DIFFERENTIALLY PRIVATE DEEP LEARNING WITH TIGHT PRIVACY-COST COMPUTATION

Jun Zhao

Department of Electrical and Computer Engineering
Carnegie Mellon University
junzhao@alumni.cmu.edu

ABSTRACT

Deep learning has been used in various applications to achieve outstanding results. Yet, massive data collection required for deep learning presents obvious privacy issues. To this end, the popular privacy framework called differential privacy has been recently applied to deep learning to protect data from the adversary. In this paper, we derive the privacy cost of differentially private deep learning. Compared with the recent result of Abadi *et al*. [1] in ACM CCS 2016, our expression for the privacy cost is tighter (i.e., giving a lower cost), applicable to a broader set of parameters, and more efficient to compute.

***Index Terms* —** Differential privacy, deep learning, privacy cost, Gaussian mechanism.

1. INTRODUCTION

Deep learning has recently become hugely popular [1–4]. Its success is due to a combination of recent algorithmic breakthroughs, increasingly powerful computers, and access to significant amounts of data. However, massive data collection required for deep learning presents obvious privacy issues.

Differential privacy by Dwork *et al.* [5] is a robust privacy standard that has been used in a range of data analysis tasks, since it provides a rigorous foundation for defining and preserving privacy. Differential privacy has received considerable attention in the literature [1, 6–8]. Apple incorporated differential privacy into its mobile operating system iOS 10 [9]. Google implemented a differentially private tool called RAPPOR in the Chrome browser to collect information about clients [10]. Intuitively, a randomized mechanism achieving (ϵ, δ)-differential privacy means that except with a (typically small) probability δ, altering a record in a database cannot change the probability that an output is seen by more than a multiplicative factor e^{ϵ}. Formally, for x and x' iterating through all pairs of neighboring databases (i.e., databases that differ by one record), and for $\mathcal{Y}$ iterating through all subsets of the output range of some mechanism Y, the mechanism Y achieves (ϵ, δ)-differential privacy if $\mathbb{P}[Y(x) \in \mathcal{Y}] \leq e^{\epsilon}\mathbb{P}[Y(x') \in \mathcal{Y}] + \delta$, where $\mathbb{P}[\cdot]$ denotes the probability, and the probability space is over the coin flips of the mechanism Y. If $\delta = 0$, (ϵ, δ)-differential privacy becomes ϵ-differential privacy. In several studies [11–13], (ϵ, δ)-differential privacy and ϵ-differential privacy are also referred to as approximate differential privacy and pure differential privacy, respectively.

Abadi *et al.* [1] recently applied (ϵ, δ)-differential privacy to deep learning techniques based on stochastic gradient descent, where the Gaussian noise is added to the gradient for achieving differential privacy.

Improvements of this paper over Abadi *et al.* [1]. We derive the privacy cost of differentially private deep learning. Compared with the result of Abadi *et al*. [1], our expression for the privacy cost is tighter (i.e., giving a lower cost), applicable to a broader set of parameters, and more efficient to compute. Specifically, the improvements are as follows:

1. Our method of computing the privacy cost applies to all parameters, whereas [1]'s approach does not apply to all noise amounts. In particular, with σ denoting the noise amount and q being the sampling probability ([1] constructs a mini-batch used in stochastic gradient descent by sampling each example with probability q), [1]'s approach does not work for $\sigma < 1$ (i.e., small σ) or $\sigma \geq 1/q$ (i.e., large σ).

2. We give a tighter privacy cost than [1]. For example, for a set of parameters discussed after Theorem 1 of [1], Abadi *et al*. [1] give a privacy cost of 1.26, while our expression induces 1.136 (i.e., 10% improvement). The improvement will be further higher for other set of parameters since (i) our privacy cost is tight for any set of parameters, and (ii) the closer σ is to either 1 or $1/q$, the loose [1]'s privacy cost is (note that [1]'s result is applicable to only $1 \leq \sigma < 1/q$.

3. Our expression is exact and does not involve asymptotics. In contrast, much of [1]'s analysis is either asymptotic (e.g., Lemma 3 on Page 12 in the full version of [1] at https://arxiv.org/abs/1607.00133) or involves unspecified constants (e.g., Theorem 1 on Page 4 of [1]). Results without exact expressions are not quite useful to evaluate how large the privacy cost is.

4. Our expression of privacy cost is more efficient to compute. Note that Theorem 1 of [1] is not used to evaluate the privacy cost in [1]'s experiments due to the following reason: even if the constants in Theorem 1 of [1] can be given, the privacy cost computed in this way will be quite loose. Instead, [1] proposes a moment accountant method to compute the privacy cost without giving an exact expression of the privacy cost. However, the computation is inefficient and complex. Specifically, [1]'s computation of the privacy cost lets a parameter λ iterate through a set of values. In the experiments of [1], λ is set as an integer from 1 to 32 so that similar computations have to be invoked for 32 times. In contrast, our approach calculates the privacy cost efficiently without invoking similar computations multiple times.

5. Finally, we also consider the general setting of heterogeneous sampling probabilities, gradient clipping bounds and noise amounts, whereas [1] studies only the homogeneous case.

Roadmap. The rest of the paper is organized as follows. Section 2 introduces some preliminaries. We present privacy-cost analysis of differentially private deep learning in Sections 3 and 4, where Section 3 considers the case of homogeneous sampling rates, gradient clipping bounds and noise amounts, and Section 4 considers the case of heterogeneous sampling rates, gradient clipping bounds and noise amounts. Section 6 surveys related work, and Section 7 concludes the paper.

2. PRELIMINARIES

2.1. Deep Learning

Deep learning extracts complex features from high-dimensional data and uses them to build a model that relates inputs to outputs (e.g., classes). Deep learning architectures are often constructed as multi-layer networks where more abstract features are computed as nonlinear functions of lower-level features. In a typical multi-layer network, each neuron receives the output of the neurons in the previous layer plus a bias signal from a special neuron that emits one. It then computes a weighted average of its inputs, referred to as the total input. The output of the neuron is computed by applying a nonlinear activation function to the total input value. The output vector of neurons in layer k is $\boldsymbol{a}_k = f(W_k \boldsymbol{a}_{k-1})$, where f is an activation function and W_k is the weight matrix that determines the contribution of each input signal. If the neural network is applied to classifying input data into a finite number of classes, the activation function in the last layer is usually a softmax function $f(z_j) = e^{z_j}/(\sum_k e^{z_k}),\ \forall j$. In this case, the output of each neuron j in the last layer is the score or probability that the input belongs to class j. The main challenge in deep learning is to automatically learn from training data the values of the weight parameters that maximize the learning accuracy.

Learning network parameters using gradient descent. Learning the parameters of a neural network is a nonlinear optimization problem. The algorithms to solve this problem are often variants of gradient descent. After beginning at a random set of parameters, gradient descent at each step computes the gradient of the nonlinear function being optimized and updates the parameters in order to decrease the gradient. This process continues iteratively until the algorithm converges to a local optimum.

In a neural network, the gradient of each weight parameter is computed via feed-forward and back-propagation procedures. Feed-forward sequentially computes the output of the network and calculates the error defined as the difference between this output and the true value of the function. Back-propagation propagates this error back through the network to compute the contribution of each neuron to the total error. The gradients of individual parameters are computed according to the neurons' activation values and their contribution to the error.

Stochastic gradient descent. A naive approach is to average the gradients of the parameters over all N data samples. However, this algorithm is inefficient, especially if the dataset is large. A better alternative known as stochastic gradient descent (SGD) computes the gradient over a small subset (called minibatch) of the whole dataset.

Let $\boldsymbol{\theta}$ be the vector of all parameters in a neural network. Let $\mathcal{L}$ be the loss function which denotes the difference between the true value of the objective function and the computed output of the network. $\mathcal{L}$ is often based on ℓ_2 norm or cross entropy. The back-propagation algorithm calculates the partial derivative of $\mathcal{L}$ with respect to each parameter in $\boldsymbol{\theta}$ and updates the parameter in order to reduce its gradient. In [1], at step t, a minibatch L_t is constructed by sampling each example with a certain probability q. Then the update rule of SGD (without privacy) for a parameter θ_j is $\theta_j \leftarrow \theta_j - \eta \cdot \frac{1}{|L_t|} \sum_{x_i \in L_t} \frac{\partial \mathcal{L}}{\partial \theta_j}$, where η denotes the learning rate. For simplicity, [1] replaces the above $\frac{1}{|L_t|}$ by its expectation qN ([1] uses the notation L to represent qN). When differential privacy is incorporated, the gradient $\frac{\partial \mathcal{L}}{\partial \theta_j}$ is added with certain amount of Gaussian noise before being used in the above SGD update.

2.2. Differential Privacy

Among various mechanisms to achieve (ϵ, δ)-differential privacy, the Gaussian mechanism for real-valued queries by Dwork and Roth [14] has received much attention, where a certain amount of Gaussian noise is added independently to each dimension of the query result.

Recall that differential privacy intuitively means that the adversary cannot determine whether the randomized output comes from a database x or its neighboring database x' that

Algorithm 1 `HetergeneousDPDL` with heterogeneous sampling probabilities, gradient clipping bounds and noise amounts

Input: Data $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\boldsymbol{\theta}, x_i)$, sampling probabilities q_t, gradient clipping bounds C_t, noise amounts σ_t, learning rates η_t for $t = 1, 2, \ldots, T$

1: Initialize $\boldsymbol{\theta}_0$ randomly
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Take a random sample L_t with sampling probability q_t
4: **for** $x_i \in L_t$ **do**
5: Compute gradient: $\boldsymbol{g}_t(x_i) \leftarrow \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t, x_i)$
6: Clip gradient: $\overline{\boldsymbol{g}}_t(x_i) \leftarrow \boldsymbol{g}_t(x_i) \big/ \max\left(1, \frac{\|\boldsymbol{g}_t(x_i)\|_2}{C_t}\right)$
7: **end for**
8: Take average: $\overline{\boldsymbol{g}}_t \leftarrow \frac{1}{q_t N} \sum_{x_i \in L_t} \overline{\boldsymbol{g}}_t(x_i)$
9: Add noise: $\widetilde{\boldsymbol{g}}_t \leftarrow \overline{\boldsymbol{g}}_t + \mathcal{N}(0, \sigma_t^2 \times \boldsymbol{I})$
//Comment: $\boldsymbol{I}$ is an all-one matrix.
10: Descent: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta_t \widetilde{\boldsymbol{g}}_t$
11: **end for**

Output: $\boldsymbol{\theta}_T$ and compute the overall privacy cost (ϵ, δ)

Algorithm 2 `HomogeneousDPDL` with homogeneous sampling probabilities, gradient clipping bounds and noise amounts

Input: Data $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\boldsymbol{\theta}, x_i)$, sampling probability q, gradient clipping bound C, noise amount σ, learning rates η_t for $t = 1, 2, \ldots, T$

1: Initialize $\boldsymbol{\theta}_0$ randomly
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Take a random sample L_t with sampling probability q
4: **for** $x_i \in L_t$ **do**
5: Compute gradient: $\boldsymbol{g}_t(x_i) \leftarrow \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t, x_i)$
6: Clip gradient: $\overline{\boldsymbol{g}}_t(x_i) \leftarrow \boldsymbol{g}_t(x_i) \big/ \max\left(1, \frac{\|\boldsymbol{g}_t(x_i)\|_2}{C}\right)$
7: **end for**
8: Take average: $\overline{\boldsymbol{g}}_t \leftarrow \frac{1}{qN} \sum_{x_i \in L_t} \overline{\boldsymbol{g}}_t(x_i)$
9: Add noise: $\widetilde{\boldsymbol{g}}_t \leftarrow \overline{\boldsymbol{g}}_t + \mathcal{N}(0, \sigma^2 \times \boldsymbol{I})$
//Comment: $\boldsymbol{I}$ is an all-one matrix.
10: Descent: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta_t \widetilde{\boldsymbol{g}}_t$
11: **end for**

Output: $\boldsymbol{\theta}_T$ and compute the overall privacy cost (ϵ, δ)

Figure 1: Differentially private deep learning (DPDL) algorithms `HeterogeneousDPDL` and `HomogeneousDPDL`, where the former considers heterogeneous sampling probabilities, gradient clipping bounds and noise amounts, and the latter considers the homogeneous setting.

differs from x by one record. In unbounded differential privacy, the size of two neighboring databases differ by one; i.e., one record is in one database but not in the other database. In bounded differential privacy, two neighboring databases have the same size N, and have different records at only one of the N positions. The same as Abadi *et al.* [1], we consider unbounded differential privacy in this paper. Yet, it is straightforward to extend our results to bounded differential privacy.

2.3. Differentially Private Deep Learning (DPDL)

Abadi *et al.* [1] recently applied (ϵ, δ)-differential privacy to deep learning techniques based on stochastic gradient descent, where the Gaussian mechanism is used for adding noise to the gradient. Based on [1], we consider the differentially private deep learning (DPDL) algorithm with heterogeneous sampling probabilities, gradient clipping bounds and noise amounts, and call the algorithm `HeterogeneousDPDL`. We refer to the homogeneous case studied by [1] as `HomogeneousDPDL`. Figure 1 presents the details of `HeterogeneousDPDL` and `HomogeneousDPDL`.

3. PRIVACY-COST ANALYSIS OF DIFFERENTIALLY PRIVATE DEEP LEARNING UNDER HOMOGENEOUS SAMPLING RATES, GRADIENT CLIPPING BOUNDS AND NOISE AMOUNTS

Theorem 1 below analyzes the privacy cost of differentially private deep learning under homogeneous sampling rates, gradient clipping bounds and noise amounts.

Theorem 1. *For a query Q on a database x, suppose that adding a Gaussian noise with standard deviation σ_Q to the true query result $Q(x)$ achieves (ϵ_Q, δ_Q)-differential privacy, where ϵ is given by $f(\Delta_Q/\sigma_Q, \delta_Q)$ for Δ_Q being the ℓ_2-sensitivity of query Q (such function f will be given in Theorem 2 on Page 4). Then with Δ denoting the ℓ_2-sensitivity of $\overline{\boldsymbol{g}}_t := \frac{1}{qN} \sum_{i \in L_t} \overline{\boldsymbol{g}}_t(x_i)$ in Line 8 of the `HomogeneousDPDL` algorithm in Figure 1 on Page 3 (i.e., $\Delta := \frac{C}{qN}$), the `HomogeneousDPDL` algorithm achieves (ϵ, δ)-differential privacy for*

$$\epsilon = f\left(\Delta \Big/ \frac{\sigma}{q\sqrt{T}}, \delta\right). \tag{1}$$

Also, if we define function $\Delta_Q/\sigma_Q = g(\epsilon_Q, \delta_Q)$ such that $\epsilon_Q = f(\Delta_Q/\sigma_Q, \delta_Q) \Longleftrightarrow \Delta_Q/\sigma_Q = g(\epsilon_Q, \delta_Q)$. Then (1) can also be written as

$$\Delta \Big/ \frac{\sigma}{q\sqrt{T}} = g(\epsilon, \delta). \tag{2}$$

We prove Theorem 1 in Appendix B of the full version [15].

From (1), the privacy cost of the `HomogeneousDPDL` algorithm is the same as that of adding a Gaussian noise with standard deviation $\frac{\sigma}{q\sqrt{T}}$ to the true result of a query with ℓ_2-sensitivity Δ (or adding a Gaussian noise with standard deviation 1 to the true result of a query with ℓ_2-sensitivity $\Delta \big/ \frac{\sigma}{q\sqrt{T}}$). In addition, it follows from (1) that if we fix the privacy cost of the `HomogeneousDPDL` algorithm and the ℓ_2-sensitivity Δ, the resulting σ scales with q and $\sqrt{T}$.

We derive the above function f in Theorem 3 on Page 3. Substituting the expression of function f into (1), the privacy cost ϵ of the `HomogeneousDPDL` algorithm is

$$\epsilon = \frac{q^2 \Delta^2 T}{2\sigma^2} + \frac{q\Delta\sqrt{2T} \times \text{erfc}^{-1}(\delta)}{\sigma}, \tag{3}$$

where $\text{erfc}^{-1}()$ is the inverse of the complementary error function; i.e., y being $\text{erfc}^{-1}(x)$ satisfies $\frac{2}{\sqrt{\pi}}\int_y^\infty e^{-t^2}\,dt = x$.

Note that $\epsilon_Q = f(\Delta_Q/\sigma_Q,\, \delta_Q) \Longleftrightarrow \Delta_Q/\sigma_Q = g(\epsilon_Q, \delta_Q)$. In Theorem 3 on Page 3, after giving the function f, we also present the corresponding function g. Applying the expression of function g into (2), we have

$$\sigma = \epsilon^{-1} q \Delta \sqrt{T} \times \left(\sqrt{\left[\text{erfc}^{-1}(\delta)\right]^2 + \epsilon} \;+\; \text{erfc}^{-1}(\delta)\right) \Big/ \sqrt{2}.$$

4. PRIVACY-COST ANALYSIS OF DIFFERENTIALLY PRIVATE DEEP LEARNING UNDER HETEROGENEOUS SAMPLING RATES, GRADIENT CLIPPING BOUNDS AND NOISE AMOUNTS

Theorem 2 below analyzes the privacy cost of differentially private deep learning under heterogeneous sampling rates, gradient clipping bounds and noise amounts.

Theorem 2. *For a query Q on a database x, suppose that adding a Gaussian noise with standard deviation σ_Q to the true query result $Q(x)$ achieves $(\epsilon_Q,\, \delta_Q)$-differential privacy, where ϵ is given by $f(\Delta_Q/\sigma_Q,\, \delta_Q)$ for Δ_Q being the ℓ_2-sensitivity of query Q (such function f will be given in Theorem 2 on Page 4). Then with Δ_t denoting the ℓ_2-sensitivity of $\overline{\boldsymbol{g}}_t := \frac{1}{q_t N}\sum_{i\in L_t} \overline{\boldsymbol{g}}_t(x_i)$ in Line 8 of the* `HeterogeneousDPDL` *algorithm in Figure 1 on Page 3 (i.e., $\Delta_t := \frac{C_t}{q_t N}$), the* `HeterogeneousDPDL` *algorithm achieves $(\epsilon,\, \delta)$-differential privacy for*

$$\epsilon = f\left(\sqrt{\sum_{t=1}^{T} \frac{{q_t}^2 {\Delta_t}^2}{{\sigma_t}^2}},\; \delta\right). \tag{4}$$

Also, if we define function $\Delta_Q/\sigma_Q = g(\epsilon_Q, \delta_Q)$ such that $\epsilon_Q = f(\Delta_Q/\sigma_Q,\, \delta_Q) \Longleftrightarrow \Delta_Q/\sigma_Q = g(\epsilon_Q, \delta_Q)$. Then (4) can also be written as

$$\sqrt{\sum_{t=1}^{T} \frac{{q_t}^2 {\Delta_t}^2}{{\sigma_t}^2}} = g(\epsilon, \delta). \tag{5}$$

We prove Theorem 2 in Appendix C of the full version [15].

From (4), the privacy cost of the `HeterogeneousDPDL` algorithm is the same as that of adding a Gaussian noise with standard deviation 1 to the true result of a query Q with ℓ_2-sensitivity $\sqrt{\sum_{t=1}^{T} \frac{{q_t}^2 {\Delta_t}^2}{{\sigma_t}^2}}$.

We derive the above function f in Theorem 3 on Page 3. Substituting the expression of function f into (4), the privacy cost ϵ of the `HeterogeneousDPDL` algorithm is

$$\epsilon = \frac{1}{2}\sum_{t=1}^{T} \frac{{q_t}^2 {\Delta_t}^2}{{\sigma_t}^2} + \text{erfc}^{-1}(\delta) \times \sqrt{2\sum_{t=1}^{T} \frac{{q_t}^2 {\Delta_t}^2}{{\sigma_t}^2}}, \tag{6}$$

where $\text{erfc}^{-1}()$ is the inverse of the complementary error function.

5. REFINING THE GAUSSIAN MECHANISM OF DWORK AND ROTH [14] FOR DIFFERENTIAL PRIVACY

The Gaussian mechanism of Dwork and Roth [14] works for only $0 < \epsilon \leq 1$ and $0 < \delta < 0.41$ as explained in Lemma 1 of our recent work [16]. Theorem 3 below presents a Gaussian mechanism, which works for any $\epsilon > 0$ and any $0 < \delta < 1$.

Theorem 3. *For a query Q on a database x, suppose that adding a Gaussian noise with standard deviation σ to the true query result $Q(x)$ achieves $(\epsilon,\, \delta)$-differential privacy, where Δ is the ℓ_2-sensitivity of query Q. Then ϵ can be given by $f(\Delta/\sigma,\, \delta)$ for some function f, and Δ/σ can be given by $g(\epsilon, \delta)$ for some function g. Then f and g can be*

$$f(\Delta/\sigma,\, \delta) = \frac{\Delta^2}{2\sigma^2} + \sqrt{2} \times \mathit{erfc}^{-1}(\delta) \times \frac{\Delta}{\sigma}, \text{ and} \tag{7}$$

$$g(\epsilon, \delta) = \epsilon^{-1} \times \left(\sqrt{\left[\mathit{erfc}^{-1}(\delta)\right]^2 + \epsilon} \;+\; \mathit{erfc}^{-1}(\delta)\right) \Big/ \sqrt{2}. \tag{8}$$

We establish Theorem 3 as follows. From Mechanism 2 of our recent work [16], we can set $g(\epsilon, \delta)$ according to (8), which further induces $f(\Delta/\sigma,\, \delta)$ in (7). In the appendices of the full version [15], we use Theorem 3 to show Theorem 2, which is further used to prove Theorem 1.

6. RELATED WORK

Shokri and Shmatikov [2] first applies differential privacy to deep learning, but the privacy cost is too high. Abadi *et al.* [1] improves [2] by reducing the privacy cost in differentially private deep learning. They propose the moment accountant method to compute the privacy cost, while we present a better approach in this paper. Compared with [1], our approach presents a tighter result, applies to more general parameters, and is more efficient. As in [2], Zhang *et al.* [3] and Chase *et al.* [4] present systems for privacy-preserving multiparty deep learning. Hitaj *et al.* [17] present information leakage attacks in multiparty deep learning. Differential privacy has also been incorporated into machine learning algorithms other than deep learning. These studies include differentially private expectation maximization [18], differentially private K-nearest neighbors classification [19], and differentially private support vector machines [20].

7. CONCLUSION

In this paper, we derive the privacy cost of differentially private deep learning. Compared with the recent result of Abadi *et al*. [1], our expression for the privacy cost is tighter (i.e., giving a lower cost), applicable to a broader set of parameters, and more efficient to compute.

8. REFERENCES

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM Conference on Computer and Communications Security (CCS)*, 2016, pp. 308–318.

[2] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM Conference on Computer and Communications Security (CCS)*, 2015, pp. 1310–1321.

[3] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 1442–1452.

[4] M. Chase, R. Gilad-Bachrach, K. Laine, K. Lauter, and P. Rindal, "Private collaborative neural network learning," *Cryptology ePrint Archive, Report 2017/762*, 2017.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference (TCC)*, 2006, pp. 265–284.

[6] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1673–1693, 2012.

[7] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 2879–2887.

[8] I. Mironov, "Rényi differential privacy," in *IEEE Computer Security Foundations Symposium (CSF)*, 2017, pp. 263–275.

[9] Apple Incorporated, "iPhone user guide for iOS 10," 2016.

[10] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *ACM Conference on Computer and Communications Security (CCS)*, 2014, pp. 1054–1067.

[11] C. Dwork and G. N. Rothblum, "Concentrated differential privacy," *arXiv preprint arXiv:1603.01887*, 2016.

[12] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference (TCC)*, 2016, pp. 635–658.

[13] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "Generalization in adaptive data analysis and holdout reuse," in *Conference on Neural Information Processing Systems (NIPS)*, 2015, pp. 2341–2349.

[14] 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.

[15] J. Zhao, "Differentially private deep learning with tight privacy-cost computation," 2017. Full version of this paper. Available online at http://sites.google.com/site/workofzhao/ICASSP.pdf

[16] J. Zhao, "On the correct use of the Gaussian mechanism for differential privacy," 2017. Available online at http://sites.google.com/site/workofzhao/Gaussian.pdf

[17] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[18] M. Park, J. Foulds, K. Choudhary, and M. Welling, "DP-EM: Differentially private expectation maximization," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 896–904.

[19] M. E. Gursoy, A. Inan, M. E. Nergiz, and Y. Saygin, "Differentially private nearest neighbor classification," *Data Mining and Knowledge Discovery*, vol. 31, no. 5, pp. 1544–1575, 2017.

[20] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for SVM learning," *Journal of Privacy and Confidentiality*, vol. 4, no. 1, pp. 65–100, 2012.