LOCAL GRAPH-HOMOMORPHIC PROCESSING FOR PRIVATIZED DISTRIBUTED SYSTEMS

Elsa Rizk, Stefan Vlaski, Ali H. Sayed

ABSTRACT

We study the generation of dependent random numbers in a distributed fashion in order to enable privatized distributed learning by networked agents. We propose a method that we refer to as local graph-homomorphic processing; it relies on the construction of particular noises over the edges to ensure a certain level of differential privacy. We show that the added noise does not affect the performance of the learned model. This is a significant improvement to previous works on differential privacy for distributed algorithms, where the noise was added in a less structured manner without respecting the graph topology and has often led to performance deterioration. We illustrate the theoretical results by considering a linear regression problem over a network of agents.

***Index Terms*—** distributed systems, distributed learning, differential privacy, random number generator

1. INTRODUCTION AND RELATED MATERIAL

Distributed systems consist of networks of agents that collaborate to achieve a common goal. Some examples include distributed computing [1] when components of a software are shared over a network, or distributed machine learning [2, 3] where the goal is to fit a global model to the data dispersed over different computing locations. During collaboration in such systems, communication between neighbours is necessary. However, the shared information might be sensitive, such as in distributed systems handling health or financial data. Thus, there is a need to privatize the communication channels. One way to achieve secure communications is through cryptographic methods [4–7], while another is by adding random noise to make the communication differentially private [8–12].

In the standard implementations, agents add *independent* noise to their shared messages. This property degrades the performance of the learned model since the noises propagate over the graph through cooperation, as already shown in Theorem 1 of [13]. In order to endow agents with enhanced privacy with minimal effect on performance, it is necessary for the additional noise sources to be mindful of the graph topology [12]. However, this information is not available globally and, therefore, one needs to devise a scheme to generate graph-dependent random noise sources in a distributed manner and without assuming any global information about the graph structure. Motivated by this observation, we develop in this work a scheme that constructs privacy perturbations in a manner that their negative effect on performance is canceled out. One solution was suggested in [4] for the case of federated learning. Pairs of agent collaborate to add noise that cancels out at the server. However, the suggested method generates pseudo-random numbers, which is less secure than true random numbers [14] and without any guarantees of differential privacy.

The objective of this work is therefore to generate dependent random numbers in a distributed manner across a graph. The problem is challenging for at least two reasons. First, generating random numbers is usually difficult without enforcing beforehand some distribution for the random process. In practice, random number generators exploit a variety of entropy sources in a computer such as mouse movements, available memory, or temperature [15]. Second, it is not evident how agents should exploit independent entropy sources to generate *dependent* random numbers. Most available solutions [16–22] rely on a central orchestrator or consider a fully connected network. A truly distributed method does not appear to exist. While the work done in [23] relies on a similar solution presented in this work, they choose to add another noise that negatively effects the performance.

2. LOCAL GRAPH-HOMOMORPHIC PROCESS

2.1. Problem Setup

We consider a network of K agents connected by some graph topology (Fig. 1). We let $a_{mk} > 0$ denote the weight attributed to the message sent by neighbour m to agent k and let $A = [a_{mk}]$ denote the corresponding combination matrix. We assume A is symmetric and doubly-stochastic, i.e.,

$$\mathbb{1}^{\mathsf{T}} A = \mathbb{1}^{\mathsf{T}}, \quad A\mathbb{1} = \mathbb{1}. \tag{1}$$

We further denote the neighbourhood of agent k by $\mathcal{N}_k$; it consists of all agents connected to k by an edge.

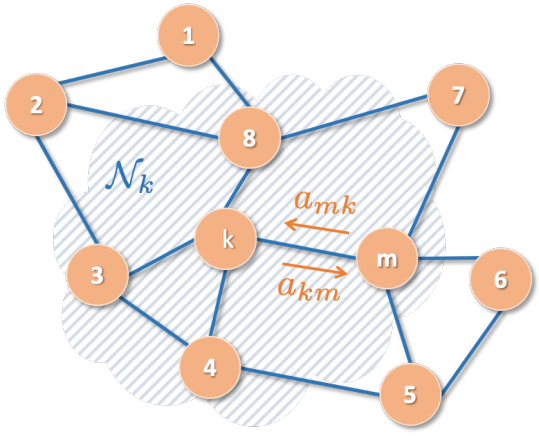


Fig. 1: Illustration of a network of agents.

We consider problems where agents aggregate the received messages from their neighbours. In other words, if we let $\boldsymbol{\psi}_{mk,i}$ denote

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

Department of Electrical and Electronic Engineering, Imperial College London (e-mail: s.vlaski@imperial.ac.uk).

the message sent by agent m to agent k at time i, then:

$$\boldsymbol{w}_{k,i} = \sum_{m \in \mathcal{N}_k} a_{mk} \boldsymbol{\psi}_{mk,i}, \tag{2}$$

which is the aggregate of all messages arriving at agent k. We wish to secure the communication between the agents. One method is to mask the messages with some random noise to guarantee some level of differential privacy. If we denote by $\boldsymbol{g}_{mk,i}$ the noise added to the message $\boldsymbol{\psi}_{mk,i}$, then the secure aggregation becomes:

$$\boldsymbol{w}_{k,i} = \sum_{m \in \mathcal{N}_k} a_{mk} \left(\boldsymbol{\psi}_{mk,i} + \boldsymbol{g}_{mk,i} \right). \tag{3}$$

Ideally, we would like that no or minimal information is lost by the added noise and that the aggregate message is equivalent to the non-noisy version. This is guaranteed if the noise sources added in (3) satisfy the following condition over every neighbourhood:

$$\sum_{m \in \mathcal{N}_k} a_{mk} \boldsymbol{g}_{mk,i} = 0. \tag{4}$$

Noises that satisfy (4) are said to arise from a *local graph-homomorphic process*. This is in contrast to the *global graph-homomorphic process* proposed in [12] where condition (4) is replaced by one that should hold over the entire graph, namely,

$$\sum_{k=1}^{K} \sum_{m \in \mathcal{N}_k} a_{mk} \boldsymbol{g}_{mk,i} = 0. \tag{5}$$

We would also like the noises $\boldsymbol{g}_{mk,i}$ added in (3) to ensure some level of differential privacy. This means that if the agent m chooses to share different messages $\boldsymbol{\psi}'_{mk,i}$, then an observer would be oblivious to this change. This is more formally defined as follows.

Definition 1 ($\epsilon(i)-$**Differential privacy**). *We say the communication is $\epsilon(i)-$differentially private for agent m at time i if the following condition on the probability density functions of the respective events holds for all agents:*

$$\frac{f\left(\left\{\left\{\boldsymbol{\psi}_{mk,j} + \boldsymbol{g}_{mk,j}\right\}_{k \in \mathcal{N}_m \setminus \{m\}}\right\}_{j=0}^{i}\right)}{f\left(\left\{\left\{\boldsymbol{\psi}'_{mk,j} + \boldsymbol{g}_{mk,j}\right\}_{k \in \mathcal{N}_m \setminus \{m\}}\right\}_{j=0}^{i}\right)} \leq e^{\epsilon(i)}. \tag{6}$$

□

2.2. Process Description

To motivate the local graph-homomorphic process, we examine the following example. Alice and Bob wish to communicate to Charlie the aggregate of their messages without Charlie knowing the individual messages. Alice and Bob decide to send a noisy version of their messages to Charlie. However, they wish when their noisy messages are aggregated by Charlie that he will still be able to retrieve the original sum. One way to do so is by ensuring that the noises generated by Alice and Bob cancel out when Charlie computes a weighted sum of the messages. For example, they could agree on some random number $\boldsymbol{x}$ that Alice would add to her message while Bob would subtract it from his message. Now assume that all communications between Alice and Bob need to go through Charlie, i.e., no direct communication channel exists between Alice and Bob. Then, in this case, both Alice and Bob will need to agree on the random number $\boldsymbol{x}$ without explicitly mentioning it. In other words, secure communication between them will need to be set up through Charlie. One way of doing so is through the Diffie-Helman key exchange protocol [24].

Let Alice and Bob have individual secret keys $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, respectively. Let p be a known prime number and b a base. Then, both Alice and Bob will broadcast their public keys $\boldsymbol{V}_1 = b^{\boldsymbol{v}_1} \mod p$ and $\boldsymbol{V}_2 = b^{\boldsymbol{v}_2} \mod p$. When they raise the public key of the other by their secret key and take the modulus p, they will now share a common secret key $\boldsymbol{v}_{12} = b^{\boldsymbol{v}_1 \boldsymbol{v}_2} \mod p$. This secret key can be used as the added noise; while Alice adds $\boldsymbol{v}_{12}$ to her message, Bob can subtract it. However, to ensure the communication is differentially private, one choice of distribution for the noise is the Laplace distribution $\text{Lap}(0, \sigma_g/\sqrt{2})$. A Laplace random variable can be generated from two uniform random variables by taking the log of the ratio of the two variables and then multiplying by the inverse of the scale parameter, namely, $\sqrt{2}/\sigma_g$. Thus, to generate a Laplace random variable, we require two secret keys $\{\boldsymbol{v}_{12}, \boldsymbol{v}'_{12}\}$ that are uniformly distributed. For $\boldsymbol{v}_{12}$ to be a uniform random variable, one of the local secret keys must be uniformly distributed over $[0, 1]$ while the other must be sampled from a gamma distribution $\Gamma(2, 1)$. Furthermore, the base must be set to $b = e^{-1}$ and then scaled by some constant a that is a multiple of the prime number p. Therefore, for instance, Alice should sample two uniformly distributed secret keys $\{\boldsymbol{v}_1, \boldsymbol{v}'_1\} \sim U([0, 1])$, and Bob must generate two secret keys $\{\boldsymbol{v}_2, \boldsymbol{v}'_2\}$ from a gamma distribution. The resulting two shared secret keys will be uniformly distributed on $[0, p]$. Then, setting:

$$\boldsymbol{x} = \frac{\sqrt{2}}{\sigma_g} \ln\left(\frac{\boldsymbol{v}_{12}}{\boldsymbol{v}'_{12}}\right), \tag{7}$$

results in a Laplace noise, which Alice can add to her message while Bob subtracts it from his.

Returning to the network setting, we describe the process by which the agents generate their local graph-homomorphic noises. Each agent randomly splits its neighbourhood into two groups, $\mathcal{N}_k = \mathcal{N}_+ \bigcup \mathcal{N}_-$, and communicates the split to its neighbourhood. One method of splitting the neighbourhood is by attributing to each neighbour a number, and then placing all the even-numbered agents in one set, and the odd-numbered agents in the other set. Then, every pair of agents from the two sub-neighbourhoods will generate together a shared noise, with the agent in $\mathcal{N}_+$ adding the noise to its message and the agent in $\mathcal{N}_-$ subtracting it. The communication betwen the agents of the sub-neighbourhoods occurs through the main agent k, since these agents might not be neighbours (e.g., agents 4 and m in Fig. 1). The messages are scaled by the weights attributed to the neighbours by a given agent. Thus, we force each neighbour to scale its noise by the inverse of the attributed weight. For agents $\ell \in \mathcal{N}_+$ and $m \in \mathcal{N}_-$, we denote the generated noise by $\boldsymbol{g}_{\{\ell m\}k,i}$ where we now add the subscript $\{\ell m\}$ to indicate that the noise was generated by the pair of agents. We follow the convention of writing the subscript of the agent from the positive set first, followed by that from the negative set. Then, every neighbour ℓ will send agent k its message masked by the sum of all the noise it generated with the agents from the adjacent sub-neighbourhood. A more detailed description of the process is found in Algorithm 1. An illustrative example is found in Fig. 2.

3. PRIVATIZED DISTRIBUTED LEARNING

We apply the above construction to the problem where a network of agents aims to solve an aggregate convex optimization problem of

Algorithm 1 (Local graph-homomorphic processing)

for each iteration $i = 1, 2, \cdots$ **do**

for each agent $k = 1, 2, \cdots, K$ **do**

Split $\mathcal{N}_k = \mathcal{N}_+ \bigcup \mathcal{N}_-$ and communicate the split to the neighbours.

for each pair of agents $\ell \in \mathcal{N}_+$ and $m \in \mathcal{N}_-$ **do**

Agent ℓ samples two secret keys $\{\boldsymbol{v}_\ell, \boldsymbol{v}'_\ell\} \sim U([0,1])$, and angent m samples two secret keys $\{\boldsymbol{v}_m, \boldsymbol{v}'_m\} \sim \Gamma(2,1)$.

Calculate and broadcast the public keys

$$\boldsymbol{V}_\ell = a e^{-v_\ell} \mod p$$
$$\boldsymbol{V}'_\ell = a e^{-v'_\ell} \mod p$$
$$\boldsymbol{V}_m = a e^{-v_m} \mod p$$
$$\boldsymbol{V}'_m = a e^{-v'_m} \mod p$$

Calculate the shared secret keys

$$\boldsymbol{v}_{\ell m} = a e^{-\boldsymbol{v}_\ell \boldsymbol{v}_m} \mod p$$
$$\boldsymbol{v}'_{\ell m} = a e^{-\boldsymbol{v}'_\ell \boldsymbol{v}'_m} \mod p$$

Set the noise

$$\boldsymbol{g}_{\{\ell m\}k,i} = \frac{\sqrt{2}}{\sigma_g} \ln\left(\frac{\boldsymbol{v}_{\ell m}}{\boldsymbol{v}'_{\ell m}}\right)$$

end for

for each agent $\ell \in \mathcal{N}_k$ **do**

if $\ell \in \mathcal{N}_+$ **then**

Send $\boldsymbol{\psi}_{\ell k,i} + \sum_{m \in \mathcal{N}_-} \boldsymbol{g}_{\{\ell m\}k,i}/a_{\ell k}$

else

Send $\boldsymbol{\psi}_{\ell k,i} - \sum_{m \in \mathcal{N}_+} \boldsymbol{g}_{\{m\ell\}k,i}/a_{\ell k}$

end if

end for

end for

end for

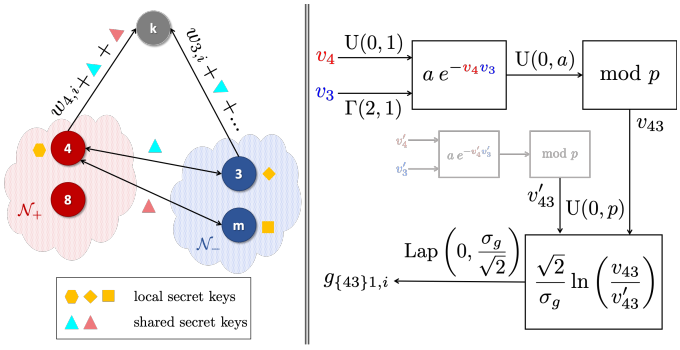


Fig. 2: Illustration of the local graph-homomorphic process with the Diffie-Helman key exchange protocol on the left and the transformation of the random variable on the right .

the form:

$$w^o \triangleq \operatorname*{argmin}_{w \in \mathbb{R}^M} \frac{1}{K} \sum_{k=1}^{K} \left\{ J_k(w) \triangleq \frac{1}{N_k} \sum_{n=1}^{N_k} Q_k(w; x_{k,n}) \right\}, \quad (8)$$

where the risk function $J_k(\cdot)$ is associated with agent k and is defined as an empirical average of the loss function $Q_k(\cdot;\cdot)$ evaluated over the local dataset $\{x_{k,n}\}_{n=1}^{N_k}$. We assume the loss functions are convex with Lipschitz continuous gradients and the risk functions are strongly convex.

Assumption 1 (Convexity and smoothness). *The empirical risks $J_k(\cdot)$ are $\nu-$strongly convex, and the loss functions $Q_k(\cdot;\cdot)$ are convex and twice differentiable, namely, for some $\nu > 0$:*

$$J_k(w_2) \geq J_k(w_1) + \nabla_{w^\mathsf{T}} J_k(w_1)(w_2 - w_1) + \frac{\nu}{2}\|w_2 - w_1\|^2, \quad (9)$$

$$Q_k(w_2;\cdot) \geq Q_k(w_1;\cdot) + \nabla_{w^\mathsf{T}} Q_k(w_1;\cdot)(w_2 - w_1). \quad (10)$$

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

$$\|\nabla_{w^\mathsf{T}} Q_k(w_2; x_{k,n}) - \nabla_{w^\mathsf{T}} Q_k(w_1; x_{k,n})\| \leq \delta \|w_2 - w_1\|. \quad (11)$$

□

We next make an assumption on the drift between the local optimal models $w_k^o = \operatorname{argmin} J_k(w)$ and the global optimal model w^o. For collaboration to make sense, the drift must be bounded. In case the difference is not bounded, then the agents should not collaborate to find one global model since that global model will not perform well locally.

Assumption 2 (Model drifts). *The distance of each local model w_k^o to the global model w^o is uniformly bounded, $\|w^o - w_k^o\| \leq \xi$.* □

To approximate the optimal model w^o, the agents can collaborate and run a distributed algorithm like consensus [25–27] or diffusion [28–30], while at the same time adding noise to their messages to ensure a certain level of privacy. For instance, the privatized adapt-then-combine (ATC) diffusion algorithm would take the following form:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu \nabla_{w^\mathsf{T}} Q_k(\boldsymbol{w}_{k,i-1}; \boldsymbol{x}_{k,b}), \quad (12)$$

$$\boldsymbol{w}_{k,i} = \sum_{m \in \mathcal{N}_k} a_{mk} \left(\boldsymbol{\psi}_{m,i} + \boldsymbol{g}_{mk,i}\right), \quad (13)$$

where we now drop the second subscript k from the message $\boldsymbol{\psi}_{m,i}$ since the same message is sent to all the neighbours of agent m, i.e., $\boldsymbol{\psi}_{mk,i} = \boldsymbol{\psi}_{m,i}$. Then, because $\boldsymbol{g}_{mk,i}$ is sampled from a Laplacian distribution, this construction ensures that the algorithm is $\epsilon(i)-$differentially private for some choice of variance σ_g^2 (see Theorem 2 in [13]). Recal that the local graph-homomorphic noises in (13) are generated from the Laplacian noises $\boldsymbol{g}_{mk,i}$:

$$\boldsymbol{g}_{mk,i} = \begin{cases} -\sum_{\ell \in \mathcal{N}_+} \boldsymbol{g}_{\{\ell m\}k,i}, & m \in \mathcal{N}_- \\ \sum_{\ell \in \mathcal{N}_-} \boldsymbol{g}_{\{m\ell\}k,i}. & m \in \mathcal{N}_+ \end{cases} \quad (14)$$

Since, by construction, the noises cancel out, the performance of the privatized ATC diffusion strategy (12)–(13) ends up being equivalent to the performance of the traditional non-privatized strategy without degradation. Thus, the algorithm will still converge to an $O(\mu)$ neighbourhood of the optimal model w^o. This is a significant improvement compared to earlier results where the algorithm was shown to converge to a neighbourhood on the order of $O(\mu^{-1})$ or $O(1)$ — see, e.g., [12, 13] and the discussions therein.

Theorem 1 (MSE convergence). *Under assumptions 1 and 2, the privatized diffusion strategy* (12)−(13) *with noise generated from the local graph-homomorphic process described earlier, converges exponentially fast for a small enough step-size to a neighbourhood of the optimal model:*

$$\limsup_{i\to\infty} \mathbb{E}\|\widetilde{\boldsymbol{w}}_{k,i}\|^2 \leq O(\mu), \tag{15}$$

where the error is defined as $\widetilde{\boldsymbol{w}}_{k,i} = w^o - \boldsymbol{w}_{k,i}$.

Proof. Since the noise cancels out locally during each iteration, the algorithm is equivalent to the non-privatized version. The proof then follows the arguments used to establish Theorem 9.1 in [30]. □

In the next theorem, we explain that the proposed algorithm is differentially private.

Theorem 2 (Privacy of distributed learning). *Under the local graph-homomorphic process, the privatized diffusion algorithm* (12)–(13) *is* $\epsilon(i)$*−differentially private with:*

$$\epsilon(i) \triangleq \frac{2\sqrt{2}}{\sigma_g}\left\{\left(\frac{1-(1-O(\mu))^{i+1}}{O(\mu)} - 1\right) a + b + O(\mu^{0.5})(i-1)\right\}, \tag{16}$$

where a *and* b *are some constants.*

Proof. We provide a sketch of the proof. We first show that the generated noise from the local graph-homomorphic process is Laplacian. Then, using a bound on the gradients at each step of the algorithm, we can bound the sensitivity of the algorithm. This can then be used to establish condition (6) in Definition 1. □

As time passes, $\epsilon(i)$ increases which means higher privacy loss. To mitigate this problem, the noise variance can be increased to guarantee a certain level of privacy. Since the variance of the perturbations does not affect the MSE bound, we do not hinder the model utility by increasing the variance, as opposed to the traditional differentially privatized algorithms (where the noises are not graph-homomorphic); in these cases, the MSE will worsen by an $O(\mu^{-1})\sigma_g^2$ factor.

4. EXPERIMENTAL RESULTS

We study a linear regression problem over a network of $K = 30$ agents with a regularized quadratic loss:

$$\min_{w\in\mathbb{R}^2} \frac{1}{30\times 100}\sum_{k=1}^{30}\sum_{n=1}^{100}\|\boldsymbol{d}_k(n) - \boldsymbol{u}_{k,n}^{\mathsf{T}} w\|^2 + 0.01\|w\|^2. \tag{17}$$

We generate for each agent 100 data samples $\{\boldsymbol{u}_{k,n}, \boldsymbol{d}_k(n)\}$. We sample two-dimensional feature vectors $\boldsymbol{u}_{k,n} \sim \mathcal{N}(0, R_u)$ and an independent noise $\boldsymbol{v}_p(n) \sim \mathcal{N}(0, \sigma_{v,k}^2)$ such that $\boldsymbol{d}_k(n) = \boldsymbol{u}_{k,n}^{\mathsf{T}} w^\star + \boldsymbol{v}_k(n)$ for some generative model $w^\star$. The optimal model is given by:

$$w^o = (\widehat{R}_u + 0.01I)^{-1}(\widehat{R}_u w^\star + \widehat{r}_{uv}), \tag{18}$$

where $\widehat{R}_u$ and $\widehat{r}_{uv}$ are the respective sample covariance matrix and cross-covariance.

We set the step-size $\mu = 0.4$, the noise variance $\sigma_g^2 = 0.01$, and the total number of iterations 1000. We repeat the algorithm 20 times and calculate the average MSD of the centroid model defined as:

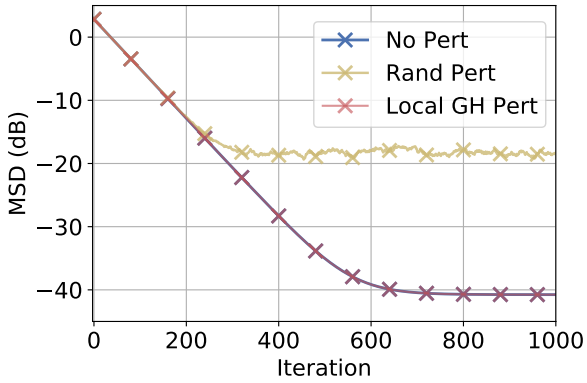
$$\boldsymbol{w}_{c,i} \triangleq \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_{k,i}, \tag{19}$$

and the individual models:

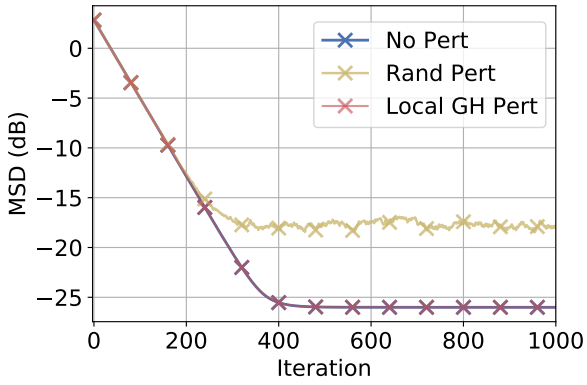
$$\text{MSD}_i \triangleq \|\boldsymbol{w}_{c,i} - w^o\|^2, \tag{20}$$

$$\text{MSD}_{\text{avg},i} \triangleq \frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{w}_{k,i} - w^o\|^2. \tag{21}$$

We plot the results of the non-privatized algorithm, the privatized algorithm with random perturbations, and the privatized algorithm with local graph-homomorphic perturbations. As expected from Theorem 1, the local graph-homomorphic perturbations do not affect the performance of the algorithm.



(a) Centroid MSD



(b) Average individual MSD

Fig. 3: MSD plots for the distributed learning algorithms.

5. CONCLUSION

We introduce a distributed random number generator and apply it to a distributed learning setting to ensure differential privacy without degradation in performance.

6. REFERENCES

[1] K. R. Apt, E.-R. Olderog, and K. Apt, "Distributed programs," in *Verification of Sequential and Concurrent Programs*. London: Springer, 2009, pp. 373–406.

[2] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[3] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1–33, 2020.

[4] 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. ACM SIGSAC Conference on Computer and Communications Security*, New York, USA, 2017, pp. 1175–1191.

[5] 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, 2017, pp. 19–38.

[6] D. Froelicher, J. R. Troncoso-Pastoriza, A. Pyrgelis, S. Sav, J. S. Sousa, J.-P. Bossuat, and J.-P. Hubaux, "Scalable privacy-preserving distributed learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 2, pp. 323–347, 2021.

[7] 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, 2013, pp. 334–348.

[8] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[9] 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*, Montreal, Canad, 2018, p. 6346–6357.

[10] 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.

[11] 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, 2010, pp. 1876–1884.

[12] S. Vlaski and A. H. Sayed, "Graph-homomorphic perturbations for private decentralized learning," in *Proc.* IEEE ICASSP, Toronto, Canada, June 2021, pp. 5240–5244.

[13] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications*, Lucca, Italy, 2021, pp. 1–5.

[14] S. N. Cohney, M. D. Green, and N. Heninger, "Practical state recovery attacks against legacy rng implementations," in Proc. *ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, 2018, p. 265–280.

[15] D. Johnston, *Random Number Generators – Principles and Practices*. De Gruyter Press, 2018.

[16] T. Nguyen-Van, T.-D. Le, T. Nguyen-Anh, M.-P. Nguyen-Ho, T. Nguyen-Van, M.-Q. Le-Tran, Q. N. Le, H. Pham, and K. Nguyen-An, "A system for scalable decentralized random number generation," in *IEEE International Enterprise Distributed Object Computing Workshop*, 2019, pp. 100–103.

[17] I. Cascudo and B. David, "Scrape: Scalable randomness attested by public entities," in *International Conference on Applied Cryptography and Network Security*, Kanazawa, Japan, 2017, pp. 537–556.

[18] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *IEEE Symposium on Security and Privacy*, San Jose, California, 2017, pp. 444–460.

[19] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv:1805.04548*, 2018.

[20] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "Hydrand: Practical continuous distributed randomness," *Cryptology ePrint Archive*, 2018.

[21] S. Popov, "On a decentralized trustless pseudo-random number generation algorithm," *Journal of Mathematical Cryptology*, vol. 11, no. 1, pp. 37–43, 2017.

[22] M. Blum, "Coin flipping by telephone a protocol for solving impossible problems," *SIGACT News*, vol. 15, no. 1, p. 23–27, jan 1983.

[23] H. Imtiaz, J. Mohammadi, R. Silva, B. Baker, S. M. Plis, A. D. Sarwate, and V. D. Calhoun, "A correlated noise-assisted decentralized differentially private estimation protocol, and its application to fMRI source separation," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6355–6370, 2021.

[24] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[25] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association.*, vol. 69, no. 345, pp. 118–121, 1974.

[26] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. IEEE Conf. Dec. Control (CDC)*, Cancun, Mexico, December 2008, pp. 4185–4190.

[27] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[28] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, Aug 2012.

[29] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, Dec 2012.

[30] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.