# A Graph Federated Architecture with Privacy Preserving Learning

Elsa Rizk and Ali H. Sayed

***Abstract*—Federated learning involves a central processor that interacts with multiple agents to determine a global model. The process consists of repeatedly exchanging estimates, which may end up divulging some private information from the local agents. This scheme can be inconvenient when dealing with sensitive data, and therefore, there is a need for the privatization of the algorithm. Furthermore, the current architecture of a server connected to multiple clients is highly sensitive to communication failures and computational overload at the server. In this work, we develop a private multi-server federated learning scheme, which we call graph federated learning. We use cryptographic and differential privacy concepts to privatize the federated learning algorithm over a graph structure. We further show under convexity and Lipschitz conditions, that the privatized process matches the performance of the non-private algorithm.**

***Index Terms*—federated learning, distributed learning, differential privacy, secure aggregation, network**

## I. Introduction

Federated learning (FL) [1] is a useful distributed learning algorithm that aims at finding a global model to fit local data. The FL algorithm consists of two steps: an update step done locally at each client, and an aggregation step done at a central server. During these two steps, communication occurs between the clients and the *one* server. Unfortunately, such a structure is not robust, since it relies on one server to carry out all the communications and aggregation. One solution was suggested in [2] introducing hierarchical federated learning; the architecture consists of one cloud server connected to a number of edge servers that, in turn, are connected to multiple clients, thus forming a tree structure. In this work, we consider a more general framework and introduce *graph federated learning* (GFL), which consists of several servers each connected to their own subset of clients. The servers in turn are connected by a graph topology. Such an architecture is more suitable, for example, when considering cellular networks that consist of multiple cellphone towers, each open for communication with numerous cellular devices. However, it is not without its challenges. Introducing multiple servers and requiring them to collaborate adds some more communication/synchronization effort.

In addition, when multiple servers are used, it becomes essential to focus on the privacy of the federated learning algorithm. It is not sufficient that the raw local data is not explicitly communicated for the algorithm to be private. The model and gradient updates shared by each client can convey information about the data [3]–[6]. For example, if we consider a logistic risk function, the gradient can be expressed as a constant multiplying the feature vector. Therefore, there is a need to privatize the federated algorithm in order to stop information leakage.

Multiple solutions exist to privatize distributed learning algorithms. They can be split into two frameworks: differential privacy [7]–[15] and cryptography [16]–[20]. No framework prevails over the other. While differential privacy is easy to implement, it adds a bias to the solution. On the other hand, cryptographic methods such as secure multi-party computation (SMC) are harder to implement and impose hard limitations on the number of participating parties. Thus, in this work we wish to benefit from the two approaches. We develop a protocol based on the works in [21] and [16]. The former reference [21] utilizes differential privacy to privatize a distributed learning algorithm on a graph. Unlike standard differential privacy schemes, the perturbations are not independent but are choosen to satisfy a nullspae condition determined by the graph structure. While reference [16] encorporates multiple SMC tools into the federated learning architecture. Their scheme coresponds to adding local perturbations, which will be canceled out at the server.

In this work, we first study the effect of privatization on the performance of the learning algorithm. We study general private algorithms whose privatization schemes can be modelled as added noise, whether it be using differential privacy or SMC. We then present the protocol we adopt and specialize the results.

## II. Problem Formulation

The graph federated learning architecture consists of $P$ servers, each connected to a set of $K$ clients, as depicted in Figure 1. The graph connecting the servers is represented by a combination matrix $A \in \mathbb{R}^{P\times P}$ whose elements are denoted by $a_{mp}$. The goal is to minimize the average empirical risk:

$$w^o \triangleq \underset{w\in\mathbb{R}^M}{\operatorname{argmin}} \frac{1}{P}\sum_{p=1}^{P}\frac{1}{K}\sum_{k=1}^{K} P_{p,k}(w), \tag{1}$$

where each individual cost is a sample average of a local loss function:

$$P_{p,k}(w) \triangleq \frac{1}{N_{p,k}}\sum_{n=1}^{N_{p,k}} Q_{p,k}(w; x_{p,k,n}), \tag{2}$$

We introduce the subscript $p$ to denote the server, while the subscript $k$ refers to the client and $n$ to the data. To solve

School of Engineering, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (e-mail:{elsa.rizk, ali.sayed}@epfl.ch).

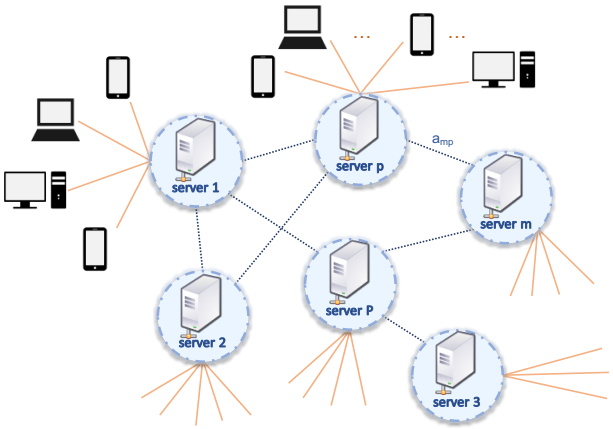


Fig. 1: Graph federated architecture.

problem (1), each server with its clients runs the federated averaging (FedAvg) algorithm [1], and then the servers amongst themselves run a consensus or diffusion-type algorithm. In our previous works [22] and [23], we have shown that when each agent runs a different number of epochs before sending their final update to the server, the resulting incremental error is on the order of $O(\mu^2)$ and is dominated by gradient noise. Let $\mathcal{L}_{p,i}$ denote the $L$ sampled clients at iteration $i$ by server $p$. To simplify the presentation, we assume in this work that the $L$ sampled clients run one stochastic gradient descent (SGD) step during each iteration. More formaly, at iteration $i$, each agent $k \in \mathcal{L}_{p,i}$ updates the model at the server $\boldsymbol{w}_{p,i-1}$ to $\boldsymbol{w}_{p,k,i}$, which they then send to server $p$:

$$\boldsymbol{w}_{p,k,i} = \boldsymbol{w}_{p,i-1} - \mu \frac{1}{B_{p,k}} \sum_{b \in \mathcal{B}_{p,k,i}} \nabla_{w^\intercal} Q_{p,k}(\boldsymbol{w}_{p,i-1}; \boldsymbol{x}_{p,k,b}), \tag{3}$$

where $\mathcal{B}_{p,k,i}$ is the mini-batch sampled by client $k$, connected to server $p$, at iteration $i$ and of size $B_{p,k}$. Next, neighbouring servers communicate amongst each other the recieved updates:

$$\boldsymbol{\psi}_{p,i} = \frac{1}{L} \sum_{k \in \mathcal{L}_{p,i}} \boldsymbol{w}_{p,k,i}, \tag{4}$$

to finally get:

$$\boldsymbol{w}_{p,i} = \sum_{m \in \mathcal{N}_p} a_{mp} \boldsymbol{\psi}_{m,i}. \tag{5}$$

Next, to introduce privacy, updates sent during each communication round can be perturbed by some noise. Thus, at iteration $i$, let $\boldsymbol{g}_{mp,i}$ be the noise added by server $m$ to the update sent to server $p$, and $\boldsymbol{g}_{p,k,i}$ be the noise added by agent $k$ to the update sent to server $p$. Then, the algorithm can be described by a client update step (6), a server aggregation step (7), and a server combination step (8).

$$\boldsymbol{w}_{p,k,i} = \boldsymbol{w}_{p,i-1} - \mu \frac{1}{B_{p,k}} \sum_{b \in \mathcal{B}_{p,k,i}} \nabla_{w^\intercal} Q_{p,k}(\boldsymbol{w}_{p,i-1}; \boldsymbol{x}_{p,k,b}) \tag{6}$$

$$\boldsymbol{\psi}_{p,i} = \frac{1}{L} \sum_{k \in \mathcal{L}_{p,i}} \left( \boldsymbol{w}_{p,k,i} + \boldsymbol{g}_{p,k,i} \right), \tag{7}$$

$$\boldsymbol{w}_{p,i} = \sum_{m \in \mathcal{N}_p} a_{mp} \left( \boldsymbol{\psi}_{m,i} + \boldsymbol{g}_{mp,i} \right) \tag{8}$$

Furthermore, if we assume we are using SMC tools, like secret sharing, we can model the protocol by an invertible function $f(\cdot)$ that maps the local update to an encrypted version. Thus, in the server aggregation (7) and server combination (8) steps, we replace $\boldsymbol{w}_{p,k,i}$ and $\boldsymbol{\psi}_{m,i}$ with $f(\boldsymbol{w}_{p,k,i})$ and $f(\boldsymbol{\psi}_{m,i})$, respectively. For the remainder of the paper, we shall continue with the algorithm formulation in (6)-(8) instead of introducing $f(\cdot)$, for ease of notation.

## III. Performance Analysis

### A. Modeling Conditions

Certain reasonable assumptions on the nature of the graph and the cost functions are made to allow for a tractable convergence analysis.

**Assumption 1 (Adjacency matrix).** *The adjacency matrix $A$ describing the graph is symmetric and doubly-stochastic, i.e.:*

$$a_{pm} = a_{mp}, \quad \sum_{m=1}^{P} a_{mp} = 1. \tag{9}$$

*Furthermore, it is fully connected, satisfying:*

$$\lambda \triangleq \rho(A - \frac{1}{P} \mathbb{1}\mathbb{1}^\intercal) < 1. \tag{10}$$

□

**Assumption 2 (Convexity and smoothness).** *The empirical risks $P_{p,k}(\cdot)$ are $\nu-$strongly convex, and the loss functions $Q_{p,k}(\cdot;\cdot)$ are convex, namely:*

$$P_{p,k}(w_2) \geq P_{p,k}(w_1) + \nabla_{w^\intercal} P_{p,k}(w_1)(w_2 - w_1) + \frac{\nu}{2} \|w_2 - w_1\|^2, \tag{11}$$

$$Q_{p,k}(w_2;\cdot) \geq Q_{p,k}(w_1;\cdot) + \nabla_{w^\intercal} Q_{p,k}(w_1;\cdot)(w_2 - w_1). \tag{12}$$

*Furthermore, the loss functions have $\delta-$Lipschitz gradients:*

$$\|\nabla_{w^\intercal} Q_{p,k}(w_2;\cdot) - \nabla_{w^\intercal} Q_{p,k}(w_1;\cdot)\| \leq \delta \|w_2 - w_1\|. \tag{13}$$

□

Note that in our previous work [22], [23], we assumed that the local optimal models, which optimize $P_{p,k}(\cdot)$ at the agents, do not differ too much from the global optimal model at the server. We do not make such an assumption here since we are assuming each agent perfroms one epoch during the agent update step. More explicitly, the bound on the model dissagreement only appears in the incremental error term

which we do not have here. If we were to assume that the clients perform mulitple SGD steps in one model update step, then we would need such an assumption to make sure the incremental noise is bounded. However, this assumption is not restrictive, since if the local models differed too much, then collaboration would be nonsensical.

**Assumption 3 (Bounded gradients).** *The norm of the stochastic gradients is bounded along the trajectory of the algorithm:*

$$\|\nabla_{w^\intercal} Q_{p,k}(w;\cdot)\| \leq B \tag{14}$$

□

The bound on the gradient norm is required in the privacy analysis of the algorithm. In general, it is assumed that the gradients are uniformly bounded, and when that does not hold, as in the case of strongly convex cost functions, normalized gradients are used instead. However, we consider the less restrictive condition of bounding the gradients only on the models calculated by the algorithm.

### B. Error Recursion

We focus on the network centroid $\boldsymbol{w}_{c,i}$ defined by:

$$\boldsymbol{w}_{c,i} \triangleq \frac{1}{P}\sum_{p=1}^{P} \boldsymbol{w}_{p,i}. \tag{15}$$

By combining the three steps of the algorithm, we can get the following recursion for the network centroid:

$$\begin{aligned}\boldsymbol{w}_{c,i} =& \boldsymbol{w}_{c,i-1} - \mu\frac{1}{P}\sum_{p=1}^{P}\widehat{\nabla_{w^\intercal} P_p}(\boldsymbol{w}_{p,i-1}) \\ &+ \frac{1}{PL}\sum_{p=1}^{P}\sum_{k\in\mathcal{L}_{p,i}} \boldsymbol{g}_{p,k,i} + \frac{1}{P}\sum_{p=1}^{P}\sum_{m=1}^{P} a_{mp}\boldsymbol{g}_{mp,i},\end{aligned} \tag{16}$$

where we define the stochastic gradient at server $p$ as:

$$\widehat{\nabla_{w^\intercal} P_p}(\cdot) \triangleq \frac{1}{L}\sum_{k\in\mathcal{L}_{p,i}} \frac{1}{B_{p,k}} \sum_{b\in\mathcal{B}_{p,k,i}} \nabla_{w^\intercal} Q_{p,k}(\cdot;\boldsymbol{x}_{p,k,b}). \tag{17}$$

This expression approximates the true gradient $\nabla_{w^\intercal} P_p(\cdot)$. By defining $\widetilde{\boldsymbol{w}}_{c,i} = w^o - \boldsymbol{w}_{c,i}$ and the gradient noise:

$$\boldsymbol{s}_i = \frac{1}{P}\sum_{p=1}^{P}\left(\widehat{\nabla_{w^\intercal} P_p}(\boldsymbol{w}_{p,i-1}) - \nabla_{w^\intercal} P_p(\boldsymbol{w}_{p,i-1})\right), \tag{18}$$

we can write the following error recursion:

$$\widetilde{\boldsymbol{w}}_{c,i} = \widetilde{\boldsymbol{w}}_{c,i-1} + \mu\frac{1}{P}\sum_{p=1}^{P}\nabla_{w^\intercal} P_p(\boldsymbol{w}_{p,i-1}) + \mu\boldsymbol{s}_i - \boldsymbol{g}_{c,i}, \tag{19}$$

with $\boldsymbol{g}_{c,i}$ capturing the total added noise.

### C. Convergence Results

Before moving to the result on the network convergence, we introduce the following preliminary lemma. We show that all models at the servers $\{\boldsymbol{w}_{p,i}\}_{p=1}^{P}$ remain significantly close to the network centroid $\boldsymbol{w}_{c,i}$.

**Lemma 1 (Network disagreement).** *The average deviation from the centroid is bounded during each iteration $i$:*

$$\frac{1}{P}\sum_{p=1}^{P}\mathbb{E}\|\boldsymbol{w}_{c,i} - \boldsymbol{w}_{p,i}\|^2 \leq 2\mathbb{E}\|\widetilde{\boldsymbol{w}}_{c,i-1}\|^2 + O(\mu\sigma_s^2) + O(\sigma_{g_c}^2), \tag{20}$$

*where $O(\sigma_{g_c}^2)$ is a variance term that depends on the variance of the added noise $\boldsymbol{g}_{mp,i}$ and $\boldsymbol{g}_{p,k,i}$, and $\sigma_s^2$ is the variance of the gradient noise given by:*

$$\sigma_s^2 \triangleq \frac{2}{PK}\sum_{p=1}^{P}\sum_{k=1}^{K}\mathbb{E}\|\nabla_{w^\intercal} Q_{p,k}(w^o;\boldsymbol{x})\|^2. \tag{21}$$

*Proof.* Proof omitted due to space limitations □

We observe that the added noise contributes an added $O(\sigma_{g_c}^2)$ to the bound, which does not exist in the non-private algorithm. Furthermore, the bound is in terms of the centroid error $\widetilde{\boldsymbol{w}}_{c,i-1}$. As seen in the main theorem below, that term converges to a neighbourhood around zero.

**Theorem 1 (Convergence of MSE).** *Under Asumptions 1 and 2, the network centroid converges to the optimal point $w^o$ exponentially fast for a sufficientlt small step size $\mu$ :*

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_{c,i}\|^2 \leq \lambda^i\mathbb{E}\|\widetilde{\boldsymbol{w}}_{c,0}\|^2 + \frac{1-\lambda^i}{1-\lambda}O\big(\mu^2(\sigma_s^2+\sigma_{g_c}^2)+\sigma_{g_c}^2\big), \tag{22}$$

*where $\lambda = 1 - O(\mu) + O(\mu^2) \in (0,1)$. Then, repeating the algorithm infinitly many times, we get:*

$$\limsup_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{w}}_{c,i}\|^2 \leq O(\mu(\sigma_s^2+\sigma_{g_c}^2)) + O(\mu^{-1}\sigma_{g_c}^2). \tag{23}$$

*Proof.* Proof omitted due to space limitations. □

Thus, a close examination of the above theorem reveals that all privatized algorithms that can be modelled by added noise, add a noise variance term scaled by $O(\mu+\mu^{-1})$. The $O(\mu^{-1})$ term comes from the noise added at the client level to the updates sent to the server, while the $O(\mu)$ term comes from the network disagreement between the models at the server and the centroid model. The result does not come as a surprise, since it quantifies the trade-off between privacy and accuracy.

### D. Performance of the hybrid scheme

We now specialize the above results to the scheme adopted in this work. The protocol developed in [16] utilizes a secret sharing method to insure that the messages sent by the clients arrive to the server encoded. The method is equivalent to

applying a mask to the updates by each client, which cancels out at the server, i.e., at every server $p$ the following holds:

$$\sum_{k\in\mathcal{L}_{p,i}} \boldsymbol{g}_{p,k,i} = 0. \tag{24}$$

Furthermore, we apply graph homomorphic perturbations, introduced in [21]. Let each server $p$ sample independently from the Laplace distribution $\boldsymbol{g_{p,i}} \sim Lap(0, \sigma_g/\sqrt{2})$ with variance $\sigma_g^2$. Then, the noise sent among servers can be constructed as:

$$\boldsymbol{g}_{mp,i} = \begin{cases} \boldsymbol{g}_{m,i}, & \text{if } m \neq p, \\ -\frac{1-a_{mm}}{a_{mm}}\boldsymbol{g}_{m,i}, & \text{if } m = p. \end{cases} \tag{25}$$

Thus, the following result holds:

$$\frac{1}{P}\sum_{p=1}^{P}\sum_{m=1}^{P} a_{mp}\boldsymbol{g}_{mp,i} = 0. \tag{26}$$

Therefore, with this scheme, the centroid model recursion (16) has no noise component. This implies that the $O(\mu^{-1})$ term disappears from the bound of the MSE (23). Eventhough the effect of the noise added by the servers remains, it is scaled by $\mu$ in the MSE bound.

## IV. Privacy Analysis

We focus on the privacy of the hybrid scheme described in the previous section. We quantify privacy using differential privacy [24]. Thus, we first need to find the sensitivity of the graph FedAvg algorithm, since it is used to callibarate the perturbations. To do so, consider, without loss of generality, that client 1 connected to server 1 decided not to participate, and instead its data $\boldsymbol{x}_{1,1}$ was replaced by some other data $\boldsymbol{x}'$ with a different distribution. Then, the algorithm will follow a different trajectory $\boldsymbol{w}'_{p,k,i}$. The sensitivity of the function is thus given by:

$$\Delta(i) \triangleq \max_{(p,k)} \|\boldsymbol{w}_{p,k,i} - \boldsymbol{w}'_{p,k,i}\| \leq 2\mu B i \tag{27}$$

Next, we wish to show that the algorithm is differentially private, but before doing so we present the definition of $\epsilon(i)-$differential privacy.

**Definition 1** ($\epsilon(i)-$**Differential privacy**). *We say that the algorithm given in* (6)-(8) *is* $\epsilon(i)-$*differentially private for server* $p$ *at time* $i$ *if the following condition holds:*

$$\frac{\mathbb{P}\left(\left\{\{\boldsymbol{\psi}_{p,j} + \boldsymbol{g}_{mp,j}\}_{m\in\mathcal{N}_p\setminus\{p\}}\right\}_{j=0}^{i}\right)}{\mathbb{P}\left(\left\{\{\boldsymbol{\psi}'_{p,j} + \boldsymbol{g}_{mp,j}\}_{m\in\mathcal{N}_p\setminus\{p\}}\right\}_{j=0}^{i}\right)} \leq e^{\epsilon(i)} \tag{28}$$

□

The above definition states that the probability of any trajectory is comparable whether or not a client shares its data. Furthermore, it will be our goal to have a small $\epsilon(i)$ to get a higher privacy guarantee.

**Theorem 2** (**Privacy of GFL algorithm**). *If the algorithm* (6)-(8) *adopts the hybrid privacy scheme described in the previous section, then it is* $\epsilon(i)-$*differentially private, at time* $i$ *for a standard deviation of* $\sigma_g = \sqrt{2}\mu B(1+i)i/\epsilon(i)$ .

*Proof.* Proof omitted due to space limitations. □

Thus, if we wish to keep the privacy high as more iterations are performed, then the variance of the added noise ought to be increased. This clearly decreases the model utility, as seen in Theorem 1. Another way of interpreting the privacy theorem is as follow: If we keep $\sigma_g$ fixed, then $\epsilon(i) = \sqrt{2}\mu B(1+i)i/\sigma_g = O(i^2)$, which increases as more iterations are performed. Thus, the privacy decreases quadratically with time. This does not come as a surprise, since the longer the algorithm runs, the more information across servers and clients is difussed.

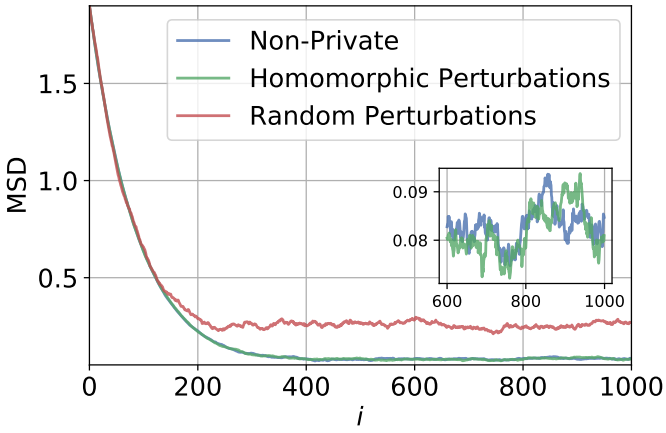


Fig. 2: Performance plots with $M = 2$, $\mu = 0.1$, $\rho = 0.01$, $\sigma_g = 0.2$

## V. Experimental Results

To illustrate the theoretical results numericaly, we simulate a GFL consisting of $P = 10$ servers, each with $K = 50$ clients, whose goal is to solve a logistic regression binary problem. We generate a set of data points $\{\gamma_{p,k}(n), h_{p,k,n}\}_{n=1}^{100}$ for each client, where $\gamma_{p,k}(n) = \pm 1$, and $h_{p,k,n} \in \mathbb{R}^M$ with $f(h_{p,k,n}|\gamma_{p,k}(n) = \gamma) = \mathcal{N}(\gamma; \sigma^2_{h,p,k})$. We compare our private scheme with a standard private algorithm that uses standard perturbations, and with the non-private algorithm. The results are found in Figure 2. We observe that our hybrid scheme performs well at approximating the non-private scheme. We also increase the noise variance, and we observe that while the IID private scheme does not converge, our scheme continues to perform as well as the non-private one.

## VI. Conclusion

In this work, we extended the federated learning architecture to GFL. We privatized the algorithm by using non-random perturbations. We quantified the privacy of our algorithm using differential privacy and provided a performance analysis. Both the theoretical and experimental results showed that dependent perturbations among servers reduce the negative effect of added noise to the model utility.

## REFERENCES


[1] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," *Proc. International Conference on Artificial Intelligence and Statistics*, vol. 54, pp. 1273–1282, 20–22 April 2017.

[2] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *IEEE International Conference on Communications (ICC)*, 7 – 11 Jun 2020, pp. 1–6.

[3] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2017, p. 603–618.

[4] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 19–23 May 2019, pp. 691–706.

[5] 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 *IEEE symposium on security and privacy (SP)*, San Jose, CA, USA, 19 – 23 May 2019, pp. 739–753.

[6] L. Zhu and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 8 – 14 Dec 2019, pp. 17–31.

[7] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[8] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9530–9539, 2020.

[9] A. Triastcyn and B. Faltings, "Federated learning with bayesian differential privacy," in *IEEE International Conference on Big Data*, Los Angeles, California, USA, 9 – 12 Dec 2019, pp. 2587–2596.

[10] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 61–66.

[11] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[12] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Distributed learning without distress: Privacy-preserving empirical risk minimization," in *Advances in Neural Information Processing Systems*, vol. 31, Montreal, Canad, 3 – 8 Dec 2018.

[13] C. Li, P. Zhou, L. Xiong, Q. Wang, and T. Wang, "Differentially private distributed online learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1440–1453, 2018.

[14] J. Zhu, C. Xu, J. Guan, and D. O. Wu, "Differentially private distributed online algorithms over time-varying directed networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 4–17, 2018.

[15] M. A. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers." in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 6 – 11 Dec 2010, pp. 1876–1884.

[16] 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 *Proc. of ACM SIGSAC Conference on Computer and Communications Security*, New York, USA, 2017, p. 1175–1191.

[17] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 345–364, 2017.

[18] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 22-26 May 2017, pp. 19–38.

[19] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, 19 – 22 May 2013, pp. 334–348.

[20] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 19 – 23 May 2019, pp. 724–738.

[21] S. Vlaski and A. H. Sayed, "Graph-homomorphic perturbations for private decentralized learning," in *Proc. ICASSP*, Toronto, Canada, June 2021.

[22] E. Rizk, S. Vlaski, and A. H. Sayed, "Federated learning under importance sampling," December 2020. [Online]. Available: arXiv:2012.07383.

[23] ——, "Optimal importance sampling for federated learning," in *Proc. ICASSP*, Toronto, Canada, June 2021.

[24] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.