SwiftAgg+: Achieving Asymptotically Optimal Communication Loads in Secure Aggregation for Federated Learning

Tayyebeh Jahani-Nezhad, Member, IEEE, Mohammad Ali Maddah-Ali, Fellow, IEEE, Songze Li, Member, IEEE, and Giuseppe Caire, Fellow, IEEE

Abstract—We propose SwiftAgg+, a novel secure aggregation protocol for federated learning systems, where a central server aggregates local models of \( N \in \mathbb{N} \) distributed users, each of size \( L \in \mathbb{N} \), trained on their local data, in a privacy-preserving manner. SwiftAgg+ can significantly reduce the communication overheads without any compromise on security, and achieve optimal communication loads within diminishing gaps. Specifically, in presence of at most \( D = o(N) \) dropout users, SwiftAgg+ achieves a per-user communication load of \( (1 + \mathcal{O}(\frac{1}{N}))L \) symbols and a server communication load of \( (1 + \mathcal{O}(\frac{1}{N}))L \) symbols, with a worst-case information-theoretic security guarantee, against any subset of up to \( T = o(N) \) semi-honest users who may also collude with the curious server. Moreover, the proposed SwiftAgg+ allows for a flexible trade-off between communication loads and the number of active communication links. In particular, for \( T < N - D \) and for any \( K \in \mathbb{N} \), SwiftAgg+ can achieve the server communication load of \( (1 + \frac{2}{K})L \) symbols, and per-user communication load of up to \( (1 + \frac{2}{K+2})L \) symbols, where the number of pair-wise active connections in the network is \( \frac{1}{2}(K + T + D + 1) \).

Index Terms—Federated learning, secure aggregation, secret sharing, dropout resiliency, optimal communication load.

I. INTRODUCTION

FEDERATED learning (FL) is an emerging distributed learning framework that allows a group of distributed users (e.g., mobile devices) to collaboratively train a global model of size \( L \) with their local private data, without sharing the data [1], [2], [3]. Specifically, in a FL system with a central server and \( N \) users, during each training iteration, the server sends the current state of the global model to the users. Receiving the global model, each user then trains a local model with its local data, and sends the local model to the server. By aggregating the local models, the server can update the global model for the next iteration. While the local datasets are not directly shared with the server, several studies have shown that a curious server can launch model inversion attacks to reveal information about the training data of individual users from their local models (see, e.g., [4], [5]). Therefore, the key challenge to protect users’ data privacy is to design secure aggregation protocols, which allow the aggregation of local models to be computed without revealing each individual model even if the server colludes with a limited number of users, called semi-honest users, in the system. Moreover, as some users may randomly drop out of the aggregation process (due to low batteries or unstable connections), the server should be able to robustly recover the aggregated local models of the surviving users in a privacy-preserving manner.

As such motivated, a secure aggregation protocol SecAgg is proposed in [6]. In SecAgg, each pair of users agree on a shared random vector. In addition, each user generates a private random vector and secret shares it with other users. Before sending its local model to the server, each user masks the local model in a particular way. The mask is created based on the private and shared random vectors such that the shared part can be canceled out when aggregated at the server. In order to recover the result and cancel the remaining masks, the private random vector of each surviving user and the shared random vectors of each dropped user must be reconstructed using the collected shares from surviving users in the second round of communication.

Note that to have an information-theoretic security, the shared and the private random vectors must be chosen uniformly at random from vectors of size \( L \). However, to make it practical in SecAgg, it is assumed that the users agree on common seeds for a pseudo-random generator rather than on the entire shared random vectors which is not secure in the context of information theory. In this paper, we focus on information-theoretic security guarantees.
One of the major challenges for SecAgg to scale up is the communication load. The secret sharing among users requires all-to-all communication, which incurs quadratic cost in the number of users $N$. There has been a series of works that aim to improve the communication efficiency of SecAgg (see, e.g., [7], [8], [9]). In [7], TurboAgg is proposed to perform secure aggregation following a circular topology, achieving a communication load of $O(LN \log N)$ at the server and $O(L \log N)$ at each user. The idea of TurboAgg is partitioning the users into several groups each of size $O(\log N)$ to accelerate model aggregation phase. To protect individual models, each user masks its local model in a particular way. Furthermore, the masked local models are encoded using Lagrange coded computing [10] such that the data of dropped users can be recovered at each stage of the protocol. Each user in a group computes the aggregation of the masked models and the aggregation of coded models of all users in the previous groups, and sends them to all users in the next group, as well as the masked and coded version of its model. The missing terms of aggregated masked models, caused by the dropped users, can be recovered in the next group if at least half of the users in a group do not drop out. TurboAgg guarantees the privacy of individual local models with high probability as long as the number of semi-honest users is less than $N/2$.

In [8], SecAgg+ is proposed which considers a $k$-regular communication graph among users instead of the complete graph used in SecAgg to reduce the communication and computation loads, where $k = O(\log N)$. It means that each user secret shares with a subset of other users instead of sharing with all of them. It is shown that SecAgg+ requires a communication of $O(LN + N \log N)$ at the server and $O(L + \log N)$ at each user. In [9], another similar idea, CCESA, is proposed in which a sparse random graph is used as communication network instead of the complete graph. Thus, secret sharing is used for a subset of users instead of all-to-all communication. While SecAgg+, TurboAgg and CCESA improve the communication efficiency of SecAgg, they only provide probabilistic privacy guarantees as opposed to the worst-case guarantee of SecAgg.

Some other secure aggregation protocols focus on reducing the computation complexity of SecAgg, rather than the communication load. FastSecAgg is proposed in [11] which presents a multi-secret sharing scheme based on a finite field version of the Fast Fourier Transform to achieve a trade-off between the number of secrets, the dropout tolerance and the privacy threshold. FastSecAgg reduces the computation load of the server as well as per-user to $O((L \log N)$, while achieves the same order of communication as SecAgg with lower privacy threshold and dropout tolerance.

In [12], LightSecAgg is proposed to reduce the computation complexity bottleneck in secure aggregation. In LightSecAgg, each user independently samples a random mask, encodes it using $T$-private MDS (maximum distance separable) coding, generating secret shares of the mask to send to other users. Then, each user adds mask to its local model and sends the masked model to the server. The server can reconstruct and cancel out the aggregated mask of surviving users via a one-shot decoding, using the aggregated coded masks received in a second phase of communication with surviving users. Similar idea of one-shot reconstruction of aggregated mask was also utilized in [13], which nevertheless requires a trusted third party, and much more randomness needs to be generated, as well as more storage at each user.

As an earlier and conference version of this manuscript, in [14], we propose SwiftAgg which reduces the server communication load via partitioning users into several groups and using Shamir’s secret sharing [15] to keep the local models private. SwiftAgg can able to achieve correct aggregation in presence of up to $D$ dropout users, with formally-proven the worst-case security guarantee against up to $T$ semi-honest users. Independently, similar ideas have been used in CodedSecAgg [16], but dropout users in groups have not been addressed and there is no formal definition and proof of privacy. The communication loads and the computation loads of the existing approaches are compared in Table I.

In this paper, we propose SwiftAgg+ as a new scheme for secure aggregation in federated learning, which is robust against up to $D$ user dropouts and achieves worst-case security guarantees against up to $T < N - D$ semi-honest users which may collude with each other as well as with the server to gain some information. In SwiftAgg+, in order to compute the aggregation of local models in a privacy-preserving manner, the number of symbols received by the server is $(1 + O(LN^{1/2}))L$ as long as $T = o(N)$ and $N - D = O(N)$. In addition, the number of symbols sent by each user is $(1 + O(LN^{1/2}))L$, for $T = o(N)$ and $D = o(N)$.

Moreover, since in federated learning many of the users may not be accessible from each other, the number of active communication links, among users or from users to the server, is also an important factor when designing secure aggregation protocols. SwiftAgg+ allows for a flexible trade-off between the communication loads and the network connections. To achieve this trade-off, we first partition the users into groups of size $T + D + K$ for some $K \in \mathbb{N}$ (see Fig. 4), and the groups are arranged on an arbitrary hierarchical tree with the server as the root. Then in the first phase, each user partitions its local model into $K$ parts. In the second phase, users within each group secret share the partitions of their local models using ramp secret sharing [17], and aggregate the shares locally. In the third phase, each user aggregates in-group shares and the messages received from the corresponding users in its children groups and sends the result to the corresponding user in its parent group, which is finally sent to the server. In the third phase, if one user in a group drops out, the rest of the users in the sequence that includes the dropped user remain silent. This aggregation tree provides flexibility to adjust the delay of each iteration of training process based on the network connections. Additionally, parameter $K$ controls the trade-off between communication loads and network connectivity which can be chosen based on the constraints of the network.
SwiftAgg+ simultaneously achieves the following advantages compared to the existing works:

1) For $T = o(N)$ and $N - D = O(N)$, the server communication load in SwiftAgg+ is within factor 1 of the cut-set lower bound.

2) SwiftAgg+ has the flexibility to reduce the number of active connections between the users at the cost of increasing the communication load away from the optimum.

3) It is resilient to up to $D$ user dropouts. As opposed to the existing schemes, in SwiftAgg+, due to the structure of the scheme, the effects of user dropouts are already handled in one round of communication with the server, and there is no need to take place another round of communication.

4) It achieves worst-case information-theoretic security against a curious server and any subset of up to $T < N - D$ semi-honest users.

5) Based on the possible connections among users, SwiftAgg+ have a flexibility to control the delay of each training iteration.

Table I compares different schemes for secure aggregation problem in terms of the communication and computation loads. For a fair comparison, we consider two metrics: the server communication load and the per-user communication load. Server communication load indicates the aggregated size of all messages which are sent or received by the server, and per-user communication load denotes the aggregated size of all messages which are sent by each user. Furthermore, server and per-user computations indicate the computation loads of the server and each user respectively. In this table, the computational complexity of the schemes in [7], [12], and [14], and SwiftAgg+, is calculated based on Reed-Solomon decoding complexity [18]. Any improvements in this complexity can be applied to those schemes.

Compared with existing schemes, as shown in Table I, SwiftAgg+ significantly reduces the server communication load and per-user communication load, and for $T = o(N)$ and $N - D = O(N)$, it requires $L(1 + O(\frac{1}{N}))$ symbols for the server communication load and for $T = o(N)$ and $D = o(N)$, it requires $L(1 + O(\frac{1}{N}))$ symbols for the per-user communication load.

The proposed SwiftAgg+ also enjoys small server computation complexity with worst-cast security guarantees.

Notation: For $n \in \mathbb{N}$ the notation $[n]$ represents set $\{1, \ldots, n \}$. The notation $a|b$ means that $a$ divides $b$. Furthermore, the cardinality of set $S$ is denoted by $|S|$. In addition, we denote the difference of two sets $A \setminus B$, that means the set of elements which belong to $A$ but not $B$. $H(X)$ denotes the entropy of random variable $X$ and $I(X; Y)$ is the mutual information of two random variables $X$ and $Y$.

II. PROBLEM FORMULATION

We consider the secure aggregation problem, for a federated learning system, consisting of a server and $N$ users $U_1, \ldots, U_N$. For each $n \in [N]$, user $n$ has a private local model of size $L$, denoted by $W_n \in F^L$, for some finite field $F$. Consider that the local models are from some joint distribution $W_1, W_2, \ldots, W_N \sim P_{W_1, W_2, \ldots, W_N}(W_1, W_2, \ldots, W_N)$ which is often unknown. Each entry of the local models, after a universal linear (affine) mapping, is represented with a non-negative integer number, less than $\ell$ for some $\ell \in \mathbb{N}$. Each user $n$ also has a collection of random variables $Z_n$, whose elements are selected uniformly at random from $\mathbb{F}_L$, and independently from each other and from the local models. Users can send messages to each other and also to the server, using error-free private communication links. $M^{L}_{n \rightarrow n'} \in \mathbb{F}^L \cup \{\perp\}$ denotes the message that user $n$ sends to user $n'$. In addition, $X^{(L)}_n \in \mathbb{F}^L \cup \{\perp\}$ denotes the message sent by node $n$ to the server. The null symbol $\perp$ represents the case no message is sent.

The message $M^{L}_{n \rightarrow n'}$ is a function of $W_n$, $Z_n$, and the messages that node $n$ has received from other nodes so far. We denote the corresponding encoding function by $f^{L}_{n \rightarrow n'}$. Similarly, $X^{(L)}_n$ is a function of $W_n$, $Z_n$, and the messages that node $n$ has received from other nodes so far. We denote the corresponding encoding function by $g^{L}_n$. For a subset $S \subseteq [N]$, we let $X^{(L)}_S = \{X^{(L)}_n\}_{n \in S}$ represent the set of messages the server receives from users in $S$.

We assume that a subset $D \subseteq [N]$ of users drop out, i.e., stay silent (or send $\perp$ to other nodes and the server) during the protocol execution. We denote the number of dropped out users as $D = |D|$. We also assume that a subset $T \subseteq [N]$ of the users, whose identities are not known, are
semi-honest. It means that users in $T$ follow the protocol faithfully; however, they are curious and may collude with each other or with the server to gain information about the local models of the honest users. We assume $|T| \leq T$, for some known security parameter $T < N - D$.

A secure aggregation scheme consists of the encoding functions $f^{(L)}_{n \rightarrow n}$ and $g^{(L)}_{0, n, n' \in [N]}$, such that the following conditions are satisfied:

1. Correctness: The server is able to recover $W = \sum_{n \in [N] \setminus T} W_n$, using $X_{[N] \setminus T} = (X^{(L)}_n)_{n \in [N] \setminus T}$. More precisely,

$$H \left( \sum_{n \in [N] \setminus T} W_n | {X_{[N] \setminus T}} \right) = 0. \quad (1)$$

2. Privacy Constraint: For any joint distribution $P_{W_1, W_2, \ldots, W_N}(W_1, W_2, \ldots, W_N)$, after receiving $X_{[N] \setminus T}$ and colluding with semi-honest users in $T$, the server should not gain any information about local models of the honest users, beyond the aggregation of them, and beyond what it infers from their correlation with the local models of the semi-honest users.\(^2\) Formally,

$$I \left( W_n, n \in [N] \setminus T ; X_{[N] \setminus T} ; \bigcup_{k \in T} \left( M^{(L)}_{k \rightarrow k', k' \in [N]} \right) \right) \leq 0. \quad (2)$$

For a secure aggregation scheme satisfying the above two conditions, we define per-user communication load and server communication load as follows:

**Definition 1** (Normalized Average Per-User Communication Load): denoted by $R^{(L)}_{\text{user}}$, is defined as the aggregated size of all messages sent by users, normalized by $NL$, i.e.,

$$R^{(L)}_{\text{user}} = \frac{1}{NL} \sum_{n \in [N], n' \in [N] \setminus n} \left( H(M^{(L)}_{n \rightarrow n'}) + H(X^{(L)}_n) \right).$$

**Definition 2** (Normalized Server Communication Load): denoted by $R^{(L)}_{\text{server}}$, is defined as the aggregated size of all messages received by the server, normalized by $L$, i.e.,

$$R^{(L)}_{\text{server}} = \frac{1}{L} \sum_{n \in [N]} H(X^{(L)}_n).$$

We say that the pair of $(R_{\text{server}}, R_{\text{user}})$ is achievable, if there exist a sequence of secure aggregation schemes with rate tuples $(R^{(L)}_{\text{server}}, R^{(L)}_{\text{user}})$, $L = 1, 2, \ldots$, such that

$$R_{\text{server}} = \limsup_{L \rightarrow \infty} R^{(L)}_{\text{server}},$$

$$R_{\text{user}} = \limsup_{L \rightarrow \infty} R^{(L)}_{\text{user}}.$$ 

**Definition 3** (Communication Graph for an Achievable Scheme): For an achievable scheme $\mathcal{A}(N, T, D)$ for secure aggregation problem, consisting of $N$ users and a server, we define the undirected communication graph $G_{\mathcal{A}}(V, E_{\mathcal{A}})$. In this graph, $V = \{U_1, \ldots, U_N, \text{server}\}$ is the set of vertices representing users and the server, and $E_{\mathcal{A}}$ is the set of the edges corresponding to active communication links among these vertices. In this graph, edge $e \in E_{\mathcal{A}}$ between two vertices $v_1, v_2 \in V$ exists if the corresponding users communicate directly with each other (they do not send null message $\bot$).

## III. MAIN RESULTS

In this section, we present our main results on per-user and server communication loads, achieved by the proposed SwiftAgg+ scheme. Note that the operations are done in a finite field, which is large enough to avoid hitting the boundary in the process of aggregation. We choose a finite field $GF(p)$ denoted by $\mathbb{F}_p$, for some prime number $p$, where $N < p < 2N$. Moreover, $N(\ell - 1) < p < 2N(\ell - 1)$.

**Theorem 1:** Consider a secure aggregation problem, with $N$ users and one server, where up to $T$ users are semi-honest and up to $D$ users may drop out. There is an achievable scheme which need

$$R_{\text{server}} = \left( 1 + \frac{T}{N - T - D} \right),$$

$$R_{\text{user}} \leq \left( 1 + \frac{T + D}{N - T - D} \right), \quad (3)$$

symbols from $\mathbb{F}_p$.

**Proof:** The proof can be found in Section VI-A.

To achieve the communication loads of (3) in Theorem 1, we propose SwiftAgg+, a novel secure aggregation scheme in federated learning. In SwiftAgg+, under the condition of $T = o(N)$ and $D = o(N)$, the number of symbols sent by each user is $(1 + O(\frac{1}{N}))L$ and the number of symbols received by the server is $(1 + O(\frac{1}{N}))L$ in order to compute the aggregation of local models in a privacy-preserving manner. In other words, for large $N$, the communication load per user tends to $L$ symbols from $\mathbb{F}_p$, and the total communication load of the server also tends to $L$ symbols from $\mathbb{F}_p$.

**Theorem 2:** For $T = o(N)$ and $N - D = O(N)$, in terms of the number of required bits, the server communication load in SwiftAgg+ is within factor 1 of the cut-set lower bound.

**Proof:** The proof can be found in Section VI-C.

**Theorem 3:** For local models with uniform distributions, in terms of the number of required bits, the per-user communication load in SwiftAgg+ is within factor $\log_2 L$ of the cut-set lower bound, for $T = o(N)$ and $D = o(N)$.

**Proof:** The proof can be found in Section VI-D.

**Remark 1:** Note that the case $T = o(N)$ is well justified. Restricted with a limited budget, it often becomes prohibitively difficult for the adversary to hack a constant fraction of the users as the number of users grows. For example, it is seems easier to hack one of only two users rather than 50 out of 100 ones.

As the number of active communication links is important in federated learning, SwiftAgg+ also allows us to reduce the number of active network connections in the secure aggregation problem at the cost of increasing the communication
load away from the optimum. The result is presented in the following theorem.

**Theorem 4:** Consider a secure aggregation problem, with \( N \) users and one server, where up to \( T \) users are semi-honest and up to \( D \) users may drop out. For any \( K \in \mathbb{N}, \) where \( (T + D + K)|N, \) there exist an achievable scheme \( \mathcal{A}_K(N,T,D) \) with

\[
R^s_{\text{server}} = 1 + \frac{T}{K},
\]

\[
R^s_{\text{user}} \leq \left( 1 + \frac{T + D}{K} \right),
\]

symbols from \( \mathbb{F}_p. \) The communication graph of this achievable scheme \( \mathcal{G}_{\mathcal{A}_K}(V, E_{\mathcal{A}_K}) \) consists of \( |E_{\mathcal{A}_K}| = \frac{N}{T}(K + T + D + 1) \) edges.

**Proof:** The proof can be found in Section VI-A. \( \square \)

**Remark 2:** In (4) by increasing parameter \( K, \) both per-user communication load and server communication load are reduced, while the number of active connections in the achievable scheme \( |E_{\mathcal{A}_K}| \) is increased and vice versa. This allows us to have a flexible trade-off between communication loads and the complexity of the communication pattern (in terms of number of active links).

To achieve the trade-off between communication loads and network connections \( |E_{\mathcal{A}_K}|, \) SwiftAgg+ partitions the users into some disjoint groups, each of size \( K + D + T \) users labeled as user 1 to user \( K + D + T \) of that group. The \( N \) groups are arranged in an arbitrary tree with the server as the root. The model aggregation starts from the leaves of the tree, and proceeds in three main phases: Phase 1: Each user partitions the vector of its local model of length \( L \) into \( K \) sub-vectors of length \( \frac{L}{K}, \) Phase 2: Each user uses a ramp sharing \([17]\) to share the sub-vectors of its model to other users within its group. Phase 3: Each user aggregates the shares that it receives in Phase 2 and also adds it to the messages received from the corresponding users in its children groups and sends the result to the corresponding user in its parent group. These messages are finally reach to the server and are used to recover the aggregated local models.

**Remark 3:** The minimum communication loads in Theorem 1 are achieved by setting \( K, \) to its maximum value of \( N - D - T. \) The minimum number of communication connections is achieved by setting \( K = 1, \) which reduces to the SwiftAgg scheme in [14].

**Remark 4:** While in problem formulation we let \( L \to \infty, \) the proposed scheme works for finite values of \( L, \) where \( K \mid L. \) If \( K \nmid L, \) we can zero-pad the local models.

**IV. ILLUSTRATIVE EXAMPLES**

In this section, we present two illustrative examples of the proposed SwiftAgg+ scheme to solve a secure aggregation problem, under different choices of the design parameter \( K. \) This also demonstrates the trade-off between communication loads and the number of communication links achieved by varying \( K \) in SwiftAgg+.

We consider a secure aggregation problem over a federated learning system of one server and \( N = 12 \) users, \( U_1, U_2, \ldots, U_{12}. \) At most \( D = 1 \) user may dropout during the aggregation process, and up to \( T = 2 \) semi-honest users may collude with each other to gain some information about the local models of other users. Each user \( n \) has a local model \( W_n \) which is a vector of size \( L. \)

### A. Minimum Communication Loads With \( K = 9 \)

Each user \( n \) samples uniformly at random two vectors \( Z_n = \{Z_{n,1}, Z_{n,2}\} \) from \( \mathbb{F}_p^d, \) and then takes the following steps:

1. **Partitioning the Local Models:** User \( n \) partitions its local model into \( K = 9 \) parts, i.e.,

\[
W_n = [W_{n,1}, W_{n,2}, \ldots, W_{n,9}]^T,
\]

where each part \( W_{n,k}, k \in [9] \) is a vector of size \( \frac{L}{9}. \)

2. **Secret Sharing and Aggregation:** User \( n \) forms the following polynomial.

\[
F_n(x) = W_{n,1} + W_{n,2}x + \cdots + W_{n,9}x^8 + Z_{n,1}x^9 + Z_{n,2}x^{10},
\]

where the coefficients of the first \( K = 9 \) terms are the partitions of the local model of user \( n, n \in [12]. \)

Let \( \alpha_t \in \mathbb{F}, t \in [12], \) be distinct non-zero constants. We assign \( \alpha_t \) to user \( t. \) In this step, each user \( n \) sends the evaluation of its polynomial function at \( \alpha_t, i.e., F_n(\alpha_t), \) to user \( t, \) for \( t \in [12]. \) Fig. 1 represents the connection links among the users and server. As shown in Fig. 1, \( \mathcal{G}(V, E) \) for this example is a complete graph which means all users communicate with each other and to the server. If a user \( m \) drops out and stays silent, \( F_m(\alpha_t) \) is just presumed to be zero.

Each user \( n \) calculates \( S_n = \sum_{i=1}^{12} F_{n,i}(\alpha_n). \) In this example, assume that \( U_3 \) drops out and does not send its share to other users. Other users treat its share as zero. In this phase, at most 132 communications take place, each of size \( \frac{L}{9}. \)

3. **Communication with the Server:** User \( n \) sends \( S_n \) to the server, for \( n \in [12]. \) Clearly in this example, user 3 remains silent and sends nothing (or null message \( \bot \)) to the server.

4. **Recovering the Result:** Let us define

\[
F(x) \triangleq \sum_{n=1}^{12} F_n(x) = \sum_{k=1}^{9} x^{k-1} \sum_{n \neq 3}^{12} W_{n,k} + x^9 \sum_{n \neq 1}^{12} Z_{n,1} + x^{10} \sum_{n \neq 3}^{12} Z_{n,2}.
\]

One can verify that \( S_n, \) for \( n \in [12]\setminus\{3\} \) that are received by the server are indeed equal to \( F(\alpha_1), F(\alpha_2), F(\alpha_4), \ldots, F(\alpha_{12}). \)

Since \( F(x) \) is a polynomial function of degree 10, based on Lagrange interpolation rule the server can recover all the coefficients of this polynomial using the outcomes of users which are not dropped. In particular, the server can recover \( \sum_{n=1}^{12} W_n \) using the coefficients of \( x^k, k \in [0:8] \) in the recovered polynomial function. Thus, the server
is able to recover the aggregation of local models of surviving users and the correctness constraint is satisfied.

In this example, the per-user communication load is \( \frac{4}{3} L \) and the server communication load is \( \frac{11}{3} L \). In addition, the communication graph is a complete graph with 78 edges and 13 vertices, where because of user dropout no communication occurs on 12 edges.

B. Trading Off Communication Loads for Less Connections

Now we consider a scenario in which not all user can communicate with each other. SwiftAgg+ allows for sacrificing the communication load to reduce the number of required communication links. In this case, we take \( K = 3 \), and each user locally samples two vectors \( \mathbb{Z}_n = \{ \mathbb{Z}_{n1}, \mathbb{Z}_{n2} \} \) at random from \( \mathbb{F}_q^2 \). To reduce the number of connections, each user takes the following steps:

1) Partitioning the Local Models: User \( n \) partitions its local model into \( K = 3 \) parts, i.e.,

\[
W_n = [W_{n,1}, W_{n,2}, W_{n,3}]^T,
\]

where each part \( W_{n,k} \) is a vector of size \( \frac{L}{3} \).

2) Grouping: The set of users are arbitrarily partitioned into \( \Gamma = 2 \) groups of size \( \nu \approx K + D + T = 6 \), denoted by \( G_1, G_2 \). Figure 2 represents one example of this partitioning, where \( G_1 = \{ U_1, U_2, U_3, U_4, U_5, U_6 \} \), and \( G_2 = \{ U_7, U_8, U_9, U_{10}, U_{11}, U_{12} \} \). We also order the users in each group arbitrarily. For simplicity of exposition, we may refer to user \( n \) based on its location in a group of users. If user \( n \) is the 7th user in group \( \gamma \), we call it as user \( (\gamma,t) \). For example in Fig. 2, user 9 is the 9th user as user \( (2,3) \). We use indices \( n \) or \( (\gamma,t) \) interchangeably.

3) Intra-group Secret Sharing and Aggregation: User \( n \) forms the following polynomial.

\[
F_n(x) = W_{n,1} + W_{n,2}x + W_{n,3}x^2 + \mathbb{Z}_{n,1}x^3 + \mathbb{Z}_{n,2}x^4.
\]

Let \( \alpha_t \in \mathbb{F} \), \( t \in [6] \), be six distinct non-zero constants. We assign \( \alpha_t \) to user \( t \) of all groups, i.e., users \( (\gamma,t) \), \( \gamma = 1, 2 \).

In this step, each user \( (\gamma,t) \) sends the evaluation of its polynomial function at \( \alpha_t \), i.e., \( F_{(\gamma,t)}(\alpha_t) \), to user \( (\gamma,t') \), for \( t' \in [6] \). For example, in Fig. 2, user \( (2,1) \), which is indeed \( U_7 \), sends \( F_{(2,1)}(\alpha_1) \), \( F_{(2,1)}(\alpha_2) \), \( F_{(2,1)}(\alpha_3) \), \( F_{(2,1)}(\alpha_4) \), \( F_{(2,1)}(\alpha_5) \), \( F_{(2,1)}(\alpha_6) \), to user \( (2,1) \) (or user \( U_7 \) which is basically itself), user \( (2,2) \) (or user \( U_8 \)), user \( (2,3) \) (user \( U_9 \)), user \( (2,4) \) (user \( U_{10} \)), user \( (2,5) \) (user \( U_{11} \)), and user \( (2,6) \) (user \( U_{12} \)), respectively. If a user \( (\gamma,t) \) drops out and stays silent, \( F_{(\gamma,t)}(\alpha_t) \) is just presumed to be zero.

The 7th user, \( t \in [6] \), in \( G_1 \) calculates

\[
Q_{(1,t)}(t) = F_{(1,1)}(\alpha_t) + F_{(1,2)}(\alpha_t) + F_{(1,3)}(\alpha_t) + F_{(1,4)}(\alpha_t) + F_{(1,5)}(\alpha_t) + F_{(1,6)}(\alpha_t),
\]

and the 7th user in \( G_2 \) calculates

\[
Q_{(2,t)}(t) = F_{(2,1)}(\alpha_t) + F_{(2,2)}(\alpha_t) + F_{(2,3)}(\alpha_t) + F_{(2,4)}(\alpha_t) + F_{(2,5)}(\alpha_t) + F_{(2,6)}(\alpha_t).
\]

In this example, assume that \( U_3 \) or user \( (1,3) \) drops out and does not send its share to other users in the first group. Other users within the group treat its share as zero. In this phase, within each group, at most 30 communication take place, each of size \( \frac{L}{3} \).

4) Inter-group Communication and Aggregation: In this phase, user \( (1,t) \), \( t \in [6] \), calculates message \( S_{(1,t)} = Q_{(1,t)} \), and it to user \( (2,t) \). User \( (2,t) \), \( t \in [6] \), calculates message \( S_{(2,t)} = S_{(1,t)} + Q_{(2,t)} \), upon receiving \( S_{(1,t)} \). If user \( (2,t) \) does not receive \( S_{(1,t)} \), it also remains silent for the rest of the protocol. In this particular example that user 3 drops out, it sends no message to user 7, and thus user 9 also remains silent.

5) Communication with the Server: User \( t \) of the last group, i.e., user \( (2,t) \) sends \( S_{(2,t)} \) to the server, for
t ∈ [6]. Clearly in this example, user 9 remains silent and sends nothing (or null message ⊥) to the server.

6) Recovering the result: Let us define

\[ F(x) \triangleq \sum_{n=1}^{12} F_n(x) = \sum_{n=1}^{12} W_{n,1} + x \sum_{n=1}^{12} W_{n,2} + x^2 \sum_{n=1}^{12} W_{n,3} + x^3 \sum_{n=1}^{12} \tilde{Z}_{n,1} + x^4 \sum_{n=1}^{12} \tilde{Z}_{n,2}. \]

One can verify that \( S_{(2, t)} \), for \( t = 1, 2, 4, 5, 6 \) that are received by the server are indeed equal to \( F(\alpha_k) \), for \( t = 1, 2, 4, 5, 6 \).

Since \( F(x) \) is a polynomial function of degree 4, based on Lagrange interpolation rule the server can recover all the coefficients of this polynomial using the outcomes of 5 users which are not dropped. Thus, the server is able to recover the aggregation of local models of the surviving users, i.e., \( \sum_{n=1}^{12} W_n \) and the correctness constraint is satisfied.

In this example, the per-user communication load is \( 2L \) and the server communication load is \( \frac{5}{2} L \). In addition, the communication graph has 42 edges with 13 vertices, where because of user dropouts no communication occurs on 7 edges.

As compared to Example 1, the communication load is increased while the number of connections among users are decreased. The number of connections can be minimized by choosing \( \nu = T + D + 1 = 4 \) and having three groups, where no partitioning of local models is performed.

V. THE PROPOSED \textsc{SwiftAgg}+ SCHEME

In this section, we formally describe the proposed \textsc{SwiftAgg}+, which introduces a trade-off between the communication loads and the number of active communication links among users and the server. Consider a network consisting of one server and \( N \) users, \( U_1, U_2, \ldots, U_N \), where up to \( T \) of them are semi-honest which may collude with each other or with the server to gain some information about other users’ local models. Furthermore, up to \( D \) users may drop out, and their indices are denoted by \( D \). The operations need to be done in a finite field, which is large enough to avoid hitting the boundary in the process of aggregation. We choose a finite field \( GF(p) \) denoted by \( \mathbb{F}_p \), for some prime number \( p \). Each user \( n \) has a local model \( W_n \in \mathbb{F}_p^L \) and a set of random variables \( Z_n = \{Z_{n,j}, j \in [T]\} \) whose elements are chosen independently and uniformly at random from \( \mathbb{F}_p^T \), for some parameter \( K \in \mathbb{N} \) such that \( (T + D + K) | N \).

The server wants to recover the aggregated local models of the surviving users, i.e., \( W = \sum_{n \in [N]} Z_n W_n \), while the individual models remain private from semi-honest users and the server. To reach this goal, \textsc{SwiftAgg}+ takes the following steps.

1) Grouping: The set of \( N \) users are arbitrarily partitioned into \( \Gamma \) groups, each of size \( \nu = D + T + K \), denoted by \( G_1, G_2, \ldots, G_{\Gamma} \). In each group, the users are labeled as user 1 to user \( K + D + T \). For simplicity, we refer to user \( n \) based on its location in a group of users. Without loss of generality, we let \( \gamma = \lfloor n/\nu \rfloor + 1 \), and \( t = n \mod \nu + 1 \), and place user \( n \) on the \( t \)th location in group \( \gamma \). Alternatively, we label user \( n \) using \( (\gamma, t) \).

2) Arranging the Groups: The groups are arranged on an arbitrary tree, which is called the aggregation tree and denoted by \( F(\hat{V}, \hat{E}) \). The aggregation tree represents the flow of aggregation. In the aggregation tree, \( \hat{V} \) is a set of vertices representing groups and the server, \( \hat{V} = \{G_1, G_2, \ldots, G_{\Gamma}, \text{Server} \} \) and \( \hat{E} \) is the set of edges representing the active connections between groups. In the aggregation tree, the server is the root and it has only one child which is the last group, \( G_{\Gamma} \). For example, five different aggregation trees for an example with seven groups of users, \( G_1, G_2, \ldots, G_7 \) are shown in Fig. 3.

For group \( G_{\gamma}, \) located in one vertex of this tree, let \( G_{\gamma} \) represent the parent group of \( G_{\gamma} \), and \( F(\gamma, \text{child}) \) be the set of indices of its children groups. In a rooted tree, a descendant of group \( \gamma \) is any group \( \gamma_1 \) whose path from the root contains group \( \gamma \), and group \( \gamma_2 \) is an ancestor of group \( \gamma \) if and only if \( \gamma \) is a descendant of group \( \gamma_2 \). Let us define \( F(\gamma, \text{child}) \) as the set of indices of descendant groups of group \( \gamma \), and \( F(\gamma, \text{anc}) \) as the set of indices of ancestor groups of group \( \gamma \) (excluding the server). For example, in Fig. 3 (d), \( G_{\gamma} = G_7 \) is the parent group of \( G_5 \) and \( F(\text{child}) = \{1, 2, 3\} \). In addition, \( G_7 \) has set \( F(\text{desc}) = \{1, 2, 3, 4, 5, 6\} \) and \( G_1 \) has set \( F(\text{anc}) = \{5, 7\} \).

3) Partitioning the Local Models: User \( n \) partitions its local model into \( K \) parts, denoted by

\[ W_n = [W_{n,1}, W_{n,2}, \ldots, W_{n,K}], \]

where each part \( W_{n,k}, k \in [K] \) is a vector of size \( \frac{L}{K} \).

4) Intra-Group Secret Sharing and Aggregation: User \( n \in [N] \) forms the following polynomial.

\[ F_n(x) = \sum_{k=1}^{K} W_{n,k} x^{k-1} + \sum_{j=1}^{T} Z_{n,j} x^{K+j-1}. \]

This polynomial function is designed such that the coefficient of \( x^k \) is the \((k + 1)\)th partition of the local model, for \( k \in [0: K - 1] \). Each user \( n \) uses its polynomial \( F_n(.) \) to share its local model with other users.
Let \( \alpha_t \in \mathbb{F}_p \), for \( t \in [\nu] \), be \( \nu \) distinct non-zero constants. We assign \( \alpha_t \) to the \( t \)th user of all groups, i.e., users \((\gamma, t)\), \( \gamma = 1, \ldots, \Gamma \).

Within each group \( \gamma \), each user \((\gamma, t)\) sends the evaluation of its polynomial function at \( \alpha_t \), i.e., \( F(\gamma, t)(\alpha_t) \), to user \((\gamma', t')\), for \( t' \in [\nu] \). If a user \((\gamma, t)\) drops out and stays silent, \( F(\gamma, t)(\alpha_t) \) is just presumed to be zero.

Each user \((\gamma, t)\) calculates

\[
Q(\gamma, t) = \sum_{t' \in [\nu]} F(\gamma, t')(\alpha_t). 
\]  

(7)

Note that in this phase, within each group, at most \( \nu(\nu - 1) \) communication take place, each of size \( \frac{1}{T} \).

5) Inter-group Communication and Aggregation: In this phase, user \( t \) of group \( \gamma \) calculates a message denoted by \( S(\gamma, t) \), and sends it to the server, for \( \gamma \in [\Gamma - 1] \).

Particularly, if user \((\gamma, t)\), \( \gamma \in [\Gamma - 1] \) does not have any children groups, i.e., \( P(\gamma, child) = 0 \), it sets

\[
S(\gamma, t) = Q(\gamma, t),
\]

(8)

and sends \( S(\gamma, t) \) to user \((\gamma^+, t)\).

If \( \gamma, child \neq \emptyset \), user \((\gamma, t)\) calculates \( S(\gamma, t) \) as

\[
S(\gamma, t) = Q(\gamma, t) + \sum_{\gamma' \in \gamma, child} S(\gamma', t),
\]

(9)

upon receiving \( S(\gamma', t)(\gamma' \in \gamma, child) \). If user \((\gamma, t)\) does not receive \( S(\gamma', t) \) from at least one group in \( \gamma, child \), it also remains silent for the rest of the protocol. In this phase, at most \( \nu(\nu - 1) \) messages are communicated, each of size \( \frac{1}{T} \). Figure 4 demonstrates the intra-group secret sharing and inter-group communication in a sequential aggregation tree.

6) Communication with the Server: User \( t \) of the last group, i.e., user \((\Gamma, t)\) computes

\[
S(\Gamma, t) = Q(\Gamma, t) + \sum_{\gamma \in \gamma, child} S(\gamma, t),
\]

(10)

and sends it to the server, for \( t \in [\nu] \).

7) Recovering the Result: Having received the messages from a subset of users in \( G_\Gamma \) of size at least \( T + K \), the server can recover the aggregated local models.

Remark 5: In (6), we use ramp sharing \([17]\) to keep individual local models private, but any other arbitrary \( T \)-private MDS code can be used for \( F_n(x) \). Ramp sharing allows us to reduce the size of shares and consequently achieve the cut-set outer bound within a diminishing gap.

Remark 6: The formation of the groups and the topology of the aggregation tree in \textit{SwiftAgg+} may be constrained by the actual connectivity in the network.

Next, we demonstrate the correctness of \textit{SwiftAgg+} in aggregating user models, by showing that the server can recover \( \sum_{n \in [N] \setminus D} W_n \) from the messages received from group \( \Gamma \). Using the recursive (8) to (10), \( S(\gamma, t)(\alpha_t) \) is either a null message, or it is equal to

\[
S(\gamma, t)(\alpha_t) = \sum_{k=1}^{K} \alpha_t^{k-1} \sum_{\gamma' \in \gamma, desc} \sum_{n \in \gamma', D} W_{n,k} + \sum_{j=1}^{T} \alpha_t^{K+j-1} \sum_{\gamma' \in \gamma, desc} \sum_{n \in \gamma', D} Z_{n,j},
\]

(11)

Thus if user \( t \) in group \( \Gamma \) sends a message to the server, it is equal to \( S(\Gamma, t)(\alpha_t) \). From (11), it is easy to see that \( S(\Gamma, t)(\alpha_t) = F(x) \), where

\[
F(x) = \sum_{k=1}^{K} \sum_{n \in [N] \setminus D} W_{n,k} + \sum_{j=1}^{T} \sum_{n \in [N] \setminus D} Z_{n,j}.
\]

(12)

\( F(x) \) is a polynomial of degree \( K + T - 1 \). Thus if the server receives at least \( T + K \) messages from the last group, it can use Lagrange interpolation to recover \( F(x) \).

VI. THEORETICAL ANALYSIS

In this section, we analyze the communication and computation complexities of the proposed \textit{SwiftAgg+} scheme, and provide an information-theoretic proof on its privacy guarantee in protecting local model information from colluding users and the server.

A. Communication Loads and Number of Active Connections

According to (12), the total number of symbols that are needed to be received by the server is \( T + K \), each of size \( \frac{1}{T} \). Thus, the normalized server communication load in \textit{SwiftAgg+} is \( F_{server}^{(L)} = 1 + \frac{T}{K} \).

In each group, at most \( \nu(\nu - 1) \) symbols are sent by the members, each of size \( \frac{1}{T} \), and there are \( \frac{N}{\nu} \) groups. In addition, at most \( \nu \) symbols are sent from a child group to a parent group. Thus, the normalized per-user communication load in \textit{SwiftAgg+} is upper-bounded as \( R_{user}^{(L)} \leq (1 + \frac{T+D}{K}) \), and the number of active communication links in the communication graph of \textit{SwiftAgg+}, i.e., \( |E_{SwiftAgg+}| \), is upper bounded by

\[
\frac{N}{\nu} \left( \frac{\nu(\nu-1)}{2} + \nu \right) = \frac{N}{\nu} (K + T + D + 1).
\]

It is clear that through adjusting the value of \( K \) between 1 and \( N - D - T \), we can have a trade-off between the communication loads and the number of active connections.
in the network. One choice of interest is $K = N - D - T$ as the largest value for $K$, which leads to Theorem 1 and minimizes the communication loads. It is also possible to minimize the number of network connections by choosing $K = 1$, as proposed in [14].

While the communication loads and the number of active communication links are determined by the value of $K$ using SwiftAgg+, the actual communication delay of the model aggregation also depends on the topology of the adopted aggregation tree. Assume that each communication link between two groups has maximum delay $\delta_{\text{inter}}$. In the case of more than one group, the total delay of each training iteration, $\Delta$, can be minimized if $F(\Gamma_{\text{child}}) = \{\gamma_1, \gamma_2, \ldots, \Gamma - 1\}$. Thus, in a synchronous system $\Delta = 2\delta_{\text{inter}} + \delta_{\text{intra}}$, where $\delta_{\text{intra}}$ is the delay for intra-group communications. This case is shown for an example consisting of 7 groups in Fig. 3(e). In contrast, the maximum delay occurs when the groups are located sequentially in the aggregation tree which is shown for the example in Fig. 3(a), and $\Delta = \Gamma \delta_{\text{inter}} + \delta_{\text{intra}}$.

B. Computation Loads

In this subsection, we analyze the computation loads at the server and at each user with respect to the maximum number of semi-honest users, $T$, maximum number of dropouts, $D$, and size of the local models, $L$, and the design parameter $1 \leq K \leq N - T - D$ that is the number of sub-vectors each local model is broken into.

1) **Computational Complexity at the Server**: In SwiftAgg+, to recover the aggregation of local models, the server needs to receive $T + K$ outcomes of the users in the last group, each of size $\frac{L}{K}$. Therefore, the computation performed by the server includes interpolation of a polynomial of degree $T + K - 1$, where the coefficients are vectors of size $\frac{L}{K}$. The complexity of interpolation of a polynomial of degree $d$ is $O(d \log^2 d)$, when the field supports FFT [18]. Thus, the computational complexity at the server is $O((1 + \frac{T + 2}{K})L \log^2 (K + T - 1))$.

2) **Computational Complexity at the User**: In SwiftAgg+, each user performs the following two operations:

1) It evaluates one polynomial in $\nu = K + T + D$ distinct values. If the field supports FFT, evaluation of a polynomial function of degree $d$ has a computational complexity of $O(\log^2 d)$ [18]. In SwiftAgg+, we need to compute one polynomial of degree $T + K - 1$, where the coefficients are vectors of size $\frac{L}{K}$, at $\nu$ points. Thus, this step requires a complexity of $O((T + K + D) \frac{L}{K} \log^2 (T + K - 1))$.

2) On average each user computes the summation of $\nu + 1$ vectors of size $\frac{L}{K}$ which has a complexity of $O((T + K + D) \frac{L}{K})$.

Therefore, the computation load on each user is $O((1 + \frac{T + 2}{K})L(1 + \log^2 (T + K - 1)))$.

C. Proof of Theorem 2

In SwiftAgg+, we choose a finite field $\mathbb{F}_p$, for some prime number $p$, where $N(\ell - 1) < p \leq 2N(\ell - 1)$. According to Bertrand’s postulate [20] such prime number exists for $N(\ell - 1) \geq 1$. In terms of number of required bits, the server communication load achieved by SwiftAgg+ is as follows.

$$R_{\text{server}} \leq (1 + \frac{T}{N - T - D})[\log_2(p)]$$
$$\leq (1 + \frac{T}{N - T - D})\left(\log_2 ((\ell - 1)N) + 1\right). \quad (13)$$

**Cut-Set Lower Bound**: We know that the local models are from some joint distribution $W_1, W_2, \ldots, W_N \sim P_{W_1, W_2, \ldots, W_N}(W_1, W_2, \ldots, W_N)$. However, neither the users nor the server know about this joint distribution. On the other hand, due to the privacy conditions, the users cannot, implicitly or explicitly, learn the joint distribution and adapt their transmission schemes to the joint distribution of the local models. For any joint distribution, the server should be able to recover the aggregation of local models using the received messages from the users. For the cut separating the users from the server, the worst joint distribution occurs when the distribution of the aggregation of local models on the server becomes uniform. Therefore, in terms of the number of required bits, the cut-set lower bound for the server
communication load is derived as follows.

\[ R_{\text{server}}^{(L)} \geq \log_2 \left( (\ell - 1)N + 1 \right). \] (14)

Comparing (14) with (13) shows that for \textit{SwiftAgg+} we have

\[ \frac{R_{\text{server}}^{(\text{Achievable})}}{R_{\text{server}}^{(\text{Lower})}} \leq \left( 1 + \frac{T}{N - T - D} \right) \frac{N - T - D}{\log_2 \left( (\ell - 1)N + 1 \right)} + \left( 1 + \frac{T}{N - T - D} \right) \frac{N - T - D}{\log_2 \left( (\ell - 1)N + 1 \right)} \]

Therefore, if \( T = o(N) \) and \( N - D = O(N) \) we have

\[ \frac{R_{\text{server}}^{(\text{Achievable})}}{R_{\text{server}}^{(\text{Lower})}} - 1 \approx O \left( \frac{1}{\log_2 \left( (\ell - 1)N \right)} \right), \]

where the server communication load in \textit{SwiftAgg+} is within factor 1 of the cut-set lower bound.

D. Proof of Theorem 3

In terms of number of required bits, the per-user communication load achieved by \textit{SwiftAgg+} is as follows.

\[ R_{\text{user}} \leq \left( 1 + \frac{T + D}{N - T - D} \right) \frac{\log_2(\ell) \left( \log_2(p) \right)}{N - T - D} \leq \left( 1 + \frac{T + D}{N - T - D} \right) \log_2 \left( \left( \ell - 1 \right) N + 1 \right). \] (15)

Cut-Set Lower Bound: For uniform and independent local models, and for the cut separating each user from the rest of the network, the cut-set lower bound is as follows.

\[ R_{\text{user}}^{(L)} \geq \log_2 \ell. \] (16)

Comparing (16) with (15) shows that if \( T = o(N) \) and \( D = o(N) \) we have

\[ \frac{R_{\text{user}}^{(\text{Achievable})}}{R_{\text{user}}^{(\text{Lower})}} - \log_2 \ell N \approx O \left( \frac{\log_2 N}{N} \right), \]

where the per-user communication load in \textit{SwiftAgg+} is within factor \( \log_2 N \) of the cut-set lower bound.

E. Proof of Privacy

In this section, we prove that \textit{SwiftAgg+} satisfies the privacy constraint in (2). The privacy must be guaranteed even if the server colludes with any set \( T \subset [N] \) of at most \( T \) semi-honest users which can distribute arbitrary across the groups. At a high level, we expand the mutual information in (2) over the groups containing the semi-honest users, from the leaf to the root of the aggregation tree, and show that model privacy will be preserved at each expansion step.

In \textit{SwiftAgg+}, the \textit{Intra-group Secret Sharing} is inspired by ramp secret sharing scheme [17]. Ramp secret sharing scheme is proposed to reduce the size of shares in Shamir’s secret sharing [15]. It proposes a trade-off between security and size of the shares. In ramp sharing scheme with \( T \)-privacy, no information can be leaked from any \( T \) or less shares. This directly leads to the following corollary for \textit{SwiftAgg+}.

\textbf{Corollary 1:} Assume that user \( U_n \) is denoted by \((\gamma, t)\). In \textit{SwiftAgg+}, the local model of \( U_n \) is shared using polynomial function \( F_n(x) \) in (6). In other words, \( F_n(\gamma, t) (\alpha_{t'}) \) for \( t' \in [\nu]\backslash \{t\} \) are delivered to user \((\gamma, t')\). According to (6) and directly from the privacy guarantee in ramp sharing, we have \( I(W_n; \{F_n(\alpha_{t'}), t \in T\}) = 0 \).

Let the random part of inter-group message \( S(\gamma, t) \) of user \((\gamma, t)\), consisting of the random noises of non-dropped and honest descendant, be denoted by \( Z(\gamma, t) \), i.e.,

\[ Z(\gamma, t) \triangleq \sum_{j=1}^{T} \alpha_{t'}^{K+j-1} \sum_{\gamma \in \{H^c(\gamma, \text{desc})\} \cap \{\text{D} \cup T\}} \sum_{n \in H^c(\gamma, \text{desc})} Z_{n, j}. \] (17)

We show in the following lemma, that the random noise in the message sent from a child to its parent is independent of the random noise in the message sent from that parent to the ancestor group.

\textbf{Lemma 1:} For all \( \gamma \in [\Gamma] \) and \( t \in [\nu] \), and group \( \gamma^+ \) as the parent group of group \( \gamma \), we have \( I(Z(\gamma, t); Z(\gamma^+, t)) = 0 \).

\textbf{Proof:} Let us define

\[ Z_n^{(t)}(t) \triangleq \sum_{j=1}^{T} Z_{n, j} \alpha_{t'}^{K+j-1}, \] (18)

for \( n \in [N] \) and \( t \in [\nu] \). In each group, there are \( \nu = T + D + K \) users each of which uses \( T \) random vectors chosen uniformly and independently from \( F_p^K \) in its shares. In addition, we have \( Z(\gamma, t) = \sum_{\gamma \in \{H^c(\gamma, \text{desc})\} \cap \{\text{D} \cup T\}} Z_n^{(t)}(t) \). Since \( \gamma \notin \{\text{D} \cup T\} \neq \emptyset \), and the random vectors \( \{Z_n^{(t)}(t)\} \in \gamma, t \) are i.i.d., we have that

\[ I(Z(\gamma, t); Z(\gamma^+, t)) = 0 \] for \( \gamma \in \{H^c(\gamma, \text{desc})\} \),

\[ \square \]

Assume that the semi-honest users are denoted by \( \{U_1, U_2, \ldots, U_T\} \). We denote the indices of these semi-honest users as \((\gamma_1, t_1), (\gamma_2, t_2), \ldots, (\gamma_T, t_T)\) respectively. We also denote the set of indices of honest users in group \( \gamma, \gamma \in [\Gamma] \), by \( H^{\gamma} \triangleq \{n: U_n \in \gamma \} \), where \( \gamma \neq \emptyset \). We define the set of messages which are received by \( U_{\gamma} \) by \( M_{\gamma}^{\gamma} \), which consists of two kinds of messages. Particularly, \( M_{\gamma}^{\gamma} = \{F(n, t), n \in H_{\gamma}, \{S(\gamma_1, t_1), t_1 \in \{H(\gamma_1, \text{child})\} \} \} \), where \( \{F(n, t), n \in H_{\gamma}\} = \{W_n + Z(n, t), n \in H_{\gamma}\} \), and \( \{S(\gamma_1, t_1) = \sum_{\gamma \in \{H^c(\gamma, \text{desc})\} \cap \{H(\gamma_1, \text{child})\}} \sum_{n \in H_{\gamma}} W + Z(n, t) \}, \) is the message received from the child group \( \gamma \).

According to the ramp secret sharing scheme, definitions in (17), (18), and the fact that the random vectors are chosen uniformly and independently at random from \( F_p^K \), we can easily prove the following lemmas.

\textbf{Lemma 2:} For each user \((\gamma_1, t_1) \in \textit{SwiftAgg+} \) we have \( I(Z(\gamma_1, t_1), t_1 \in [\nu] \cap H_{\gamma_1}) = 0 \) for all \( t_1 \in [\nu] \) and \( \gamma_1 \in [\Gamma] \).

\textbf{Lemma 3:} Consider user \((\gamma_1, t_1) \) and user \((\gamma_2, t_2) \). Then, \( I(Z(\gamma_1, t_1), n \in H_{\gamma_1}); (Z(\gamma_2, t_2), n \in H_{\gamma_2}) = 0 \) for \( \gamma_1 \neq \gamma_2, t_1, t_2 \in [\nu] \).

\textbf{Lemma 4:} For any \( \gamma_1 \in [\Gamma] \) consider user \( n \in H_{\gamma_1} \). Then for all \( t_1 \in [\nu], I(Z(\gamma_1, t_1); (Z(\gamma_1, t_1), t \in [T']) = 0 \), where \( T' \subset [\nu]\backslash \{t_1\} \), and \( |T'| \leq T - 1 \). Similarly, \( I(Z(\gamma_1, t_1); (Z(\gamma_1, t), t \in [T')] = 0 \).
Let us define $W_{N,T} \triangleq \{W_n, n \in [N] \setminus T\}$, $K_{N,T} \triangleq \{(W_k, Z_k, k \in T), \sum_{n \in [N] \setminus \{D \cup T\}} W_n \}$, and $M_T \triangleq \bigcup_{T \in \mathcal{T}} M_{T_i}$. In addition, $S_T \triangleq \{S_{(T,t), t \in [u] \setminus D}\}$ represents the set of messages that the server receives from users in group $T$. According to privacy constraint in (2), we have to show that

$$I(W_{N,T}; M_T, S_T) \sum_{n \in [N] \setminus \{D \cup T\}} W_n, \{W_k, Z_k, k \in T\}) = 0. \quad (19)$$

**Lemma 5:** Let $(\gamma_1, t_1), (\gamma_2, t_2), \ldots, (\gamma_T, t_T)$ be $T$ semi-honest users, where $\gamma_i \in [T]$ and $t_i \in [u]$ for $i \in [T]$. Then, for $i \in [T]$ we have $I(W_{N,T}; M_{\gamma_i}, K_{N,T}, \{M_{\gamma_i}, j \in J_i\}) = 0$, where $J_i = \{j : \gamma_j \in F(\gamma_i, \text{child})\}$, for $j \in [T]$.

**Proof:** Consider the semi-honest user in group $\gamma_i, i \in [T]$. If $J_i = \emptyset$, then we have

$$I(W_{N,T}; M_{\gamma_i}, |K_{N,T}, \{M_{\gamma_i}, j \in J_i\}) = 0.$$ 

The last term follows from the fact that both terms have the same size and uniform vectors maximize the entropy. Therefore, $I(W_{N,T}; M_{\gamma_i}, |K_{N,T}) = 0$.

If $J_i \neq \emptyset$, then we have

$$I(W_{N,T}; M_{\gamma_i}, |K_{N,T}, \{M_{\gamma_i}, j \in J_i\})$$

$$= I(W_{N,T}; F(n,t_i), n \in H_{\gamma_i}),$$

$$\{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\} | K_{N,T}$$

$$\{F(n,t_i), n \in H_{\gamma_i}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\}$$

$$= I(W_{N,T}; F(n,t_i), n \in H_{\gamma_i}, \{F(n,t_i), n \in H_{\gamma_i}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\}$$

$$+ I(W_{N,T}; F(n,t_i)), n \in H_{\gamma_i}, \{F(n,t_i), n \in H_{\gamma_i}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\}$$

$$= -H\left(\{F(n,t_i), n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{F(n,t_i), n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

$$+ H\left(\{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\} \mid K_{N,T}, \{F(n,t_i), n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

$$\leq H\left(\{F(n,t_i), n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

$$- H\left(\{F(n,t_i), n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{Z_{n,t_i}, n \in H_{\gamma_i}\}, \{Z_{n,t_i}, n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

where in (a) the first and the second terms follow from the fact that $H(X|Y) \leq H(X)$.

Now we show that

$$I\left(\{Z_{n,t_i}, n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{Z_{n,t_i}, n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i), \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right) = 0. \quad (22)$$

From definition of mutual information, we have

$$I\left(\{Z_{n,t_i}, n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{Z_{n,t_i}, n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

$$= H\left(\{Z_{n,t_i}, n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{Z_{n,t_i}, n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

$$- H\left(\{Z_{n,t_i}, n \in H_{\gamma_i}\} \mid W_{N,T}, K_{N,T}, \{Z_{n,t_i}, n \in H_{\gamma_i}\}, \{S_{(\gamma_i,t_i)}, \gamma'_i \in F(\gamma_i, \text{child})\}, j \in J_i\right)$$

where the last equality holds due to Lemma 2, Lemma 3, Lemma 4, and the independence of local models and random vectors.
Lemma 6: For $\gamma_i \in [\Gamma]$, consider $Z'_\mathcal{H}_i = \{Z^{(t)}_n, n \in \mathcal{H}_i, t \in T\}$, where $\mathcal{H}_i$ is a subset of $\{H_j, j \in \mathcal{F}_{(\gamma_i, \text{desc})}\}$ of size up to $T - 1$, and $|\mathcal{T}| \leq T - 1$. Then, $I(\mathcal{Z}_{(\gamma_i,t)}; Z'_\mathcal{H}_i) = 0$.

Proof: We can consider two cases: (I) If there is at least one group like $\gamma_i' \in \mathcal{F}_{(\gamma_i, \text{desc})}$ such that $\mathcal{H}_{\gamma_i'} \notin \mathcal{H}_i$ then we can conclude that $I(\mathcal{Z}_{(\gamma_i,t)}; Z'_\mathcal{H}_i) = 0$. The reason is that there is a non-empty set of honest and non-dropped users that $\tilde{Z}_{(\gamma_i,t)}$ includes a summation of their i.i.d. random vectors, (II) If $\mathcal{H}_i = \{H_j, j \in \mathcal{F}_{(\gamma_i, \text{desc})}\}$, then based on ramp secret sharing we have $I(\mathcal{Z}_{(\gamma_i,t)}; Z'_\mathcal{H}_i) = 0$.

Similar to (22), using Lemma 1, Lemma 2, Lemma 4, and Lemma 6 we can proof

$$I\left(\mathcal{Z}_{(\gamma_i,t)}, \gamma'_i \in \mathcal{F}_{(\gamma_i, \text{child})}; W_N, \mathcal{K}_N, T, \mathcal{Z}^{(t)}_n, n \in \mathcal{H}_i\right),$$

$$= 0. \quad \text{(23)}$$

Using (22) and (23), (21) can be written as

$$H(\mathcal{F}_{(n,t)}, n \in \mathcal{H}_i) + H(\mathcal{S}_{(\gamma_i,t)}, \gamma'_i \in \mathcal{F}_{(\gamma_i, \text{child})})$$

$$- H(\mathcal{Z}^{(t)}_n, n \in \mathcal{H}_i) - H(\mathcal{Z}_{(\gamma_i,t)}, \gamma'_i \in \mathcal{F}_{(\gamma_i, \text{child})}) \leq 0,$$

where the last term follows from the fact that uniform variables maximize entropy.

Thus, $I\left(\mathcal{W}_N; \mathcal{M}_G, \mathcal{K}_N, T, \mathcal{M}_U, j \in \mathcal{J}_i\right) = 0$, where $\mathcal{J}_i = \{j; \gamma_j \in \mathcal{F}_{(\gamma_i, \text{desc})}\}$ for $j \in [T]$.

According to Lemma 5, (19) can be written as follows.

$$I\left(\mathcal{W}_N; \mathcal{M}_G, \mathcal{K}_N, T, \mathcal{M}_U, j \in \mathcal{J}_i\right)$$

$$= I(\mathcal{W}_N; \mathcal{M}_G, \mathcal{K}_N, T, \mathcal{M}_U, j \in \mathcal{J}_i)$$

$$+ \sum_{t=1}^{T} I(\mathcal{W}_N; \mathcal{M}_G, \mathcal{K}_N, T, \mathcal{M}_U, j \in \mathcal{J}_i)$$

$$= H(\mathcal{Z}_{(\gamma_i,t)}, t' \in [\mathcal{V}] \{\mathcal{T} \cup \mathcal{D}\}), \mathcal{K}_N, T, \mathcal{M}_T$$

$$= H(\mathcal{Z}_{(\gamma_i,t)}, t' \in [\mathcal{V}] \{\mathcal{T} \cup \mathcal{D}\}), \mathcal{K}_N, T, \mathcal{M}_T$$

$$= H(\mathcal{Z}_{(\gamma_i,t)}, t' \in [\mathcal{V}] \{\mathcal{T} \cup \mathcal{D}\}), \mathcal{K}_N, T, \mathcal{M}_T$$

$$= 0. \quad \text{(24)}$$

where $\mathcal{R}_T \triangleq \bigcup_{t \in [T]} \mathcal{R}_{U_t}$, and $\mathcal{R}_{U_t} \triangleq \{Z^{(t)}_n, n \in \mathcal{H}_i\}; \{\mathcal{Z}_{(\gamma_i,t)}, \gamma'_i \in \mathcal{F}_{(\gamma_i, \text{child})}\}$. Using argument similar to that in the proof of Lemma 5, independence of local models and random vectors, and according to Lemma 1, Lemma 4 and Lemma 6, both terms in (24) are equal to $H(\mathcal{Z}_{(\gamma_i,t)}, t' \in [\mathcal{V}] \{\mathcal{T} \cup \mathcal{D}\})$ and the result is 0. Therefore, the privacy constraint is satisfied, i.e.,

$$I(\mathcal{W}_N; \mathcal{M}_G, \mathcal{K}_N, T, \mathcal{M}_U, j \in \mathcal{J}_i) = 0.$$

VII. CONCLUSION

In this paper, we propose SwiftAgg+, a secure aggregation protocol for model aggregation in federated learning, which achieves a trade-off between the communication loads and network connections. Via partitioning the users into groups, careful designs of intra and inter group secret sharing and aggregation method, SwiftAgg+ is able to achieve correct aggregation in presence of up to $D$ dropout users, with the worst-case security guarantee against up to $T$ users colluding with a curious server. Moreover, SwiftAgg+ has a flexibility to control the delay of each training iteration by choosing the aggregation tree with different depth. Compared with previous secure aggregation protocols, SwiftAgg+ significantly slashes the communication load. For $T = o(N)$ and $N - D = O(N)$, in terms of the number of required bits, the server communication load in SwiftAgg+ is within factor of the cut-set lower bound. In addition, in the case of local models with uniform distributions, the per-user communication load in SwiftAgg+ is within factor of the cut-set lower bound as long as $T = o(N)$ and $N = D = o(N)$.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful comments which improve this manuscript.

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 Proc. 20th Int. Conf. Artif. Intell. Statist. (Proceedings of Machine Learning Research), vol. 54. Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pp. 1273–1282.

[2] P. Kairouz et al., “Advances and open problems in federated learning,” 2019, arXiv:1912.04977.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” IEEE Signal Process. Mag., vol. 37, no. 3, pp. 50–60, May 2020.

[4] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in Advances in Neural Information Processing Systems, vol. 32, H. Wallach and H. Larochelle and A. Beygelzimer and F. d’Alché-Buc and E. Fox and R. Garnett, Eds. Curran Associates, 2019.

[5] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients—how easy is it to break privacy in federated learning?” 2020, arXiv:2003.14053.

[6] K. Bonawitz et al., “Practical secure aggregation for privacy-preserving machine learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1175–1191.

[7] J. So, B. Guler, and A. S. Avestimehr, “Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning,” IEEE J. Sel. Areas Inf. Theory, vol. 2, no. 1, pp. 479–489, Mar. 2021.

[8] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, “Secure single-server aggregation with (poly)logarithmic overhead,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2020, pp. 1253–1269.

[9] B. Choi, J.-Y. Sohn, D.-J. Han, and J. Moon, “Communication-computation efficient secure aggregation for federated learning,” 2020, arXiv:2012.05433.

[10] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, “Lagrange coded computing: Optimal design for resiliency, security, and privacy,” in Proc. 22nd Int. Conf. Artif. Intell. Statist., 2019, pp. 1215–1225.

[11] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, “FastSec-Agg: Scalable secure aggregation for privacy-preserving federated learning,” 2020, arXiv:2009.11248.

[12] J. So et al., “LightSecAgg: A lightweight and versatile design for secure aggregation in federated learning,” Proc. Mach. Learn. Syst., vol. 4, pp. 694–720, Apr. 2022.

[13] Y. Zhao and H. Sun, “Information theoretic secure aggregation with user dropouts,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jul. 2021, pp. 1124–1129.

[14] T. Jahani-Nezhad, M. A. Maddah-Ali, S. Li, and G. Caire, “SwiftAgg: Communication-efficient and dropout-resistant secure aggregation for federated learning with worst-case security guarantees,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2022, pp. 103–108.

[15] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[16] R. Schlegel, S. Kumar, E. Rosnes, and A. G. J. Aamodt, “CodedPaddedFL and CodedSecAgg: Straggler mitigation and secure aggregation in federated learning,” 2021, arXiv:2112.08909.
[17] G. R. Blakley and C. Meadows, “Security of ramp schemes,” in Advances in Cryptology, G. R. Blakley and D. Chaum, Eds. Berlin, Germany: Springer, 1985, pp. 242–268.
[18] K. Kedlaya and C. Umans, “Fast polynomial factorization and modular composition,” SIAM J. Comput., vol. 40, no. 6, pp. 1767–1802, 2011.
[19] K. Bonawitz et al., “Practical secure aggregation for federated learning on user-held data.” 2016, arXiv:1611.04482.
[20] M. Aigner and G. M. Ziegler, Proofs From the Book, vol. 1. Berlin, Germany: Springer, 1999.

Tayyebah Jahani-Nezhad (Member, IEEE) received the B.Sc. degree in electrical engineering and the M.Sc. degree in communication systems from the Isfahan University of Technology, Iran, in 2015 and 2017, respectively, and the Ph.D. degree in communication systems from the Sharif University of Technology, Iran, in 2022. She is currently a Post-Doctoral Researcher with the Communications and Information Theory Chair (CommIT), Technische Universität Berlin, Germany. Her research interests include distributed computing and learning protocols using information theory and coding techniques.

Mohammad Ali Maddah-Ali (Fellow, IEEE) received the B.Sc. degree in electrical engineering from the Isfahan University of Technology, the M.A.Sc. degree in electrical engineering from the University of Tehran, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2007. From 2007 to 2008, he was with Wireless Technology Laboratories, Nortel Networks, Ottawa, ON, Canada. From 2008 to 2010, he was a Post-Doctoral Fellow at the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley. From September 2010 to September 2020, he was a Communication Research Scientist at Nokia Bell Labs, Holmdel, NJ, USA. He is currently working as an Associate Professor with the University of Minnesota Twin Cities. He was a recipient of the NSERC Post-Doctoral Fellowship in 2007, the Best Paper Award from the IEEE International Conference on Communications (ICC) in 2014, the IEEE Communications Society and the IEEE Information Theory Society Joint Paper Award in 2015, and the IEEE Information Theory Society Joint Paper Award in 2016. He also served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2019 to 2022 and the Lead Editor for the IEEE JOURNAL ON SELECTED AREAS IN INFORMATION THEORY.

Songze Li (Member, IEEE) received the B.Sc. degree in electrical engineering from New York University, New York, NY, USA, in 2011, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2018. He is currently an Assistant Professor with the Internet of Things Thrust, The Hong Kong University of Science and Technology (Guangzhou), and an Assistant Professor at the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology (CWB). His research interests include secure, scalable, and accurate distributed computing and learning protocols, mainly focused on the areas of coded distributed computing, federated learning, and blockchains. He was a Qualcomm Innovation Fellowship Finalist in 2017. He received the Best Paper Award at NeurIPS-20 Workshop on Scalability, Privacy, and Security in federated learning.

Giuseppe Caire (Fellow, IEEE) was born in Torino in 1965. He received the B.Sc. degree in electrical engineering from the Politecnico di Torino in 1990, the M.Sc. degree in electrical engineering from Princeton University in 1992, and the Ph.D. degree from the Politecnico di Torino in 1994. He was a Post-Doctoral Research Fellow with The European Space Agency (ESTEC), Noordwijk, The Netherlands, from 1994 to 1995. He was an Assistant Professor in telecommunications at the Politecnico di Torino, an Associate Professor at the University of Parma, Italy, a Professor with the Department of Mobile Communications, Eurecom Institute, Sophia-Antipolis, France, and a Professor of electrical engineering with the Viterbi School of Engineering, University of Southern California, Los Angeles, CA, USA. He is currently an Alexander von Humboldt Professor with the Faculty of Electrical Engineering and Computer Science, Technische Universität Berlin, Germany. His main research interests include communications theory, information theory, and channel and source coding with particular focus on wireless communications. He received the Jack Neubauer Best System Paper Award from the IEEE Vehicular Technology Society in 2003, the IEEE Communications Society and Information Theory Society Joint Paper Award in 2004 and 2011, the Okawa Research Award in 2006, the Alexander von Humboldt Professorship Award in 2014, the Vodafone Innovation Prize in 2015, an ERC Advanced Grant in 2018, the Leonard G. Abraham Prize for Best IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) paper in 2019, and the IEEE Communications Society Edwin Howard Armstrong Achievement Award in 2020. He was also a recipient of the 2021 Leibniz Prize of the German National Science Foundation (DFG). He has served on the Board of Governors of the IEEE Information Theory Society from 2004 to 2007 and as an Officer from 2008 to 2013. He was the President of the IEEE Information Theory Society in 2011.