

# Privacy-preserving Decentralized Aggregation for Federated Learning

Beomyeol Jeon<sup>\*1</sup>, S. M. Ferdous<sup>\*2</sup>, Muntasir Raihan Rahman<sup>†3</sup>, and Anwar Walid<sup>4</sup>

<sup>1</sup>University of Illinois at Urbana-Champaign, <sup>2</sup>Purdue University, <sup>3</sup>Microsoft, <sup>4</sup>Nokia Bell Labs

<sup>1</sup>bj2@illinois.edu, <sup>2</sup>sferdou@purdue.edu, <sup>3</sup>Muntasir.Rahman@microsoft.com, <sup>4</sup>anwar.walid@nokia-bell-labs.com

**Abstract**—Federated learning is a promising framework for learning over decentralized data spanning multiple regions. This approach avoids expensive central training data aggregation cost and can improve privacy because distributed sites do not have to reveal privacy-sensitive data. In this paper, we develop a privacy-preserving decentralized aggregation protocol for federated learning. We formulate the distributed aggregation protocol with the Alternating Direction Method of Multiplier (ADMM) and examine its privacy weakness. Unlike prior work that use Differential Privacy or homomorphic encryption for privacy, we develop a protocol that controls communication among participants in each round of aggregation to minimize privacy leakage. We establish its privacy guarantee against an honest-but-curious adversary. We also propose an efficient algorithm to construct such a communication pattern, inspired by combinatorial block design theory. Our secure aggregation protocol based on this novel group communication pattern design leads to an efficient algorithm for federated training with privacy guarantees. We evaluate our federated training algorithm on image classification and next-word prediction applications over benchmark datasets with 9 and 15 distributed sites. Evaluation results show that our algorithm performs comparably to the standard centralized federated learning method while preserving privacy; the degradation in test accuracy is only up to 0.73%.

**Index Terms**—federated learning, secure aggregation

## I. INTRODUCTION

In various IoT and networking applications, data is collected and stored at the source, and there is an interest in utilizing distributed data to train machine learning (ML) models. However, due to transport costs, delays, and privacy concerns, the data may not be moved to a central location for processing and learning. For example, several factory sites, employing same robots, are interested in training an ML model for a robot maintenance scheduling plan that can benefit each competing factory site without central aggregation of privacy sensitive data. To improve the scheduling plan quality, the factories may want to utilize all data collected from robots at all sites. Still, they may not want to share privacy-sensitive raw data or move it to a central location. Similar use-cases arise in hospital settings, where different hospitals want to improve diagnosis accuracy by utilizing the privacy sensitive patient data of all hospitals without central aggregation.

*Federated learning* (FL) [1], [2] has been proposed as a promising framework for learning over decentralized data spanning multiple regions that would allow such decentralized

private sites to learn a global model that trains over all data across sites without central aggregation. Instead of transferring training data, each site trains identical ML models leveraging their local data. A central server aggregates these locally trained models to generate a global model and distributes it over all the sites. In FL, private local data stays local; only the trained model parameters move. Thus it provides the privacy of local data and also avoids expensive data transfer.

Although the standard FL only exchanges model parameters, an adversary can still infer privacy-sensitive information from the leaked model parameters [3]–[5]. Even with only the trained model, an attacker can infer whether a particular data item belongs to the dataset used for the ML model training (membership inference attack [3]).

Techniques to address these privacy concerns include Secure Multi-party Computation [6]–[8], Differential Privacy [9], [10], and combinations of both [11]–[13]. However such approaches incur large computation overheads, require a trusted third party for the secret key generation, or sacrifice the quality of trained models due to the introduced noise. Importantly, these approaches require a central aggregation server. If the server is compromised, the leaked ML models cause privacy risks. Besides, the central aggregator can be a single point of failure. Many devices, simultaneously uploading model updates to the server, cause TCP incast [14], which can slow down the training process.

To avoid challenges with a central server, we choose Alternating Direction Method of Multiplier (ADMM) for decentralized aggregation. Several approaches have been proposed to augment ADMM with privacy guarantees by using Differential Privacy [15]–[17] and homomorphic encryption [18]. In this work, we propose a different approach: *control a communication pattern among participants*. The main idea is that we develop a protocol to decide which group of parties should communicate in each round of aggregation to minimize privacy leakage. We establish its theoretical privacy guarantee against an honest-but-curious adversary. Our novel communication pattern generation is inspired by combinatorial block design theory [19]. To the best of our knowledge, we are the first to apply combinatorial design theory to develop group communication patterns for secure decentralized aggregation. Our secure aggregation protocol forms the basis of a federated learning algorithm that is privacy preserving and efficient. Our contributions in this paper are as follows.

- We show that classic ADMM-based aggregation algo-

<sup>\*</sup>Both authors contributed equally to this work.

<sup>†</sup>This work was done when the author was at Nokia Bell Labs.

rithm has a privacy risk against an *honest-but-curious* adversary.

- We propose a scheme that generates disjoint groups and allows communication only within a group in each ADMM iteration. This preserves privacy and linear convergence.
- We introduce *gap*, an approximate measure of privacy, which provides a trade-off between privacy and accuracy. The gap represents the number of iterations between two devices being in the same group. If the gap is at least  $k$ , ADMM can run securely for at most  $2k - 1$  iterations without revealing private local model parameters.
- We show that our desired group formation (with the gap constraints) is equivalent to a class of *resolvable balanced incomplete block design* problems in combinatorial design theory [19]. To the best of our knowledge, we are the first to explore this connection. We also propose an efficient randomized algorithm for the group formation.
- We design a FL algorithm with our secure aggregation and evaluate it on image classification and language models over the benchmark datasets. Our decentralized secure FL shows training performance comparable to the centralized FL [1] with two-round aggregation. We also analyze the trade-off between the number of communication rounds in aggregation and the estimation quality. We evaluate the performance of our proposed randomized group construction algorithm.

## II. DISTRIBUTED AVERAGING BY ADMM

We employ an optimization technique to address the privacy concern in aggregating the local model parameters. In this section, we will first formulate the aggregation problem as a standard distributed optimization problem. Then we will explore a privacy issue in the optimization formulation. First, let us define the preliminaries. The number of users (or data sites) is denoted by  $n$ . In FL, each user  $k$  has a local model parameter vector  $w_k$ <sup>1</sup>. The global model parameter is computed by averaging local parameters. Let  $x$  be a variable of equivalent dimension of  $w_k$ . Then we can formulate an optimization problem as follows.

$$\text{minimize} \quad \sum_{k=1}^n \|(x - w_k)\|_2^2 \quad (1)$$

The optimal solution of this unconstrained least square problem is the average, i.e.,  $x^* = \frac{1}{n} \sum_{k=1}^n w_k$ . To solve this problem in a distributed way, we follow the standard distributed consensus technique described in [20]. We introduce local variables  $x_k$  for each user  $k \in [n]$  and a consensus variable  $z$ . An equivalent optimization problem of Equation 1 is as follows.

$$\begin{aligned} & \text{minimize} && \sum_{k=1}^n \|(x_k - w_k)\|_2^2 && (2) \\ & \text{subject to} && x_k = z, \quad k = 1, \dots, n. \end{aligned}$$

We compute the augmented Lagrangian  $L_\rho(x_k, \lambda_k, z)$  as,

$$\sum_{k=1}^n (\|x_k - w_k\|_2^2 + \lambda_k^T (x_k - z) + \frac{\rho}{2} \|x_k - z\|^2) \quad (3)$$

Here,  $\lambda_k$  is a dual variable defined for each user, and  $\rho \in \mathbb{R}$  is a penalty parameter.

Following the standard ADMM technique as in [20], the  $x_k$ -minimization, the  $z$ -minimization, and  $\lambda_k$  updates can be written as follows.

$$x_k^i = \frac{1}{2 + \rho} (2w_k - \lambda_k^{i-1} + \rho z^{i-1}) \quad (4)$$

$$z^i = \frac{1}{n} \sum_{k=1}^n (x_k^i + \frac{1}{\rho} \lambda_k^{i-1}) \quad (5)$$

$$\lambda_k^i = \lambda_k^{i-1} + \rho \cdot (x_k^i - z^i) \quad (6)$$

Here  $i$  is an iteration counter.

### A. The ADMM-based Aggregation Algorithm

---

#### Algorithm 1 ADMM-based Distributed Averaging

---

```

Initialize  $\lambda_k^0$  for each user  $k$ 
 $z^0 = 0$ 
for iteration  $i = 1, 2 \dots$  do  $\triangleright$  Until stopping criteria met
  for all user  $k$  in parallel do
     $x_k^i = \frac{1}{2+\rho} (2w_k - \lambda_k^{i-1} + \rho z^{i-1})$   $\triangleright x_k$  minimization
    Send  $y_k^i = x_k^i + \frac{1}{\rho} \lambda_k^{i-1}$  to all
     $z^i = \frac{1}{n} \sum_{k=1}^n y_k^i$   $\triangleright z$  minimization
     $\lambda_k^i = \lambda_k^{i-1} + \rho \cdot (x_k^i - z^i)$   $\triangleright \lambda_k$  update
  end for
end for

```

---

Algorithm 1 shows the ADMM-based distributed averaging. Each user  $k$  initializes its dual variable  $\lambda_k^0$  randomly and sets the consensus variable  $z^0$  to zero. In iteration  $i$ , user  $k$  calculates  $x_k^i$  by minimizing the augmented Lagrangian using  $z^{i-1}$  and  $\lambda_k^{i-1}$  (Equation 4). Then users communicate each other to update the consensus variable  $z^i$ . Finally, the user  $k$  updates its dual variable  $\lambda_k^i$  with the updated  $x_k^i$  and  $z^i$ . The iterations continue until the stopping criteria are met.

### B. Proof of Convergence

Lemma II.1 shows that, after the first iteration of Algorithm 1, the summation of the all the dual variables becomes a zero vector.

**Lemma II.1.**  $\forall i \geq 1, \sum_{k=1}^n \lambda_k^i = 0$

<sup>1</sup>For simplicity, a vector is used but this generalizes to a tensor.

*Proof.*

$$\begin{aligned}
\sum_{k=1}^n \lambda_k^i &= \sum_{k=1}^n (\lambda_k^{i-1} + \rho \cdot (x_k^i - z^i)) \\
&= \sum_{k=1}^n (\lambda_k^{i-1} + \rho \cdot (x_k^i - \frac{1}{n} (\sum_{j=1}^n x_j^i + \frac{1}{\rho} \sum_{j=1}^n \lambda_j^{i-1}))) \\
&= 0
\end{aligned}$$

□

Now we will show that the consensus variable  $z$  converges linearly to the true average.

**Lemma II.2.** *The consensus variable converges to the average linearly at a rate of  $\frac{\rho}{\rho+2}$ .*

*Proof.* We know the optimum value of the consensus variable  $z^* = \frac{1}{n} \sum_{k=1}^n w_k$ . Let us define a residual at  $i$ -th iteration,  $\Delta z^i = z^i - z^*$ .

$$\begin{aligned}
\Delta z^i &= z^i - z^* \\
&= \frac{1}{n} (\sum_{k=1}^n x_k^i + \frac{1}{\rho} \sum_{k=1}^n \lambda_k^{i-1}) - \frac{1}{n} \sum_{k=1}^n w_k \\
&= \frac{1}{n} \sum_{k=1}^n x_k^i - \frac{1}{n} \sum_{k=1}^n w_k \\
&= \frac{1}{n} \sum_{k=1}^n (\frac{1}{2+\rho} (2w_k - \lambda_k^{i-1} + \rho \cdot z^{i-1})) - \frac{1}{n} \sum_{k=1}^n w_k \\
&= \frac{\rho}{\rho+2} (z^{i-1} - \frac{1}{n} \sum_{k=1}^n w_k) \\
&= \frac{\rho}{\rho+2} (z^{i-1} - z^*) \\
\implies \|\Delta z^i\| &\leq \frac{\rho}{\rho+2} \|\Delta z^{i-1}\|
\end{aligned}$$

□

The third line follows Lemma II.1.  $\|\Delta z^i\|$  represents  $z^i$ 's distance to  $z^*$ . Since  $\frac{\rho}{\rho+2} < 1$ , this achieves the linear convergence.

### III. PRIVACY PRESERVING AVERAGING

We see for our problem, ADMM is easily implementable in a distributed environment and has good convergence. In Algorithm 1, the privacy preservation may be assumed since participants do not directly share their private data (i.e.,  $w_k$ ). In [21], the authors claim that ADMM has some privacy guarantee. But in this section, we will show that Algorithm 1 is insufficient to preserve privacy in our threat model: a *semi-trust* or *honest-but-curious* threat model.

**Definition III.1** (Honest-but-curious Threat Model<sup>2</sup>). *In the honest-but-curious threat model, the participants are assumed to follow the protocol (honest) but simultaneously accumulate*

*all the information they have seen, e.g., the messages sent to them (curious).*

In the honest-but-curious model, a secure protocol prevents the participants from inferring the privacy-sensitive data of others even if they gather all communicated messages.

#### A. Privacy Analysis of the Classic ADMM-based Algorithm

We show that Algorithm 1 is not secure against *honest-but-curious* participants.

**Lemma III.1.** *In all-to-all communication, any honest-but-curious participant following Algorithm 1 can retrieve the privacy-sensitive data (i.e.,  $w$ ) when it collects messages for at least two consecutive iterations.*

*Proof.* Assume that a user  $k$  is a *honest-but-curious* participant and wants to know the private data of a user  $k'$ ,  $w_{k'}$ . For simplicity, suppose that the  $\rho$  value are shared among all participants. Note that the user  $k$  knows  $z^1$  and  $\rho$  but it does not know  $\lambda_{k'}^0$ . Throughout the proof, boldface terms are constant terms with respect to the user  $k$ : i.e., the user  $k$  knows the values of them.

In the first iteration, the user  $k$  receives a message from the user  $k'$ ,  $\mathbf{y}_{k'}^1$ .

$$\mathbf{y}_{k'}^1 = x_{k'}^1 + \frac{1}{\rho} \lambda_{k'}^0 \quad (7)$$

The user  $k$  knows the consensus value  $\mathbf{z}^1$ , So it can calculate  $\lambda_{k'}^1$  as follows.

$$\begin{aligned}
\lambda_{k'}^1 &= \lambda_{k'}^0 + \rho(x_{k'}^1 - \mathbf{z}^1) \\
&= \lambda_{k'}^0 + \rho(\mathbf{y}_{k'}^1 - \frac{1}{\rho} \lambda_{k'}^0 - \mathbf{z}^1) \\
&= \rho(\mathbf{y}_{k'}^1 - \mathbf{z}^1)
\end{aligned} \quad (8)$$

At this point, the user  $k$  cannot retrieve  $w_{k'}$  as Equation 7 have two unknowns. In the second iteration, the user  $k$  receives a following message from the user  $k'$ .

$$\mathbf{y}_{k'}^2 = x_{k'}^2 + \frac{1}{\rho} \lambda_{k'}^1 \quad (9)$$

Since  $k$  knows how  $x_{k'}$  is updated, it may plug the value of  $x_{k'}$  into the following equation and obtain the private data  $w_{k'}$ .

$$\mathbf{x}_{k'}^2 = \frac{1}{2+\rho} (2w_{k'} - \lambda_{k'}^1 + \rho \mathbf{z}^1) \quad (10)$$

$$\implies w_{k'} = ((2+\rho)\mathbf{x}_{k'}^2 + \lambda_{k'}^1 - \rho \mathbf{z}^1)/2 \quad (11)$$

□

<sup>2</sup>The formal treatment of the model is provided in Chapter 7 of [22].

### B. Enhancing Privacy through the Gap in Communication

We see that the privacy is not guaranteed in all-to-all communication (Lemma III.1). A natural approach would then be to limit communication among users and minimize information to infer private data. We adopt this approach and systematically analyze it.

In Algorithm 1, users communicate messages  $y_k^i = x_k^i + \frac{1}{\rho}\lambda_k^{i-1}$ , which is required to calculate the consensus variable  $z^i$ , the average of the  $y_k^i$ s. In all-to-all communication,  $z^i$  can be computed in a single communication round.

However, we observe that all-to-all communication is not required. An alternative communication pattern can be used. For instance, we can partition users into groups; users only communicate their  $y_k^i$  within a group  $g$  and compute the intermediate  $z_g^i$ , the average  $y_k^i$ s in the group. Next, they communicate the intermediate  $z_g^i$  across groups to compute the mean of  $z_g^i$ s, which turns to be the final  $z^i$ .

This communication scheme does not affect the ADMM convergence. Interestingly, this group-based communication approach provides the desired *privacy guarantee* if the groups follow a certain gap constraint.

**Definition III.2 (Gap).** *Given any two users, a gap is the number of consecutive iterations, after which they communicate their  $y$  messages.*

**Definition III.3 (Group).** *A group of users is the participants who communicate their  $y$  messages with each other in an iteration.*

We denote  $t_g$  as the gap size,  $g$  as the set of groups and  $s$  as the number of participants in a group. Note that in Algorithm 1  $t_g = 1$ ,  $|g| = 1$ , and  $s = n$ .

A gap of  $t_g$  means that no two participants can be in the same group for consecutive  $t_g$  iterations of the ADMM aggregation. Let us now analyze the ADMM algorithm with the gap and group properties. Let  $T_p$  be the number of iterations after which the privacy is revealed.

**Theorem III.2.** *Assuming  $s > \frac{t_g}{t_g-1}$ ,  $T_p = 2t_g$*

*Proof.* Let  $T$  and  $i$  be the number of ADMM iterations and an iteration counter, respectively. Since we are free to choose  $T$  (without loss of generality), we assume  $T$  is a multiple of  $t_g$ . Also let  $k$  be an *honest-but-curious* participant who wants to learn the model parameter of  $k'$ . From  $k$ 's point of view, the best case is when  $k$  and  $k'$  are part of the same group for  $T$  iterations, which provides the maximum information about  $k'$ . At iteration  $i$ ,  $k$  has 3 equations concerning  $k'$  as follows.

$$x_{k'}^i = \frac{1}{2 + \rho}(2w_{k'} - \lambda_{k'}^{i-1} + \rho z^{i-1}) \quad (12)$$

$$y_{k'}^i = x_{k'}^i + \frac{1}{\rho}\lambda_{k'}^{i-1} \quad (13)$$

$$\lambda_{k'}^i = \lambda_{k'}^{i-1} + \rho \cdot (x_{k'}^i - z^i) \quad (14)$$

This gives a total  $3T$  equations for  $T$  iterations. Let us analyze the number of unknowns in the system of these  $3T$  equations. In Equation 12,  $w_{k'}$  is unknown across all the  $T$

iterations. Apart from that, we have two more unknowns per iteration:  $x_{k'}^{i+1}$  and  $\lambda_{k'}^i$ .

Now let us turn to Equation 13. Since the gap size is  $t_g$ ,  $k$  and  $k'$  could be in the same group in at most  $\frac{T}{t_g}$  iterations. Let  $P$  be the set of these iteration counters. For  $j \in P$ ,  $k$  knows  $y_{k'}^j$  messages. This leaves  $(T - \frac{T}{t_g})$  new unknowns in Equation 13.

In Equation 14,  $\lambda_{k'}^{T+1}$  is a new variables at the  $T$ -th iteration. Summing all these, the number of total unknowns across the  $T$  iterations is  $1 + 2T + (T - \frac{T}{t_g}) + 1 = \frac{3Tt_g - T}{t_g} + 2$ .

Retrieving the private data of  $k'$  is now equivalent to find a unique solution of this system of  $3T$  linear equations. In general the unique solution exists if the number of unknowns matches to the number of equations as follows.

$$\begin{aligned} \frac{3T_p t_g - T_p}{t_g} + 2 &= 3T_p \\ \implies T_p &= 2t_g \end{aligned} \quad (15)$$

We also have  $T$  more equations concerning  $y_{k'}^i$ , the intermediate group mean of the form  $z_{g^i} = \sum_{u \in g^i} y_u^i$ . Here  $g^i$  is the group in which  $k'$  belongs at  $i$ -th iteration. When  $k$  and  $k'$  are in the same group (this happens at most in  $\frac{T}{t_g}$  iterations), there are no unknowns. In other  $(T - \frac{T}{t_g})$  iterations, the number of unknowns is equal to the size of the group that  $k'$  belongs to (i.e.,  $s$ ). So the total number of unknowns here is at least  $(T - \frac{T}{t_g})s$ . We find out the condition of  $s$  with which this system of equations becomes over-determined.

$$\begin{aligned} (T - \frac{T}{t_g})s &> T \\ \equiv s &> \frac{t_g}{t_g - 1} \end{aligned} \quad (16)$$

This holds if  $t_g > 1$ . Assuming  $s > \frac{t_g}{t_g-1}$ , the intermediate equations can be discarded from the attacker's point of view.  $\square$

## IV. THE ‘‘GROUP’’ CONSTRUCTION ALGORITHM

Figure 1 shows an example of our group construction: a set of partitions of the nine users  $\{1, \dots, 9\}$  into groups (of size  $s = 3$ ) with a gap constraint. Each of the four partitions corresponds to a communication scheme in an ADMM iteration. The members of a group (triangles) are free to communicate their  $y$  values within themselves in an iteration. These 4 partitions create a communication gap ( $t_g = 4$ ) over the ADMM aggregation. According to Theorem III.2, users in Figure 1 do not reveal privacy information if the aggregation converges in less than eight iterations ( $2t_g = 8$ ).

### A. Connection to Combinatorial Design Theory

We connect our group formation problem to a well-known problem in combinatorial design theory. We need a couple of definitions from the combinatorial block design theory.

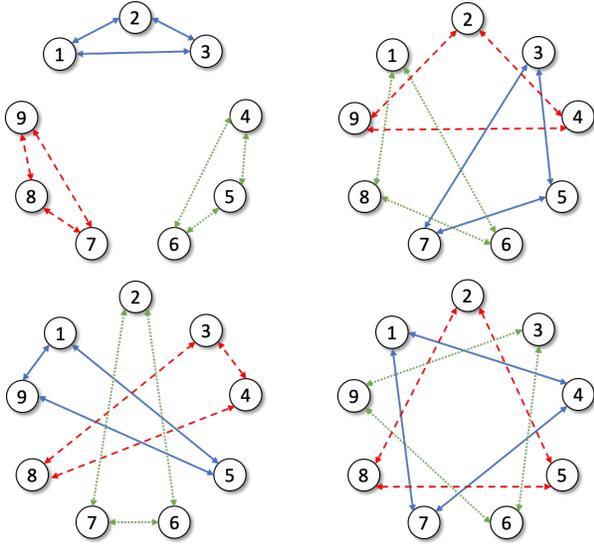


Fig. 1: Communication Group Partition Example for 9 Users

**Definition IV.1** (Balanced Incomplete Block Design (BIBD)). A balanced incomplete block design (BIBD)  $B(n, s, t)$  is a pair of  $(N, q)$  such that,

- $N$  is a set of  $n$  elements.
- $q$  is a collection of subsets of  $N$ .
- Each element of  $q$  (called blocks) is of size exactly  $s$ .
- Every pair of distinct elements of  $N$  is contained in exactly  $t$  blocks of  $q$ .

If  $t = 1$ , the blocks are pairwise disjoint. We define a parallel class as follows.

**Definition IV.2** (Parallel Classes and Resolution). A subset  $\pi \subseteq q$  is called a partial parallel class provided that the blocks of  $\pi$  are pairwise disjoint. If  $\pi$  partitions  $N$  (i.e., the intersection of the blocks are pairwise empty and the union of them covers  $N$ ), then  $\pi$  is a parallel class. The number of parallel classes in  $q$  is called a resolution.  $(N, q)$  is said to be a resolvable BIBD if  $q$  has at least one resolution.

$q$  may have more than one parallel classes, i.e., more than one partitions of  $N$ . For example, if  $N = \{1, 2, 3, 4\}$  then  $q = \{\{1, 2\}, \{3, 4\}, \{1, 3\}, \{2, 4\}\}$  contains two parallel classes (partitions) of  $N$ :  $\{\{1, 2\}, \{3, 4\}\}$  and  $\{\{1, 3\}, \{2, 4\}\}$ . The resolution of  $q$  is 2. We refer the reader to [19] for comprehensive references of block design problems.

With these definitions, we are now ready to establish a connection between our group construction and a balanced incomplete block design problem.

**Lemma IV.1.** A BIBD  $B(n, s, 1)$  with resolution  $k$  implies a gap of  $k$  with group size  $s$ .

*Proof.* Since  $t = 1$ , the blocks are pairwise disjoint. A resolution of  $k$  ensures  $k$  partitions of all users with the block size of exactly  $s$ . These blocks are groups in our construction. Each partition provides a communication scheme in each

iteration.  $\square$

Naturally we want to maximize the resolution since a large gap size allows more iterations under the privacy guarantee (Theorem III.2) and more iterations minimize estimation error (Lemma II.2).

A special case of  $B(n, s, 1)$  is when  $n \equiv 3 \pmod 6$  and  $s = 3$ . This is known as a Kirkman triple system [23]. The goal is to find  $\frac{n-3}{2} + 1$  parallel classes. Ray-Chaudhuri et al. [24] propose a solution for Kirkman triple systems. If  $s = 4$ , the corresponding design problem is known as the Social Golfer Problem [25]. The task here is to maximize the number of parallel classes for any  $n$  divisible by 4. Unfortunately, this problem still remains unsolved [26].

A constrained programming solver can be used to construct a solution. In [27], the authors propose different pruning strategies to speed up the problem-solving time. But these techniques are based on exhaustive search. They scale poorly as  $n$  increases.

Recently there have been works on the randomized analysis of combinatorial design problems [28]–[30]. Motivated by them, we design a randomized heuristic algorithm to solve a Kirkman triple system problem. Although our algorithm is not guaranteed to find partitions with the optimal resolution, it does provide partitions that guarantee the desired privacy.

#### B. Randomized Group Construction Algorithm

We intend to generate a collection of partitions  $(\Pi)$  of  $N$ . Here  $N$  is a vertex set of  $n$  users. Each of these partitions consists of pairwise disjoint groups of size 3. Before discussing our randomized algorithm, let us first state a few preliminaries.

Let  $G$  be a complete graph, and  $H$  be a 3-uniform hypergraph; both defined on the vertex set  $N$ . In graph theory, a *matching* in a hypergraph is a collection of independent sets of edges, i.e., the matching edges are vertex disjoint. A *perfect matching* is one that covers the entire vertex set. A *clique* of size  $k$  in a graph is a complete subgraph with  $k$  vertices. A  $k$ -*clique* partition of a graph is a partition of the set of vertices into cliques of size  $k$ . If  $k = 3$ , then it is called a triangle, and a corresponding partition is called a triangle partition.

We observe that a parallel class in a Kirkman triple system is a perfect matching in  $H$ . These matching edges also correspond to a triangle partition  $\pi$  of  $G$ .

Algorithm 2 shows a randomized heuristic algorithm to generate parallel classes in a Kirkman triple system. We start with the complete graph  $G$ . We randomly sample a triangle  $Q$  from  $G$ . Then we remove  $Q$  from  $N$  and all the edges of  $Q$  from  $G$ . We keep repeating this process until we find a triangle partition (i.e., when  $N$  becomes empty). In this case, we add the triangle partition  $\pi$  into  $\Pi$ , reset  $N$ , and start sampling again. We continue this process until we reach the stopping criteria.

In the algorithm, we remove the edges of  $Q$  from  $G$  when we find a triangle  $Q$ . That guarantees the desired pairwise disjointness.

Note that we can easily extend the triangle removal technique to the  $s$ -clique removal for an arbitrary  $s$ . Instead of

---

**Algorithm 2** Randomized-KirkmanTriple

---

$G$ : Complete graph with nodes  $n$   
 $N : \{1, 2, \dots, n\}$   
 $\Pi = \emptyset, \pi = \emptyset$   
**repeat**  
 $Q$ : Randomly sampled 3 vertices from  $N$  without replacement  
**if**  $Q$  forms a triangle in  $G$  **then**  $\triangleright$  A triangle is found  
 $\pi = \pi \cup Q$   
 $N = N - Q$   
Remove the edges of  $\pi$  from  $G$   
**if**  $N = \emptyset$  **then**  $\triangleright$  A partition is found  
 $\Pi = \Pi \cup \pi$   
 $\pi = \emptyset$   
 $N = \{1, 2, \dots, n\}$   $\triangleright$  Reset  $N$   
**end if**  
**end if**  
**until** Stopping criteria met

---

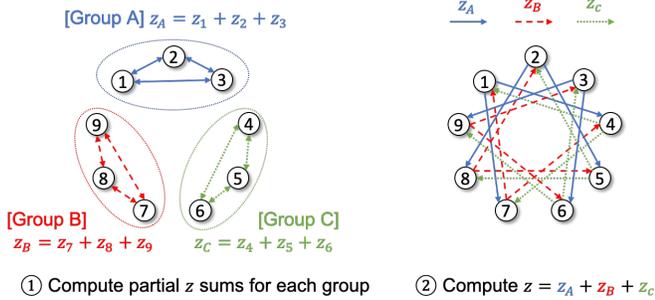


Fig. 2: A Communication Example in ADMM Aggregation

sampling a triangle, we can sample a  $s$ -clique. In Section VI, we will show the performance of the algorithm, i.e., the number of generated partitions with different  $n$  for triangles and 4-cliques.

## V. PRIVACY-PRESERVING DECENTRALIZED FEDERATED LEARNING ALGORITHM

Algorithm 3 shows our decentralized federated learning protocol (*SecureD-FL*) that preserves the privacy of each peer's local model parameters. Each peer initializes its local parameters and runs the secure ADMM-based aggregation (*SecureAggregation*) to agree on the initial global parameters. We will explain how the aggregation is performed later.

Similar to the classic federated learning, our algorithm consists of multiple rounds. In each FL round, every peer trains the model on its local dataset and updates the model parameters (*Update*). Next the peers synchronize the locally trained model parameters. Instead of sending out the trained local model parameters to a central aggregator, the peers work together to compute the average via our ADMM-based secure aggregation algorithm (*SecureAggregation*). After the aggregation finishes, all peers move to the next round. This process repeats until the model converges.

---

**Algorithm 3** Privacy-preserving Decentralized FL Algorithm

---

$N$ : a set of peers,  $n = |N|$   
Comm. pattern  $\Pi = \{\pi\}$ , a set of partitions  $\pi$  of  $N$  into groups generated by Algorithm 2

**for all** peer  $k$  **in parallel do**  
initialize  $w_k^0$   
 $w^0 = \text{SecureAggregation}(k, w_k^0)$   
**for each** FL round  $r = 1, 2, \dots$  **do**  
 $w_k^r = \text{Update}(k, w^{r-1})$   
 $w^r = \text{SecureAggregation}(k, w_k^r)$   
**end for**  
**end for**  
  
**function** *SecureAggregation*( $k, w_k$ )  
initialize  $\lambda_k^0$   
 $z^0 = \mathbf{0}$   
**for** iteration  $i = 1$  to  $I$  **do**  $\triangleright I$ : max iterations  
 $\pi$ : ( $i \bmod |\Pi|$ )-th partition in  $\Pi$   $\triangleright |\Pi|$ : gap size  
 $g$ : group in  $\pi$  such that  $k \in g$   
 $x_k^i = \frac{1}{2+\rho}(2w_k - \lambda_k^{i-1} + \rho z^{i-1})$   $\triangleright x$  minimization  
Send  $y_k^i = x_k^i + \frac{1}{\rho}\lambda_k^{i-1}$  to other peers in  $g$   
 $z_g^i = \frac{1}{n} \sum_{u \in g} y_u^i$   $\triangleright$  Partial  $z$  sum  
Send  $z_g^i$  to the other groups ( $\neq g$ )  
 $z^i = \sum_{h \in \pi} z_h^i$   $\triangleright$  Final  $z$  sum  
 $\lambda_k^i = \lambda_k^{i-1} + \rho \cdot (x_k^i - z^i)$   $\triangleright \lambda$  update  
**end for**  
**return**  $z^I$   
**end function**  
  
**function** *Update*( $k, w$ )  
**for** local epoch  $e = 1$  to  $E$  **do**  $\triangleright E$ : # local epochs  
**for** mini batch  $b \in$  local dataset  $D^k$  **do**  
 $w = w - \eta \nabla l(w; b)$   $\triangleright$  SGD  
**end for**  
**end for**  
**return**  $w$   
**end function**

---

Our secure ADMM-based aggregation consists of multiple iterations. Figure 2 shows a communication example in a single ADMM iteration. In each iteration, every peer performs the  $x$  minimization and communicates its  $y$  value with the other peers in the same group to compute the group's partial  $z$  sum. Then peers in one group send its group's partial  $z$  sum to peers in the other groups (inter-group communication) to compute the final  $z$  sum. Peers update their  $\lambda$ s with the final  $z$  sum and this concludes the iteration.

Iterations are repeated until the predefined maximum iterations  $I$ . The  $z^I$  would be an estimate of average of all peers' local model parameters.

We assume that the peers know their communication pattern generated by Algorithm 2. This can be achieved easily. For example, the same random seed can be distributed over all

TABLE I: Dataset details

Dataset	# Users	# Train Samples	# Test Samples
FEMNIST [32]	3,483	351,333	40,668
Shakespeare [32]	715	37,986	5,464
CIFAR-10 [33]	1,000	50,000	10,000

peers at the beginning. Then each peer generates a communication pattern with the same randomized group construction and follows the same communication pattern across the entire ADMM iterations. The communication pattern itself does not relate to any privacy-sensitive data. In contrast to the secret key generation in homomorphic encryption, the random seed generation and distribution can be done without any privacy concern.

## VI. EVALUATION

We implement `SecureD-FL` on top of PyTorch [31] v1.5.1 that simulates the federated learning for image classification and language model tasks. We run experiments on our local machine with an Intel i9-7960X CPU, 128 GB memory, and four NVIDIA GTX 2080 GPUs.

In the evaluation, we answer the following questions: (i) How does `SecureD-FL` influence the quality of trained models (with respect to test accuracy)? (ii) How does the number of ADMM iterations affect the estimation quality? (iii) How does our randomized group construction algorithm (Algorithm 2) generate parallel classes effectively?

### A. Experimental Setup

**Datasets.** We utilize an existing FL benchmark dataset (LEAF [32]) and generate a new federated version of CIFAR-10 [33] for a more complicated ML task. We use two datasets from LEAF [32] for image classification (FEMNIST) and next character prediction (Shakespeare). FEMNIST is a federated version of Extended MNIST [34] (10 digits and 52 alphabet characters) where the dataset is partitioned based on writers. We use only digits data and users with at least 10 samples. The Shakespeare dataset is built from the Complete Works of William Shakespeare [35] by partitioning lines in the plays based on the speaking roles. In both datasets, we split each partition into 90% training and 10% test data.

To evaluate on a large ML model, we generate a federated version of CIFAR-10 [33], a popular image classification benchmark dataset with 10 classes. Unlike the LEAF dataset, CIFAR-10 does not contain metadata to create non-i.i.d data. We randomly partition training and test samples across 1,000 clients evenly. Details about the datasets are in Table I.

**Models.** For FEMNIST, we use a convolutional neural network comprising  $5 \times 5$  and  $2 \times 2$  convolutional layers with 32 and 64 filters each of which is followed by a  $2 \times 2$  max-pooling layer, a 512-unit fully connected layer with ReLU activation, and a softmax layer. For Shakespeare, we train a recurrent neural network that consists of an embedding layer with 100 dimensions, a GRU with 128 hidden units, and a

TABLE II: Best Test Accuracy Over Training

# Peers	Datasets	Local-Only	FedAvg	SecureD-FL
9	FEMNIST	99.01%	99.52%	99.53%
	Shakespeare	50.54%	52.70%	52.71%
	CIFAR-10	57.13%	77.69%	77.33%
15	FEMNIST	98.88%	99.52%	99.53%
	Shakespeare	49.32%	51.79%	51.78%
	CIFAR-10	48.68%	76.64%	76.08%

softmax layer. We use ResNet-18 [36] for the federated version of CIFAR-10.

We use 9 and 15 peers representing distributed sites that train a ML model in a federated fashion. Since the number of peers is smaller than the number of users in the datasets, we randomly assign the same number of users in the datasets to each peer. As a result, each peer uses a collection of data for those users as its local dataset.

We compare three different methods: Local-only, FedAvg [1], and our `SecureD-FL`.

In Local-Only, every peer trains a ML model over only its local dataset without any communication.

FedAvg [1] works as follows. A central server maintains global ML model parameters. Peers receive the global model parameters from the server and perform training on local datasets for the pre-defined local epochs. Peers then send locally trained model parameters to the server. The centralized server aggregates these model parameters by averaging them. These steps, composing a single FL round, repeat until convergence.

For all the experiments, we train models for 50 FL rounds with a batch size of 32, a learning rate of 0.001, and the number of local epochs 1. We use the RMSProp [37] optimizer for FEMNIST and CIFAR-10, and the Adam [38] optimizer for Shakespeare.

For secure aggregation, peers initialize  $\lambda$  by sampling from a uniform distribution on  $[0, 1)$ . To preserve privacy in the aggregation (Section III), we create a gap in communication by using the group construction algorithm (Algorithm 2). With 9 and 15 peers, the algorithm generates 4 and 5 partitions (gaps), respectively. The gap sizes of 4 and 5 preserve the privacy of the local parameters respectively up to 7 and 9 ADMM iterations (Theorem III.2). We set the maximum ADMM iteration to 2, which not only preserves privacy ( $\leq 7$  and  $\leq 9$ ) but also makes the algorithm practical by minimizing the communication overhead while offering the estimation closer to true average. We will discuss the effect of the number of ADMM iterations in detail later (Section VI-C).

### B. Test Accuracy Comparison

Figure 3 shows test accuracy per round over all the peers for Local-Only, FedAvg, and our `SecureD-FL` with 9 and 15 peers. Table II shows the best test accuracies for different configurations in the experiments.

When peers do not synchronize each other (Local-Only), the trained models are overfitted to the local training dataset,

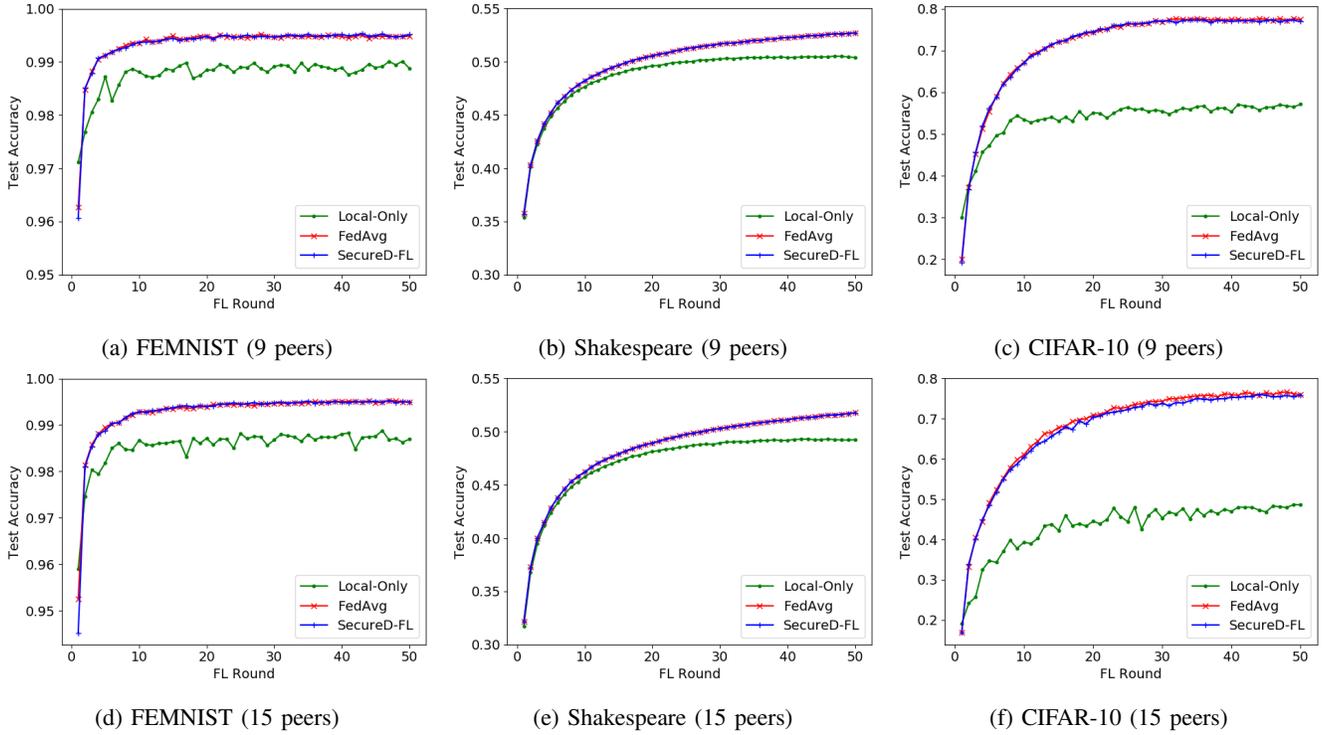


Fig. 3: Test Accuracy per Round for Training over Different Datasets with 9 and 15 Peers.

which results in lower test accuracy than FedAvg and SecureD-FL. The overfitting exacerbates with more peers due to smaller local dataset.

FedAvg and SecureD-FL show comparable test accuracies for all the datasets. For FEMNIST and Shakespeare, the test accuracy differs only less than 0.02%.

For CIFAR-10, the test accuracy decreases 0.46% for 9 peers and 0.73% for 15 peers. ResNet-18 has more parameters than the models for the other datasets. More parameters increase the accumulated estimation error, which adversely affects the trained model performance. Increasing the number of peers makes the estimation error more significant. Despite these, we do not observe any substantial degradation of test accuracy between FedAvg and SecureD-FL. In all cases, the degradation is only less than 1%.

### C. Trade-off: # ADMM Iterations vs. Estimation Quality

In this section, we explore a trade-off between the number of ADMM iterations and the quality of the estimated results. Although a large number of iterations in ADMM makes the estimated average closer to the actual average, this can increase the communication overhead and reveal privacy (Section III). We explore a sweet spot that minimizes the communication overhead, keeps the estimation error low, and still provides the privacy guarantee for local model parameters.

We compare estimated values by the ADMM-based aggregation and actual averages of model parameters in real ML training. For this, we generate 10 checkpoints of model parameters (in every five FL rounds) during the training over

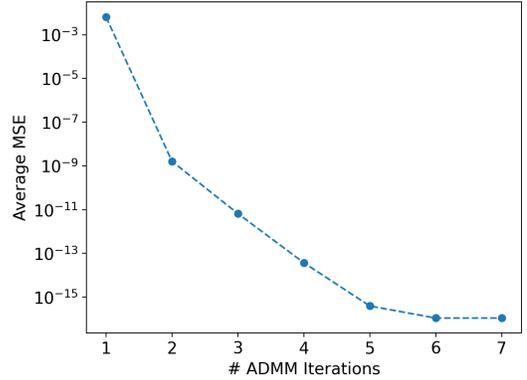


Fig. 4: Average MSE with Different # ADMM Iterations

FEMNIST by FedAvg. We run the ADMM aggregation for each checkpoint, measure mean squared error (MSE), and calculate the averaged MSE over the 10 checkpoints.

We use 9 peers for training with a gap size of 4, which allows at most seven ADMM iterations to guarantee the privacy (Theorem III.2). We vary the number of iterations in ADMM aggregations and calculates averaged MSEs. Figure 4 shows the results of the experiment. The average MSE decreases fast as the number of iterations increases. After 4 iterations, the error drops below  $10^{-13}$ .

To see how estimation error in ADMM aggregation affect test accuracy, we run another set of experiments. We perform training for the FEMNIST dataset with 9 peers with various

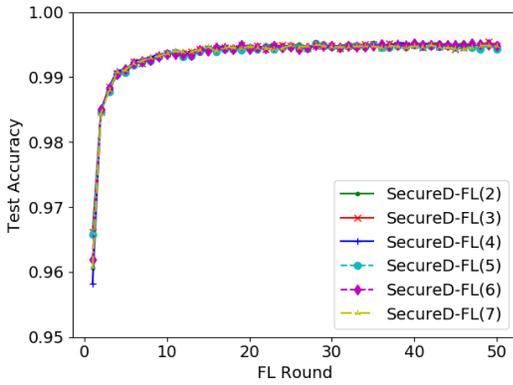


Fig. 5: Test Accuracy per FL Round with Different # ADMM Iterations (FEMNIST)

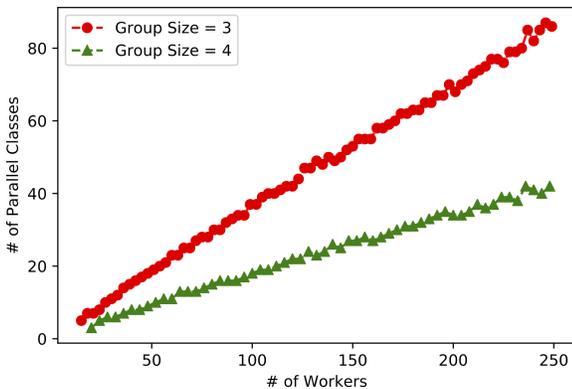


Fig. 6: Group Construction Algorithm Performance

numbers of ADMM iterations. Figure 5 shows per-round test accuracy with different numbers of ADMM iterations. For ADMM iterations greater than 1, the estimation error due to early termination does not adversely affect the test accuracy. The difference in accuracy across the ADMM iterations is negligible. So for the experiments in Section VI-B, we set the number of ADMM iterations to 2. This decreases the communication overhead and also preserves the desired privacy.

#### D. Randomized Group Construction Algorithm Performance

We evaluate the randomized group construction algorithm (Algorithm 2) to construct the communication groups. Figure 6 shows a plot of the our group construction algorithm performance. We plot the number of generated parallel classes (i.e., the number of partitions of the peers) by varying the number of peers. The red and green lines are the number of parallel classes for group sizes 3 and 4, respectively. We see that the number of generated partitions increases as the number of peers grows. This also shows the applicability of Algorithm 2 to construct groups of size more than 3.

## VII. RELATED WORK

Several approaches have been proposed to provide privacy in ADMM. In [21], the authors introduce several transformation methods to alter the objective function or constraints in a way that transmitted messages are safe under an eavesdropping attack. But as we have shown in Lemma III.1, transformation-based methods are not enough for protection against honest-but-curious attacks. Differential Privacy (DP) is also considered to preserve privacy in ADMM [15]–[17]. Although DP offers quantitative privacy guarantees, it is challenging to accurately estimate sensitivity and control the privacy-accuracy trade-off. The authors in [18] augment ADMM with expensive homomorphic encryption which we avoid.

In FL, Secure Multi-party Computation (SMC) techniques have been employed to preserve the privacy of transferring model parameters. PySyft [13] and Chen et al. [7] provide the privacy guarantee by using SPDZ [39]. Truex, et al. [11] utilizes a threshold-based additive homomorphic encryption scheme [40]. However, these approaches assume that secret keys are generated and distributed to participants securely [7], [11], [13] or a trusted third party is essential over the entire computation [12]. These are not required by our secure aggregation method. Bonawitz et al. [6] addresses the trusted third party requirement by including the secret key generation into the aggregation protocol. All these approaches to federated learning need a centralized aggregation entity.

Along with SMC, employing DP on FL has been also proposed in prior work [11]–[13], but the challenges in estimating sensitivity and privacy and model accuracy trade-off have not fully addressed. Our proposed secure aggregation algorithm is compatible with DP, which we leave as future exploration.

The authors in [41], [42] propose mechanisms for privacy preserving distributed optimization using a combination of ADMM optimization and homomorphic encryption. In comparison, our approach avoids costly homomorphic encryption. DP-ADMM [16] combines ADMM optimization with differential privacy. The authors in [17] propose a perturbation method for ADMM where the perturbed term is correlated with ADMM penalty parameters.

## VIII. CONCLUSION

In this paper, we have developed a new decentralized federated learning algorithm (*SecureD-FL*), focusing on the privacy preservation of ML models in training. We identified privacy weakness in the classical ADMM-based aggregation. To address this issue, we introduced a communication pattern (gap) that enables privacy protection of ML model parameters in the ADMM-based aggregation. We proposed a randomized algorithm to efficiently generate this communication pattern by connecting it to *balanced block design* from combinatorial mathematics. We proved the privacy guarantee of our aggregation protocol in the honest-but-curious threat model.

Our evaluation results with the benchmark datasets showed that *SecureD-FL* performed ML training comparable to the centralized standard FL method (< 0.73% degradation in test accuracy) with the privacy guarantee.

## 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 *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS '17)*, 2017, pp. 1273–1282.
- [2] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.
- [3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP '17)*, 2017, pp. 3–18.
- [4] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP '19)*, 2019, pp. 739–753.
- [5] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP '19)*, 2019, pp. 691–706.
- [6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*, 2017, pp. 1175–1191.
- [7] V. Chen, V. Pastro, and M. Raykova, "Secure computation for machine learning with spdz," in *NIPS Workshop on Privacy Preserving Machine Learning (PPML '18)*, 2018.
- [8] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón, "Quotient: Two-party secure neural network training and prediction," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*, 2019, p. 1231–1247.
- [9] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*, 2015, p. 1310–1321.
- [10] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*, 2016, pp. 308–318.
- [11] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISeC '19)*, 2019, p. 1–11.
- [12] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISeC '19)*, 2019, p. 13–23.
- [13] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," in *NIPS Workshop on Privacy Preserving Machine Learning (PPML '18)*, 2018.
- [14] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding tcp incast throughput collapse in datacenter networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking (WREN '09)*, 2009, p. 73–82.
- [15] T. Zhang and Q. Zhu, "Dynamic Differential Privacy for ADMM-Based Distributed Classification Learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 172–187, jan 2017.
- [16] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "Dp-admm: Admm-based distributed learning with differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1002–1012, 2020.
- [17] X. Zhang, M. M. Khalili, and M. Liu, "Improving the privacy and accuracy of admm-based distributed algorithms," in *International Conference on Machine Learning (ICML '18)*, 2018, pp. 5796–5805.
- [18] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2018.
- [19] D. Stinson, *Combinatorial designs: constructions and analysis*. Springer Science & Business Media, 2007.
- [20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.
- [21] P. Weeraddana and C. Fischione, "On the privacy of optimization," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9502 – 9508, 2017.
- [22] G. Oded, *Foundations of Cryptography: Volume 2, Basic Applications*, 1st ed. Cambridge University Press, 2009.
- [23] R. Abel and S. Furino, "Kirkman triple systems," *The CRC Handbook of Combinatorial Designs*, pp. 88–89, 1996.
- [24] D. K. Ray-Chaudhuri and R. M. Wilson, "Solution of kirkman's school-girl problem," in *Proceedings of Symposia in Pure Mathematics*, vol. 19, 1971, pp. 187–203.
- [25] C. J. Colbourn, *CRC handbook of combinatorial designs*. CRC press, 2010.
- [26] E. J. Pegg, "Social golfer problem," from MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein. [Online]. Available: <https://mathworld.wolfram.com/SocialGolferProblem.html>
- [27] N. Barnier and P. Brisset, "Solving kirkman's schoolgirl problem in a few seconds," *Constraints*, vol. 10, no. 1, pp. 7–21, 2005.
- [28] M. Kwan, "Almost all steiner triple systems have perfect matchings," *Proceedings of the London Mathematical Society*, vol. 121, no. 6, pp. 1468–1495, 2020.
- [29] A. Ferber and M. Kwan, "Almost all steiner triple systems are almost resolvable," *arXiv preprint arXiv:1907.06744*, 2019.
- [30] P. Keevash, "Hypergraph matchings and designs," in *Proceedings of the International Congress of Mathematicians (ICM '18)*, 2019, pp. 3113–3135.
- [31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS '19)*, 2019.
- [32] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A Benchmark for Federated Settings," in *Workshop on Federated Learning for User Privacy and Data Confidentiality (FL-NeurIPS '19)*, 2019.
- [33] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
- [34] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.
- [35] W. Shakespeare, *The complete works of William Shakespeare*, publicly available at <http://www.gutenberg.org/ebooks/100>.
- [36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, 2016, pp. 770–778.
- [37] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012. [Online]. Available: <http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf>
- [38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *the 3rd International Conference on Learning Representations (ICLR '15)*, 2015.
- [39] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology (CRYPTO '12)*, Berlin, Heidelberg, 2012, p. 643–662.
- [40] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology (EUROCRYPT '99)*, 1999, pp. 223–238.
- [41] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2019.
- [42] M. Ruan, M. Ahmad, and Y. Wang, "Secure and privacy-preserving average consensus," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy (CPS '17)*. New York, NY, USA: Association for Computing Machinery, 2017, p. 123–129.