e., the upload cost of the ACSA query and the upload cost of the ACSA increment. Note that to upload the ACSA query, a total of $\left|[N] \setminus \left(\mathcal{S}_r^{(t)} \cap \mathcal{S}_w^{(t)}\right)\right| K \max(\mu, K_c)$ $q$-ary symbols must be uploaded. On the other hand, to upload the ACSA increment, we need to upload a total of $\left(N - |\mathcal{S}_w^{(t)}|\right) U_w^{(t)} K_c$ $q$-ary symbols. Therefore, in the limit as $L/K \to \infty$, the normalized upload cost is

$$U_t = \frac{K_c\left(N - |\mathcal{S}_w^{(t)}|\right) U_w^{(t)} + \left|[N] \setminus \left(\mathcal{S}_r^{(t)} \cap \mathcal{S}_w^{(t)}\right)\right| K \max(\mu, K_c)}{L} \tag{102}$$

$$\overset{L/K\to\infty}{=} \frac{N - |\mathcal{S}_w^{(t)}|}{R_w^{(t)}} \tag{103}$$

$$= \frac{N - |\mathcal{S}_w^{(t)}|}{S_w^{\text{thresh}} - |\mathcal{S}_w^{(t)}|}. \tag{104}$$

This completes the proof of Lemma 2. □

The ACSA-RW scheme satisfies the correctness, $T$-privacy, and $X_\Delta$-security constraints for each update $t, t \in \mathbb{N}$ because of Lemma 1 and Lemma 2, which hold for all $t, t \in \mathbb{N}$. Now let us see why the $X$-security constraint is satisfied. By the definition of the ACSA storage (i.e., Definition 3) at any time $t, t \in \mathbb{N}$, the symbols of the $K$ submodels are protected by the MDS$(N, X)$ coded i.i.d. uniform random noise symbols. In other words, it forms a secret sharing of threshold $X$, thus $X$-security is guaranteed.

**Remark 4. (Byzantine Tolerance)** *It is remarkable that the answers returned by the servers in the private read phase can be viewed as codewords of an MDS code (generated by the Cauchy-Vandermonde matrix, see, e.g., [27]). Therefore, with additional $2B$ redundant answers from the servers, we can correct up to $B$ erroneous answers.*

**Remark 5. (Symmetric Security)** *If common randomness is allowed among the servers, so-called* **symmetric security** *can be achieved [64], i.e., the user will learn nothing about the global model beyond the desired submodel. Note that this does not affect the communication cost of the ACSA-RW scheme.*

**Remark 6. (External Adversaries versus Internal Adversaries)** *Recall that the $X$-security constraint only requires protection against an external adversary who can access the current storage but not the past history at any $X$-servers. However, a closer look at the ACSA-RW scheme reveals that if the number of compromised servers is no more than $\min(X_\Delta, T)$, then even an* internal *adversary, i.e., an adversary who has access to the entire history of all previous stored values and queries seen by the compromised servers, can still learn nothing about the stored submodels.*

**Remark 7. (Access Complexity)** *We define the access complexity as the number of elements over the finite field $\mathbb{F}_q$ that must be accessed/updated during the private read and private write phases. We note that at any time $t, t \in \mathbb{N}$, the access complexity of each of the responsive servers in the private read and write phases is at most $KL/K_c$. Hence with greater $K_c$, it is possible to reduce the access complexity.*

**Remark 8. (Encoding and Decoding Complexity)** *Let us consider the complexity of the encoding and decoding algorithms of our construction. It is worth noting that the computations for producing the ACSA storage, ACSA query and ACSA increment can be regarded as multiplications of (scaled) Cauchy-Vandermonde matrices with various vectors. The computation for recovering the desired submodel by the user from the answers of the $N$ servers can be viewed as solving linear systems defined by Cauchy-Vandermonde matrices. Cauchy-Vandermonde matrices are an important class of structured matrices, for which "*superfast*" algorithms have been studied extensively [65], [66]. Therefore, by these superfast algorithms, the complexity of producing the ACSA storage, ACSA query and ACSA increment is at most $\widetilde{\mathcal{O}}((LKN\log^2 N)/K_c), \widetilde{\mathcal{O}}(\mu KNK_c \log^2 N)$ and $\widetilde{\mathcal{O}}(U_t L \log^2 N)$, respectively. On the other hand, the complexity of decoding the desired submodel from the answers of the servers is at most $\widetilde{\mathcal{O}}(D_t L \log^2 N)$. It is obvious that the encoding/decoding algorithms have a complexity that is almost linear in their output/input sizes.*

## V. Conclusion

Inspired by the recent interest in $X$-secure $T$-private federated submodel learning, we explored the fundamental problem of privately reading from and writing to a distributed and secure database. By interpreting the private read and write operations as secure matrix multiplications (between query

vectors and stored data), and recognizing that CSA codes are natural solutions to such problems, we constructed a novel Adaptive CSA-RW scheme. ACSA-RW achieves synergistic gains from the joint design of private read and write operations because the same one hot vector representation of the desired message index needs to be secret shared for both the private read and write operations. In addition to allowing private read and write, ACSA-RW also provides elastic resilience against server dropouts, up to thresholds that are determined by the number of redundant storage dimensions. Surprisingly, ACSA-RW is able to fully update the distributed database even though the database is only partially accessible due to write-dropout servers. This is accomplished by exploiting the redundancy that is already required for secure storage. The scheme allows a memoryless operation of the database in the sense that the storage structure is preserved and users may remain oblivious of the prior history of server dropouts. A promising direction for future work is to explore applications of this idea to multi-version coding [67].

## REFERENCES


[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. [Online]. Available: http://dx.doi.org/10.1561/2200000083

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, 2019.

[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." New York, NY, USA: Association for Computing Machinery, 2020.

[6] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, p. 333, 2011.

[7] O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde, "Binary relevance efficacy for multilabel classification," *Progress in Artificial Intelligence*, vol. 1, no. 4, pp. 303–313, 2012.

[8] H. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *Proceedings of ICLR*, 2018.

[9] M. Abadi, A. Chu, I. Goodfellow, H. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *Proceedings of CCS*, 2016.

[10] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta, "Privacy amplification by iteration," *Proceedings of FOCS*, 2018.

[11] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948.

[12] M. Kim and J. Lee, "Information-theoretic privacy in federated submodel learning," *ICT Express*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959522000297

[13] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, 1995, pp. 41–50.

[14] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private Information Retrieval," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 965–981, 1998.

[15] H. Sun and S. A. Jafar, "The Capacity of Private Information Retrieval," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4075–4088, July 2017.

[16] ——, "The Capacity of Robust Private Information Retrieval with Colluding Databases," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2361–2370, April 2018.

[17] ——, "The capacity of symmetric private information retrieval," *IEEE Transactions on Information Theory*, 2018.

[18] Q. Wang, H. Sun, and M. Skoglund, "The capacity of private information retrieval with eavesdroppers," *IEEE Transactions on Information Theory*, vol. 65, no. 5, pp. 3198–3214, 2019.

[19] Q. Wang and M. Skoglund, "Secure private information retrieval from colluding databases with eavesdroppers," *arXiv preprint arXiv:1710.01190*, 2017.

[20] K. Banawan and S. Ulukus, "The capacity of private information retrieval from byzantine and colluding databases," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 1206–1219, Feb 2019.

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

[22] Q. Wang, H. Sun, and M. Skoglund, "The $\epsilon$-error capacity of symmetric pir with byzantine adversaries," *arXiv preprint arXiv:1809.03988*, 2018.

[23] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, C. Hollanti, and S. E. Rouayheb, "Private Information Retrieval Schemes for Coded Data with Arbitrary Collusion Patterns," *arXiv preprint arXiv:1701.07636*, 2017.

[24] Q. Wang and M. Skoglund, "Linear symmetric private information retrieval for MDS coded distributed storage with colluding servers," *arXiv preprint arXiv:1708.05673*, 2017.

[25] R. Freij-Hollanti, O. Gnilke, C. Hollanti, and D. Karpuk, "Private Information Retrieval from Coded Databases with Colluding Servers," *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 647–664, 2017.

[26] H. Sun and S. A. Jafar, "Private Information Retrieval from MDS Coded Data with Colluding Servers: Settling a Conjecture by Freij-Hollanti et al." *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1000–1022, February 2018.

[27] Z. Jia and S. A. Jafar, "$X$-secure $T$-private Information Retrieval from MDS Coded Storage with Byzantine and Unresponsive Servers," *IEEE Transactions on Information Theory*, 2020, DOI: 10.1109/TIT.2020.3013152.

[28] R. Zhou, C. Tian, H. Sun, and T. Liu, "Capacity-achieving private information retrieval codes from mds-coded databases with minimum message size," *IEEE Transactions on Information Theory*, 2020.

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

[30] Z. Jia, H. Sun, and S. A. Jafar, "Cross Subspace Alignment and the Asymptotic Capacity of $X$ -Secure $T$ -Private Information Retrieval," *IEEE Trans. on Info. Theory*, vol. 65, no. 9, pp. 5783–5798, Sep. 2019.

[31] K. Banawan, B. Arasli, Y.-P. Wei, and S. Ulukus, "The capacity of private information retrieval from heterogeneous uncoded caching databases," *IEEE Transactions on Information Theory*, vol. 66, no. 6, pp. 3407–3416, 2020.

[32] Y.-P. Wei, B. Arasli, K. Banawan, and S. Ulukus, "The capacity of private information retrieval from decentralized uncoded caching databases," *Information*, vol. 10, no. 12, p. 372, 2019.

[33] M. A. Attia, D. Kumar, and R. Tandon, "The capacity of private information retrieval from uncoded storage constrained databases," *IEEE Transactions on Information Theory*, vol. 66, no. 11, pp. 6617–6634, 2020.

[34] Y. Wei, K. Banawan, and S. Ulukus, "Fundamental limits of cache-aided private information retrieval with unknown and uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 65, no. 5, pp. 3215–3232, May 2019.

[35] R. Tandon, "The capacity of cache aided private information retrieval," *arXiv preprint arXiv:1706.07035*, 2017.

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

[37] Y. Wei, K. Banawan, and S. Ulukus, "The capacity of private information retrieval with partially known private side information," *IEEE Transactions on Information Theory*, vol. 65, no. 12, pp. 8222–8231, 2019.

[38] Z. Chen, Z. Wang, and S. A. Jafar, "The capacity of t-private information retrieval with private side information," *IEEE Transactions on Information Theory*, vol. 66, no. 8, pp. 4761–4773, 2020.

[39] H. Sun and S. A. Jafar, "Multiround Private Information Retrieval: Capacity and Storage Overhead," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5743–5754, August 2018.

[40] X. Yao, N. Liu, and W. Kang, "The capacity of multi-round private information retrieval from byzantine databases," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 2124–2128.

[41] K. Banawan and S. Ulukus, "Multi-message private information retrieval: Capacity results and near-optimal schemes," *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6842–6862, 2018.

[42] M. J. Siavoshani, S. P. Shariatpanahi, and M. A. Maddah-Ali, "Private information retrieval for a multi-message scenario with private side information," *IEEE Transactions on Communications*, pp. 1–1, 2021.

[43] Z. Wang, K. Banawan, and S. Ulukus, "Private set intersection: A multi-message symmetric private information retrieval perspective," *IEEE Transactions on Information Theory*, pp. 1–1, 2021.

[44] 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*, vol. 65, no. 11, pp. 7613–7627, 2019.

[45] H. Sun and S. A. Jafar, "The capacity of private computation," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3880–3897, June 2019.

[46] M. Mirmohseni and M. A. Maddah-Ali, "Private function retrieval," *arXiv preprint arXiv:1711.04677*, 2017.

[47] Z. Chen, Z. Wang, and S. A. Jafar, "The asymptotic capacity of private search," *IEEE Transactions on Information Theory*, vol. 66, no. 8, pp. 4709–4721, 2020.

[48] Z. Jia and S. A. Jafar, "On the asymptotic capacity of x-secure t-private information retrieval with graph-based replicated storage," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6280–6296, 2020.

[49] Y. Lu, Z. Jia, and S. A. Jafar, "Double blind t-private information retrieval," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 428–440, 2021.

[50] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7420–7437, 2021.

[51] S. Lu and R. Ostrovsky, "Distributed oblivious ram for secure two-party computation," in *Theory of Cryptography Conference*. Springer, 2013, pp. 377–396.

[52] E. Kushilevitz and T. Mour, "Sub-logarithmic distributed oblivious ram with small block size," *arXiv preprint arXiv:1802.05145*, 2018.

[53] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint arXiv:2009.11248*, 2020.

[54] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[55] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning*, 2017, pp. 3368–3376.

[56] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 903–912.

[57] J.-y. Sohn, D.-J. Han, B. Choi, and J. Moon, "Election coding for distributed learning: Protecting signsgd against byzantine attacks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 14 615–14 625.

[58] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.

[59] A. Schönhage and V. Strassen, "Schnelle multiplikation grosser zahlen," *Computing*, vol. 7, no. 3-4, pp. 281–292, 1971.

[60] K. Banawan and S. Ulukus, "The Capacity of Private Information Retrieval from Coded Databases," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1945–1956, 2018.

[61] R. Bitar and S. E. Rouayheb, "Staircase-pir: Universally robust private information retrieval," *arXiv preprint arXiv:1806.08825*, 2018.

[62] D. H. Duong, P. K. Mishra, and M. Yasuda, "Efficient secure matrix multiplication over lwe-based homomorphic encryption," *Tatra Mountains Mathematical Publications*, vol. 67, no. 1, pp. 69–83, 2016.

[63] M. Gasca, J. Martinez, and G. Mühlbach, "Computation of Rational Interpolants with Prescribed Poles," *Journal of Computational and Applied Mathematics*, vol. 26, no. 3, pp. 297–309, 1989.

[64] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "Gcsa codes with noise alignment for secure coded multi-party batch matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 306–316, 2021.

[65] V. Y. Pan, *Structured matrices and polynomials: unified superfast algorithms*. Springer Science & Business Media, 2012.

[66] T. Finck, G. Heinig, and K. Rost, "An inversion formula and fast algorithms for cauchy-vandermonde matrices," *Linear algebra and its applications*, vol. 183, pp. 179–191, 1993.

[67] Z. Wang and V. R. Cadambe, "Multi-version coding—an information-theoretic perspective of consistent distributed storage," *IEEE Transactions on Information Theory*, vol. 64, no. 6, pp. 4540–4561, 2017.


**Zhuqing Jia** (Member, IEEE) received the B.E. degree in electronic information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015, the M.S. degree in electrical and computer engineering and the Ph.D. degree in electrical and computer engineering from the University of California Irvine, USA, in 2019 and 2021, respectively.

He is currently an Assistant Professor with the School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include information theory and its applications to security, privacy, computation, and storage. Zhuqing Jia was a recipient of a UC Irvine EECS department fellowship for his PhD studies.

**Syed Ali Jafar** (Fellow, IEEE) received his B. Tech. from IIT Delhi, India, in 1997, M.S. from Caltech, USA, in 1999, and Ph.D. from Stanford, USA, in 2003, all in Electrical Engineering. His industry experience includes positions at Lucent Bell Labs and Qualcomm. He is a Chancellor's Professor of Electrical Engineering and Computer Science at the University of California Irvine, Irvine, CA USA. His research interests include multiuser information theory, wireless communications and network coding.

Dr. Jafar is a recipient of the New York Academy of Sciences Blavatnik National Laureate in Physical Sciences and Engineering, the NSF CAREER Award, the ONR Young Investigator Award, the UCI Academic Senate Distinguished Mid-Career Faculty Award for Research, the School of Engineering Mid-Career Excellence in Research Award and the School of Engineering Maseeh Outstanding Research Award. His co-authored papers have received the IEEE Information Theory Society Paper Award, IEEE Communication Society and Information Theory Society Joint Paper Award, IEEE Communications Society Best Tutorial Paper Award, IEEE Communications Society Heinrich Hertz Award, IEEE Signal Processing Society Young Author Best Paper Award, IEEE Information Theory Society Jack Wolf ISIT Best Student Paper Award, and three IEEE GLOBECOM Best Paper Awards. Dr. Jafar received the UC Irvine EECS Professor of the Year award six times from the Engineering Students Council, a School of Engineering Teaching Excellence Award in 2012, and a Senior Career Innovation in Teaching Award in 2018. He was a University of Canterbury Erskine Fellow, an IEEE Communications Society Distinguished Lecturer, an IEEE Information Theory Society Distinguished Lecturer and a Thomson Reuters/Clarivate Analytics Highly Cited Researcher. He served as Associate Editor for IEEE Transactions on Communications 2004-2009, for IEEE Communications Letters 2008-2009 and for IEEE Transactions on Information Theory 2009-2012.