# Privacy-preserving Decentralized Learning of Knowledge Graph Embeddings

Anh-Tu Hoang[1,*,†], Ahmed Lekssays[1,*,†], Barbara Carminati[1] and Elena Ferrari[1]

[1] *Università degli Studi dell'Insubria, Varese, Italy*

Abstract

Knowledge Graphs (KGs) enhance the performance of machine learning applications, such as recommendation systems and drug discovery. This is achieved through vector representations of KGs semantics, called Knowledge Graph Embeddings (KGEs). However, obtaining adequate data to train high-quality KGEs can be challenging for individual service providers. FedE and FedR address this challenge by enabling federated learning of KGEs without sharing local KGs, but they are limited by their reliance on trusted servers and lack of protection against inference attacks. Recently, FKGE has been proposed to enable collaboration between providers in the training of KGEs, exploiting differential privacy. Nevertheless, updating KGEs from all providers is time-consuming, and it does not protect against poisoning and backdoor attacks. Following this research direction, this paper focuses on the security and privacy requirements for decentralized learning of KGEs, presents a reference architecture to support these requirements, and discusses its security and privacy limitations.

**Keywords**

distributed learning, security, differential privacy, knowledge graph embeddings

## 1. Introduction

Knowledge graphs (KGs) are attracting giant tech companies thanks to their ability to represent knowledge about both entities' attributes and their relationships. The combination of entities' attributes and relationships in KGs improves the quality of various AI applications such as recommender systems (e.g., Amazon Product KGs[1]) and drug discovery (e.g., AstraZeneca's Drug Discovery KGs) [1]. These applications rely on KGs' embeddings (KGEs), which are vector representations of entities in KGs, defined such that the semantics between the entities are preserved. However, learning embeddings requires combining a large number of KGs, and sharing the KGs directly can violate the privacy of the entities.

Recently, FedE [2] has been designed to enable multiple data providers to jointly train the embeddings under federated learning (FL) settings. FedE utilizes a trusted server to collect the entities of the providers' KGs, aggregate the providers' embeddings, and distribute the aggregated embeddings to all providers. However, even though the local KGs are not shared, Zhang et al. [3] showed that relying on the trusted servers is impractical and introduced a new attack, i.e, a KG reconstruction attack, able to infer the local KGs by relying on the collusion between the server and one peer. To mitigate this attack, the authors in [3] proposed FedR. Instead of sharing entities' embeddings, FedR shares the relations ones. However, FedR failed to completely mitigate the attack. If a server is untrusted, it can perform the attack even without collusion with any peers. It can perform inference attacks on shared relations' embeddings to infer entities' existence. Moreover, it can simply refuse to aggregate model updates which makes the whole framework limited.

On the other side, Differential Privacy (DP) was proposed by Dwork et al. [4] to ensure that statistics extracted from relational data do not reveal the existence of users, even if the attacker exploits any background knowledge. To that end, DP adds noises to the data so that the statistics extracted from the data remain the same whether the user is present or not. DP has been used to train machine learning and deep learning models in such a way that the presence of users whose data are used to train the models is concealed [5, 6, 7]. In particular, [5] proposed DP-SGD, which adds noises to the updated weights of the models on data (e.g., images, text) to ensure that final weights satisfy DP. In this way, attackers cannot exploit the trained models to infer the users' existence. However, DP-SGD decreases the quality of the models when they are trained with a high number of epochs. To train a private model on the data with improved quality, PATE [6] and PATE-GAN [7] have been proposed.

In the context of KGs, recently, FKGE [8] adopted PATE-GAN [7] to ensure user privacy while allowing

*EDBT/ICDT 2023 Workshops*

*Corresponding authors.

† These authors contributed equally.

anhtu.hoang@uninsubria.it (A. Hoang); alekssays@uninsubria.it (A. Lekssays); barbara.carminati@uninsubria.it (B. Carminati); elena.ferrari@uninsubria.it (E. Ferrari)

0000-0002-1027-3905 (A. Hoang); 0000-0001-5783-8638 (A. Lekssays); 0000-0002-7502-4731 (B. Carminati); 0000-0002-7312-6769 (E. Ferrari)

[1] https://www.aboutamazon.com/news/innovation-at-amazon/making-search-easier

providers to train their entities' embeddings in federated learning settings. FKGE allows each provider to connect to another provider and improves their entities' embeddings. However, because each provider only connects to one provider at a time, connecting to and updating their embeddings from all providers will take a long time.

To address these limitations, we propose a new privacy-preserving decentralized learning framework for knowledge graph embeddings. Our approach differs from FKGE [8] in that we enable a peer to enhance its embeddings using the embeddings of multiple peers simultaneously, whereas FKGE [8] restricts each peer to connect to only one peer at a time for embedding improvement. Furthermore, FedE [2] relies on a trusted server to aggregate embeddings, whereas our work is fully decentralized. Because the training is done asynchronously, the proposed architecture (1) speeds up the embeddings' training and (2) improves the embeddings' quality because they come from a variety of sources [2]. The proposed framework is based on IOTA, a permissionless distributed ledger, where users can submit the metadata of their trained embeddings and store the embeddings on IPFS, a distributed filesystem. When a user wants to train embeddigns for the same task (e.g., link prediction, entity classification), he/she takes the most recent embeddings of the task and aggregates them with her own ones. In turn, the updated embeddings are then shared on the distributed ledger. We use DP to protect the user's privacy and prevent attackers from inferring data from the shared embeddings.

This paper is organized as follows. Section 2 reviews related work. Section 3 illustrates the problem statement, threat model and requirements of the proposed platform. In Section 4, we introduce the proposed architecture, whereas we discuss how the platform addresses the requirements in Section 5. We conclude this work in Section 6.

## 2. Related Work

In this section, we review state-of-the-art techniques for knowledge graph embeddings, differential privacy, and federated learning.

### 2.1. Knowledge Graph Embeddings

A Knowledge Graph (KG) is composed of edges (aka triplets) consisting of a head entity, a tail entity, and a relationship connecting them. For instance, $(Ken, job, Student)$ is a triplet expressing that $Ken$'s $job$ is $Student$. Knowledge graph embeddings (KGEs) are low-dimensional vectors representing entities and relations in knowledge graphs (KGs) such that the semantics between the entities and relations are preserved. The embeddings can be used as inputs in deep learning models (e.g., Convolution network, Full-connected network) to support various tasks such as link prediction, and entity classification. To preserve the semantics, the embeddings must be trained by using KGE models [9], whose train score functions estimate the plausibility of KGs' edges. KGE models can be classified into translational distance, semantic matching, and neural network models.

Translational distance models (e.g., TransE [9]) measure the plausibility of an edge as the distance between its entities' embeddings. TransE [9] first represents entities and relations as vectors with equal dimensions. Then, it estimates the plausibility of an edge through a scoring function that measures the distance between the edge's head and tail entity's embeddings which are connected by the edge's relation's embedding. By minimizing the function, the embedding of the tail entity should be close to the embedding of the head entity plus the relation's embedding. TransE has been extended to TransD, TransH, and TransR [9] to deal with 1-N, N-1, and N-N relations.

Semantic matching models (e.g., RESCAL, DisMult [9]) exploit similarity-based distance to measure the plausibility of an edge by matching their embeddings' dimensions. RESCAL [9] associates each entity with a vector, while each relation is represented by a matrix. The score function measures the plausibility of an edge by using the bilinear function of its entities' embeddings and its relation's matrix. DisMult [9] simplifies RESCAL by using diagonal matrices.

Neural network models use neural network techniques (e.g., convolution network, graph convolution network, neural network) to measure the score functions. In NTN [9], the neural network takes as input the embeddings of an edge's entities. Then, by training the network with KGs' edges, it learns the plausibility of KGs' edges. Instead of taking as input an edge, RGCN [10] receives an entity's features and its neighbors' features. Then, RGCN uses a graph convolution network to infer the edge's embedding.

In this work, we implemented a decentralized learning platform that allows data providers to train all of the above-mentioned KGEs without sharing the local KGs.

### 2.2. Differential Privacy

Differential privacy (DP) [4] has been presented to extract information from a dataset while hiding the existence of its entities. This is done by adding noises to the information before sharing it. The amount of noise required to be added depends on privacy parameters (e.g., $\epsilon$, $\delta$), and the sensitivity of the extraction function. Here, $\epsilon$ illustrates how similar the noisy information and the original ones are, whereas $\delta$ is the probability that DP fails to protect entities' privacy. The sensitivity of a function is the maximum change of the function's outputs when removing a single entity from the dataset.

DP-SGD [5] was developed to avoid making assumptions about the presence of entities in KG, whose data is used to train deep learning models. To this end, it adds noise to the models at every epoch. However, the higher the number of epochs, the more noises are added. Since the noises reduce the quality of the trained models (e.g., classification accuracy), DP-SGD generates low-quality models when training with a high number of epochs. PATE [6] addresses this issue by training models from privacy-aware datasets, that is, whose data are non-sensitive and whose labels are generated by DP. In particular, it creates many models, called teacher models, each of which is trained on a non-overlapped part of the original datasets. The teacher models are then used to generate labels for public datasets. Finally, PATE uses the public datasets and their generated labels to train the final models. The higher the size of the public datasets, the more noise PATE adds. PATE-GAN [7] has been introduced to reduce the amount of added noises by using GAN to learn the labels. In this paper, we apply PATE and PATE-GAN to generate privacy-aware KGs' embeddings.

### 2.3. Federated Learning

Federated learning allows many providers to collaboratively train their models without trusted centralized servers [11]. Its great success on image datasets led to the applications of federated learning in training KGEs. FedE [2] is the first federated learning technique allowing providers to train KGEs for link prediction tasks without sharing their local KGs. The training process starts with a trusted server finding the set containing all entities from all of the providers. Then, it initializes a random embedding for each entity. In each iteration, the server sends the current version of its embeddings to providers. Each provider keeps updating the received entity embeddings with their local KGs. When the training is finished, all providers send the updated local embeddings to the server. The server aggregates all of the local entity embeddings to create a new version of the embeddings and continues the next iteration. The training process stops after a fixed number of iterations. Although the providers do not share their KGs, attackers can exploit membership attacks on the entity embeddings sent from the providers to infer the existence of entities/triplets used to train the embeddings. Moreover, the servers can compromise users' privacy since they know which local KGs contain users' data. If the servers collude with a client, they can also reconstruct the local KGs [3]. FedR [3] has been introduced to protect entities' privacy by not sharing the entities' identities and embeddings but the relations' ones. However, although the identities are not shared, the providers can still violate entities' privacy by using the shared relations' embeddings. FKGE [8] remedies the issues by creating a distributed learning platform using PATE-GAN [7]. In particular, each provider sequentially connects to another one and uses PATE-GAN [7] to train its KG's embeddings. If the trained embeddings have higher quality than the old ones, the provider updates its embeddings. However, since FKGE only connects to a provider at a time, it takes a long time for all providers to train their KGEs.

In this work, we develop a platform that allows a provider to use embeddings of many providers at the same time. Therefore, we can improve the time required for all providers to train their KGEs while applying PATE-GAN [7] to protect entities' privacy.

## 3. Privacy-preserving decentralized learning of knowledge graph embeddings

In this section, we first introduce the problem statement. Then, we explain the privacy and security requirements.

### 3.1. Problem Statement

Given a set of providers $P$, each provider $p_i \in P$ holds a knowledge graph $G_i$, formally defined as $G_i(E_i, R_i, T_i)$, where $E_i$, $R_i$, and $T_i$ are the set of nodes (i.e., entities and attribute values), relations, and triplets of $G_i$. Each triplet $(h, r, t) \in T_i$ is an edge of $G_i$ illustrating the relations between nodes in $G_i$, where $h, t \in E_i$ and $r \in R_i$. Each provider $p_i$ wants to train embeddings of its entities (denoted as $\mathcal{E}_i$) and relations ($\mathcal{R}_i$) for a specific task (e.g., link prediction, node classification) without sharing its KG $G_i$.

To this end, the provider $p_i$ first initializes its entities' and relations' embeddings. Then, at time $t$, it collects the embeddings shared by other providers $P_{-i} \subseteq P \setminus \{p_i\}$ whose train their embeddings for the same task. Let $\mathbb{E}^t_{-i}$ be the set of collected embeddings till time $t$. It aggregates the embeddings in $\mathbb{E}^t_{-i}$ with its embeddings at time $t-1$ ($\mathcal{E}^{t-1}_i, \mathcal{R}^{t-1}_i$) to obtain the initial embeddings at time $t$: $\mathcal{E}^t_i, \mathcal{R}^t_i$. Then, it trains $\mathcal{E}^t_i, \mathcal{R}^t_i$ with its local KGs and evaluates the quality of the trained embeddings. If the quality is improved, it shares $\mathcal{E}^t_i, \mathcal{R}^t_i$ and continues selecting other providers' embeddings until it cannot improve its embeddings anymore. In Section 4, we present our platform's architecture supporting this scenario.

### 3.2. Privacy and security requirements

Federated learning and decentralized learning paradigms were proposed as a first step towards preserving the privacy of users' data since the training is done locally and the users share only the weights of the models [2]. Such paradigms, however, have been shown to be vulnerable to the inference attack [12]. In addition, training models in federated/decentralized settings are vulnerable to other security attacks (e.g., poisoning attacks) where attackers try to manipulate the models to predict specific classes wrongly or lower its performance regarding all the classes [13]. Federated/decentralized learning of KGEs shares the same issues. In this section, we analyze the privacy and security of decentralized learning of KG embeddings under the following threat model. We start by the assumptions on providers and attackers.

**Providers' Assumptions.** We assume that the majority of providers in the system are honest, and that, given a specific model, the majority of actors training that model are honest and have the same training objective.

The trained embeddings in such paradigms are vulnerable to several security attacks (e.g., poisoning attacks, both for data and models, backdoor attacks, etc.) [14]. These attacks can be performed in a coordinated way which brings up the need to mitigate collusion attacks as well [15]. In other words, the data providers could collude with each other.

**Attacker's Goal.** For security, the attacker intends to manipulate the updates of the trained embeddings by injecting malicious updates into the system. These updates can be random or crafted to manipulate the prediction of the model using the trained embeddings. For the crafted updates, we focus on two well-known attacks in decentralized learning, namely label-flipping and backdoor attacks [13]. In a label-flipping attack, the attacker flips the labels of the local training samples from one source class to another target class, while keeping the other classes unchanged. Backdoor attacks involve the attacker embedding special patterns into the original training samples, such as patches of pixels, and changing their labels to the target label. The patterns are a trigger for the target class [16]

For the privacy attacks, we consider inference attacks [12]. Here, we assume providers can access shared embeddings, which are trained on local KGs. Thus, the privacy attacks allow the providers to infer the existence of triplets used to train the embeddings.

**Attacker's Capabilities.** The attackers can individually or collaboratively perform the above-mentioned attacks. We assume that each malicious actor (i.e., provider) can manipulate her own training data (i.e., KGs), but he/she cannot access or manipulate other actors' data. For an attack to succeed, an attacker must have an influence on a target class more than the total influence of the honest clients on that class. So, an attacker might try to collude with other attackers to have the same objective of manipulating the prediction for a target class. The proposed defense is expected to be robust against colluding attacks as well.

To perform the inference attacks, a provider must obtain the shared entities' and relations' embeddings. Then, the provider can try to analyze the embeddings with the background knowledge it has. The proposed defense must be invulnerable to the analysis without having any assumption on the knowledge it uses.

## 4. Architecture

In this section, we proposed the general architecture in support of privacy-preserving decentralized learning of KGEs. It relies on a public distributed ledger technology called IOTA [17] to store embeddings' metadata and on a decentralized filesystem called IPFS [18] to store the embeddings' weights. Figure 1 shows the overall architecture of the system.

The proposed system is a fully decentralized learning framework where users with the same training objective can train embeddings collaboratively. In addition, it supports the training of multiple embeddings asynchronously. Training a KGE model begins with submitting a transaction (i.e., message) to IOTA containing the model metadata (i.e., KG embeddings) such as the model identifier, the path to model weights in IPFS, and the parent model updates that were aggregated from[3]. As a result, each model update refers to other $k$ model updates that came before it. In other words, each model update aggregates model updates that were submitted before. This aggregation is done only if old model updates improve the accuracy of the user's local model. Hence, typically, the latest model updates are better versions of the same model. So, each model will be represented as a directed acyclic graph as represented in Figure 1. It is worth noting that the parameter $k$ is chosen by the deployer of the system.

We use DP to prevent privacy attacks on shared embeddings. To prevent DP from adding too much noise, we allow each provider to specify the upper-bound threshold on the privacy budgets (i.e., $\epsilon_i$, $\delta_i$ for provider $p_i$). When training the updated embeddings with differential privacy techniques (e.g., DP-SGD[5], PATE-GAN[7]), a provider can estimate the current privacy costs by using sequential and parallel composition. In particular, let $\epsilon_i^t$ and $\delta_i^t$ be the current privacy budgets at time $t$ of provider $i$. By training with DP-SGD[5], the provider

[2]In this paper, we use models' weights and embeddings interchangeably.

[3]This field is empty for the initial transaction of a model, which is called a genesis transaction.

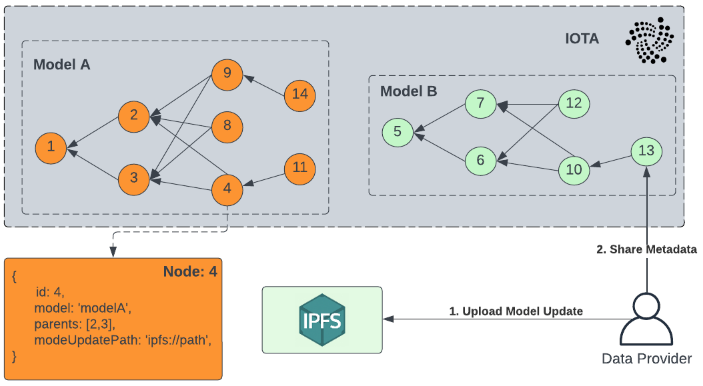

**Figure 1:** The proposed architecture for privacy-preserving decentralized learning of knowledge graph embeddings

can increase $\epsilon_i^t$ and $\delta_i^t$ by its privacy parameters (i.e., $\epsilon_i$ and $\delta_i$) every epoch. Training with PATE-GAN[7], the provider can increase $\epsilon_i^t$ and $\delta_i^t$ according to the number of triplets of its public KG. Here, DP-SGD and PATE-GAN generate noises such that the existence of any triplets in $G_i$ is hidden. Since privacy budgets can be calculated before training, the provider can easily estimate whether they are greater than the upper-bound threshold. If this is the case, the provider stops the training. Otherwise, it starts training the new embeddings. After the training process is finished, it shares the embeddings and the privacy budgets used to train them.

Honest providers will report their embeddings and the privacy budgets used to train them. In the case of malicious providers, they might aim to disturb the learning phase by setting up low $\epsilon$ and high $\delta$. Thus, DP adds more noise to the embeddings and the probability that it cannot prevent privacy attacks is also increased (Section 2.2). However, in this case, the malicious providers' embeddings will be noisy and not be used for training by honest clients.

## 5. Discussion

In this section, we show how the proposed platform addresses security and privacy requirements presented in Section 3. In addition, we discuss the scalability of our framework based on the architecture shown in Figure 1.

### 5.1. Security

The proposed framework ensures transparent, multi-model, decentralized learning. However, training models without the validation of shared weights would lead to manipulating their predictions. Model updates might be malicious, as they could be crafted to initiate (individually or collaboratively) poisoning and backdoor attacks discussed in Section 3.2. In this section, we discuss possible mitigations of these attacks. Since we assume that the majority is honest, honest actors can check the validity of the model updates by considering the actor's local model a reference (i.e., global model) and computing the euclidean distance or cosine similarity to exclude adversarial updates [19, 13]. These techniques could be extended to the mitigation of such attacks when performed in a collaborative manner (i.e., Sybil attacks) [20, 13]. This line of defense is based on the observation that malicious actors have a similar objective. Hence, when computing their cosine similarities, the angle between their model updates will be slight and the cosine similarity will be higher. However, such defenses fail when the malicious actors submit random weights (i.e., untargeted attacks). In other words, the malicious actors do not have any common objective. Thus, there is a need to combine such defenses with other defenses such as KRUM, Multi-Krum, Trimmed Mean, etc. [21] that are able to discard such updates. It is worth noting that such defenses can discard model updates coming from honest clients with strict privacy budgets since their updates could be considered random, leading to untargeted attacks. Other defenses based on Trusted Execution Environment [22] or manipulation of the models to reduce their sizes [23] were proposed. However, they come with major utility limitations that cannot be extended to a decentralized setting. The discussed defenses are designed to work in centralized environments where there is a central aggregation server that runs them. So, there is a need to replace the centralized component when validating model updates. This limitation could be addressed by forming a model-specific, randomly and periodically elected committee that reviews a batch of model updates using the aforementioned defenses. It is worth noting that such committees, often called dynamic committees are used in Proof-of-Stake blockchains such as Algorand [24].

Since some honest updates with relatively small privacy budgets could be discarded, selecting the threshold for discarding model updates is very crucial in maintaining good training performance in decentralized learning. However, the combination of security techniques with privacy requirements and their possible side effects remain a future work.

### 5.2. Privacy

The proposed framework prevents the inference attacks (Section 3.2) by only sharing the privacy-aware embeddings that are trained with DP techniques (e.g., DP-SGD [5], PATE-GAN [7]). Here, providers add noises such that their trained embeddings are similar whether a triplet is used to train them. Therefore, according to the DP Sequential and Parallel Composition [5, 7], the existence of any triplets they used to train their embeddings is hidden, no matter what background knowledge dishonest providers use.

### 5.3. Scalability

The scalability of our framework is tied up to the way we handle model updates. Since each node keeps a directed acyclic graph of each model, adding nodes (i.e., model updates) to the graph is not a heavy task since only the metadata is appended to the nodes. Our framework relies on IPFS for storing raw model updates and IOTA for storing metadata. Hence, it does not handle the burden of storage, but rather it connects through HTTP to both IPFS and IOTA. So, the clients download only the model updates that they are interested in.

## 6. Conclusion

In this paper, we highlight the security and privacy requirements of decentralized knowledge graph representation and propose an architecture for training such models. We presented the different privacy-preservation techniques used in knowledge graphs and the security defenses that mitigate various types of attacks in machine learning, such as poisoning attacks and backdoor attacks, when performed individually or collaboratively (i.e., Sybil attacks). We discussed the limitations that the interplay between security and privacy causes. The solutions and analysis of this interplay remain a future work.

## Acknowledgments

This work has received funding from the Marie Skłodowska-Curie Innovative Training Network Real-time Analytics for Internet of Sports (RAIS), supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 813162. Additionally, it has been partially supported by CONCORDIA, the Cybersecurity Competence Network supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 830927. The content of this paper reflects the views only of their author(s). The European Commission/Research Executive Agency are not responsible for any use that may be made of the information it contains.

## References

[1] A. Gogleva, D. Polychronopoulos, M. Pfeifer, V. Poroshin, M. Ughetto, M. J. Martin, H. Thorpe, A. Bornot, P. D. Smith, B. Sidders, et al., Knowledge graph-based recommendation framework identifies drivers of resistance in egfr mutant non-small cell lung cancer, Nature communications 13 (2022) 1–14.

[2] M. Chen, W. Zhang, Z. Yuan, Y. Jia, H. Chen, Fede: Embedding knowledge graphs in federated setting, in: The 10th International Joint Conference on Knowledge Graphs, IJCKG'21, Association for Computing Machinery, New York, NY, USA, 2021, p. 80–88. URL: https://doi.org/10.1145/3502223.3502233. doi:10.1145/3502223.3502233.

[3] K. Zhang, Y. Wang, H. Wang, L. Huang, C. Yang, L. Sun, Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation, CoRR abs/2203.09553 (2022). URL: https://doi.org/10.48550/arXiv.2203.09553. doi:10.48550/arXiv.2203.09553. arXiv:2203.09553.

[4] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–12.

[5] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 308–318. URL: https://doi.org/10.1145/2976749.2978318. doi:10.1145/2976749.2978318.

[6] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, K. Talwar, Semi-supervised knowledge transfer

for deep learning from private training data, in: International Conference on Learning Representations, 2017. URL: https://openreview.net/forum?id=HkwoSDPgg.

[7] J. Jordon, J. Yoon, M. van der Schaar, PATE-GAN: generating synthetic data with differential privacy guarantees, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019. URL: https://openreview.net/forum?id=S1zk9iRqF7.

[8] H. Peng, H. Li, Y. Song, V. Zheng, J. Li, Differentially private federated knowledge graphs embedding, in: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, CIKM '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1416–1425. URL: https://doi.org/10.1145/3459637.3482252. doi:10.1145/3459637.3482252.

[9] Y. Dai, S. Wang, N. N. Xiong, W. Guo, A survey on knowledge graph embedding: Approaches, applications and benchmarks, Electronics 9 (2020) 750.

[10] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, M. Alam (Eds.), The Semantic Web, Springer International Publishing, Cham, 2018, pp. 593–607. doi:10.1007/978-3-319-93417-4_38.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: A. Singh, X. J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, volume 54 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 1273–1282. URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

[12] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: Proceedings of the 12th ACM workshop on artificial intelligence and security, 2019, pp. 1–11.

[13] S. Awan, B. Luo, F. Li, Contra: Defending against poisoning attacks in federated learning, in: Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26, Springer, 2021, pp. 455–475.

[14] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Generation Computer Systems 115 (2021) 619–640.

[15] C. Fung, C. J. Yoon, I. Beschastnikh, The limitations of federated learning in sybil settings., in: RAID, 2020, pp. 301–316.

[16] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks (2017).

[17] S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer, et al., The coordicide, Accessed Jan (2020) 1–30.

[18] J. Benet, Ipfs-content addressed, versioned, p2p file system, arXiv preprint arXiv:1407.3561 (2014).

[19] D. Cao, S. Chang, Z. Lin, G. Liu, D. Sun, Understanding distributed poisoning attack in federated learning, in: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2019, pp. 233–239.

[20] C. Fung, C. J. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, arXiv preprint arXiv:1808.04866 (2018).

[21] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, Advances in neural information processing systems 30 (2017).

[22] F. Mo, H. Haddadi, Efficient and private federated learning using tee, in: Proc. EuroSys Conf., Dresden, Germany, 2019.

[23] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, L. Tassiulas, Model pruning enables efficient federated learning on edge devices, IEEE Transactions on Neural Networks and Learning Systems (2022).

[24] J. Chen, S. Gorbunov, S. Micali, G. Vlachos, Algorand agreement: Super fast and partition resilient byzantine agreement, Cryptology ePrint Archive (2018).