Differential Privacy Preservation for Graph-based Federated Learning under Malware Attacks

Mohamed Amjath
Department of Computing
Atlantic Technical University
Donegal, Ireland
amjaath22@gmail.com

Shagufta Henna
Department of Computing
Atlantic Technical University
Donegal, Ireland
shagufta.henna@atu.ie

***Abstract*—Recently, there has been an increasing emphasis on mitigating cyber attacks in federated learning (FL) under the Internet of Health Things (IoHTs). One prominent attack in FL is membership inference, in which an adversary attempts to determine whether a specific sample was included in the training dataset used during the FL process. To address membership inference attacks (MIA), various privacy-preserving approaches have been proposed, with the widespread adoption of differential privacy stochastic gradient (DP-SGD). However, these approaches primarily focus on preserving privacy for individual data points and do not account for the interactions commonly found in cascaded function calls within malware.**

To overcome this limitation, this work initially models the malware dataset as a function call graph (FCG). Subsequently, the DP-SGD-learned DotGAT model is utilized to classify both malware and benign applications, ensuring the preservation of privacy while maintaining model utility. Experimental results demonstrate that maintaining a privacy budget (ε) within the range of 12 to 18 achieves a favorable balance between privacy and accuracy, thereby affirming the practicality and effectiveness of our approach.

***Index Terms*—Differential Privacy, Membership Inference Attack, Federated Learning**

I. Introduction

Federated learning has emerged as a state-of-the-art approach, utilizing multiple computing resources to train deep neural networks on large datasets [1]. This technique has gained significant attention due to its focus on privacy preservation while also expediting the training process and effectively handling complex tasks. The collaborative nature of distributed deep learning facilitates faster convergence and enhanced performance [2]. Despite having numerous advantages, data privacy in FL is still vulnerable to various security threats that can compromise the integrity and confidentiality of the training data. Adversaries can launch diverse attacks under FL, giving rise to significant privacy and security concerns. Data poisoning attacks involve the injection of malicious data samples, which can result in biased or erroneous model outputs [3]. Model inversion attacks aim to infer sensitive training data from the outputs of a model, thus compromising the privacy of individuals [4]. The membership inference attack seeks to determine if a particular data point was included in the training dataset [4]. Byzantine attacks encompass machines deviating from correct computations, potentially disrupting the training process and compromising the integrity of the model [5]. These attacks require robust defenses and privacy-preserving mechanisms under distributed learning scenarios.

This work focuses MIA under a FL scenario with a focus on privacy and security. The successful execution of MIA can lead to the exposure of sensitive or confidential information contained within the training dataset, as well as the infringement upon individuals' privacy rights. Such attack can have adverse effects on healthcare, finance, and personal data, as it undermines the trust and integrity of machine learning models and the associated datasets.

To mitigate these consequences, various privacy-preserving techniques have been proposed. These include approaches such as differential privacy [6], privacy-preserving aggregation [7], data augmentation [8], adversarial training [9], and model distillation [10]. These methods employ strategies such as introducing noise, secure aggregation, or data transformations to safeguard individual data privacy and thwart adversaries attempting to infer membership information from trained models.

Differential privacy provides mathematical guarantees to protect individuals' data during data analysis and machine learning processes [6]. It achieves this by introducing carefully calibrated noise to the analysis results, thereby preventing the identification of specific individuals or the disclosure of sensitive information [6]. Various mechanisms and algorithms, such as randomized response, Gaussian noise addition, stochastic gradient descent (DP-SGD), private aggregation of teacher ensembles, and secure multi-party computation, have been developed to achieve differential privacy [11]. These techniques incorporate randomness into the data or analysis process to ensure that the final results are not excessively influenced by any particular individual's data.

Limited efforts have been made towards applying differential privacy under FL to address different types of attacks. Abadi et al. [12] proposed the DP-SGD algorithm, which trains machine learning models with privacy guarantees. DP-SGD modifies the standard gradient descent algorithm by introducing noise to the gradients, ensuring privacy while preserving model utility. Subsequent works, such as those by McMahan et al. [13] and Kairouz et al. [14], proposed advanced techniques like adaptive noise injection and moment accountant for tighter privacy bounds.

In other works [15], [16], DP-SGD implements an FL scenario to perform classification tasks related to pneumonia, and liver disease, respectively. These studies demonstrate the effectiveness of incorporating DP techniques into the training process to protect sensitive medical data. In another study, Le Fang et al. presented DP-SGD with variational autoencoders to develop a privacy-enhanced movie recommendation system that preserves user preferences while providing personalized suggestions [17]. The aforementioned studies primarily focus on quantifying the differential privacy loss for each data point in the dataset. However, when utilizing a graph-based dataset in a federated learning setup, existing approaches fail to account for the interactions present in the graph. Consequently, this leads to an inaccurate estimation of differential privacy for the graph dataset.

To overcome the limitations of the DP-SGD, which solely considers privacy loss on individual data points, this research proposes an alternative approach that captures the cumulative privacy loss of interactions within a graph model. Specifically, the study introduces a novel FCG representation for the malware dataset. The FCG interprets the flow of control between different functions in an application while also illustrating their relationships [22]. In addition to that, in this representation, crucial information contained within the functions can be assigned as node features of the FCG [22]. Therefore, this approach facilitates processing the graph data through graph-based algorithms easily. To perform the analysis, the graph data is processed using the dot product graph attention network (DotGAT), which applies dot product operation to address the computation complexity present in large-scale graphs. Furthermore, to ensure privacy preservation, the proposed method incorporates differential privacy using DP-SGD.

The rest of this paper is organized as follows. Section II examines the MIA. Section III provides a comprehensive explanation of the DP-SGD approach. Section IV focuses on the dataset and the creation of FCG. Section V introduces the DP-DotGAT architecture and algorithm. Section VI covers the experiment setup. Section VII presents the performance evaluation, while Section VIII concludes the paper.

II. Threat Model

The primary objective of MIA is to determine whether a specific data sample is included in the training process of a machine learning model. These attacks exploit vulnerabilities within the model to gain insights into the membership status of individual data points. Figure 1 shows different steps of MIA attack:

1) Gain network access: The attacker joins a distributed network as a legitimate user.
2) Obtaining target model: The attacker acquires the parameter of the target machine learning model, denoted as $f_{target}()$, which has been trained on a labeled dataset, $D_{train-target}$.
3) Generating shadow dataset: The attacker creates a separate dataset, $D_{train-shadow}$, to train the shadow model, $f_{shadow}()$. It is important to note that the shadow dataset is disjoint from the private target dataset used for training the target model (i.e., $D_{train-shadow} \bigcap D_{train-target} = \varnothing$).
4) Training shadow model: The attacker trains the shadow model, $f_{shadow()}$, using the generated shadow dataset, $D_{train-shadow}$. The shadow model is designed to replicate the behavior of the target model, producing similar output probabilities for given inputs.
5) Preparing attack model: The attacker collects inputs for the attack model, $f_{attack}()$, which consists of a predicted probability vector from the shadow model and the corresponding true label.
6) Training attack model: The attacker trains the attack model, $f_{attack()}$, using the collected inputs. The attack model functions as a binary classifier, aiming to distinguish between samples that are members of the target model's training dataset and those that are not.
7) Performing membership inference: Once the attack model is trained, leveraging the similarities and differences between the shadow and target models, it accurately infers membership.

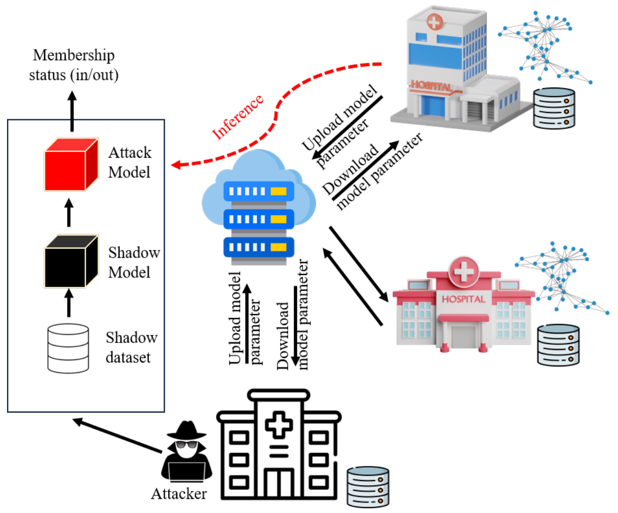

Fig. 1. Threat model for membership inference attack: The adversary exploits shadow model to infer membership in a trained machine learning model.

III. Preliminaries

This section presents preliminaries and definitions for the DP-SGD under graph-based FL. The differential privacy algorithm depends on two parameters that are defined below:

- ε is the privacy parameter controlled by the data analyst to balance the trade-off between privacy and accuracy.
- δ is the parameter that represents the probability of privacy leakage in (ε, δ) differential privacy.
- D1 and D2 be neighboring datasets that differ by only one element.

Definition 3.1 (ε - Differential Privacy):

Let ε be a positive value. A randomized function M is said to be ε-differentially private if, for any pair of neighboring input datasets D1 and D2 that differ by at most one element and $\forall S \subseteq Range(M)$, we have Equation 1 based on [6].

$$\frac{P[M(D1)] \in S]}{P[M(D2)] \in S]} \leq e^{\varepsilon} \tag{1}$$

where the probability is determined based on the coin tosses of M [6]. The given Equation 2 can be written as:

$$P[M(D1)] \in S] \geq e^{\varepsilon}.P[M(D2)] \in S] \tag{2}$$

The probability of output in S on a D1 dataset is at least e times the probability of output in S on a D2 dataset.

Definition 3.2 ((ε, δ) - Differential Privacy):

Define a random function M as (ε, δ)-differentially private if for all neighboring input datasets D1 and D2 that differ by at most one element and $\forall S \subseteq Range(M)$, we have Equation 3 based on [18].

$$P[M(D1) \in S] \geq exp(\varepsilon) \times P[M(D2) \in S] + \delta \tag{3}$$

Suppose, for instance, that Y is an output that possibly reveals X's identity or data, whereas the parallel dataset D2 does not contain X's data, so we can say that $P[M(D2) \in S] = 0$. In this scenario, M can never output X on any dataset but (ε, δ) - differential privacy may produce X with a probability up to δ.

In the above definitions, the Gaussian mechanism is utilized to achieve (ε, δ)-differential privacy [19] as described below. In this mechanism, a zero-mean Gaussian noise is applied to the query result. The addition of noise to the output is scaled based on the l2 sensitivity.

Definition 3.3 (Gaussian Mechanism):

Given a function f(x) that outputs a number, the following definition of g(x) meets (ε, δ)-differential privacy, we can express Equation 4 by incorporating σ^2 given in Equation 5.

$$\text{g(x)=f(x)+Gaussian}(\sigma^2) \tag{4}$$

$$\sigma^2 = \frac{2\Delta_f^2 log(1.25/\delta)}{\epsilon^2} \tag{5}$$

where Δ_f^2 is the l2 sensitivity ($\Delta_f^2 = \overset{max}{D1,D2} \|f(D1 - f(D2))\|_2$) of the function f and Gaussian(σ^2) is a random sample from the Gaussian distribution with a center of 0 and variance of σ^2.

IV. Dataset

To demonstrate the effectiveness of applying DP-SGD to graph-based malware detection tasks, the **DPMal** dataset is constructed by utilizing two well-known datasets, AndroZoo [20] and Maldroid [21]. AndroZoo contains 17,341 applications, while Maldroid offers an extensive collection of 22,529,765 Android applications. Importantly, both datasets include both malicious and benign applications. To build a diverse and representative dataset, we carefully selected 2,500 distinct applications, including 500 of each: adware, banking malware, riskware, SMS, and benign apps from both datasets. These applications varied in size, ranging from 50 KB to 3.50 MB. After performing a random shuffle, 80% of the DPMal was allocated for training, and the remaining portion for testing purposes.

A. Function Call Graphs

To construct the graph data, the Androguard tool is used, inspired by the work of Vinayaka et al. [22]. The process of building the graph data consists of two main stages. In the first stage, the FCG is extracted from the application by analyzing the dex code and following the function calls. The FCG captures the relationships and connections between different functions within the application [22]. In the second stage, features were assigned to each node in the graph to facilitate message passing and enhance the analysis. Specifically, the node features are obtained by combining API calls and Opcodes. API calls represent the interactions and usage of different application programming interfaces within the function [23]. By considering API calls as node features, insights into the specific functionalities and behaviors of each function were obtained. On the other hand, Opcodes represent the low-level instructions executed by the processor within the function [24]. By including Opcodes as node features, the detailed operations and instructions performed by each function were captured.

V. Differentially Private DotGAT (DP-DotGAT)

This section presents the classification using DotGAT that is trained using DP-SGD as discussed below.

A. DotGAT

When considering a large-scale application with thousands of nodes and links, it is crucial to choose a model that can efficiently handle such complexity. Recently, the DotGAT model has emerged as a powerful approach for graph analysis and processing, demonstrating superior performance compared to other graph neural network (GNN) architectures [25]. By incorporating an attention mechanism, DotGAT effectively captures the importance of neighboring nodes during information propagation, allowing it to adaptively aggregate information from the graph and enhance its ability to capture complex dependencies and patterns in the graph structure [25]. Moreover, DotGAT addresses scalability challenges by employing a computationally efficient dot product attention mechanism, making it suitable for large-scale graphs [25]. Furthermore, the multi-head attention mechanism in DotGAT significantly enhances its capacity to capture complex relationships within the graph. These promising capabilities presented in DotGAT allow them to be used in various tasks to acquire fine-grained information from a graph [22].

Therefore, to improve performance, address computation complexity concerns, and ensure privacy, we employed the DotGAT model in combination with differential privacy techniques. The overall architecture of our approach is illustrated in Figure 2. The model constructed in Section IV-A serves as the input to the DotGAT model, which is trained using DP-SGD. During the training process, gradients are clipped, averaged, and Gaussian noise is added to preserve privacy while learning meaningful patterns from the FCG. The steps involved are presented in Figure 2 and outlined in Algorithm 1.

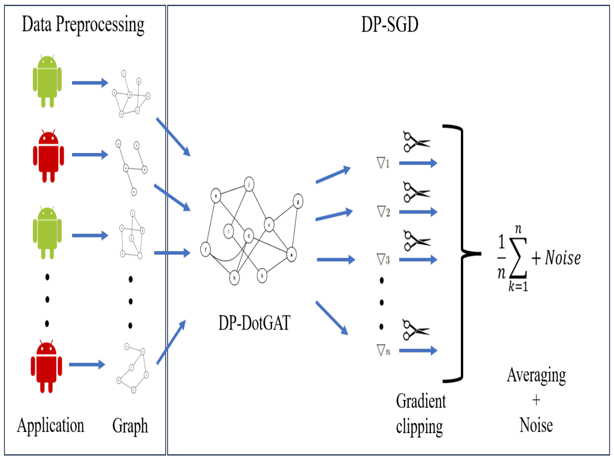

Fig. 2. DP-DotGAT architecture.

Algorithm 1 DP-DotGAT

Input: G(V, E), X, ε_{max}, α, σ, δ, N, C // DPMal, features, max_budget, learning rate, noise scale, leak_probability,sample size, and gradient_norm_bound

Output: $\bar{W}$ and total privacy cost (ε, δ) using a privacy accounting method

```
1:  $H^{(0)} \leftarrow X$
2:  Initialize $W$ randomly
3:  for each epoch $n$ in $N$ do
4:      $\varepsilon \leftarrow$ DP-SGD (N,$\sigma$,$\delta$)
5:      if $\varepsilon > \varepsilon_{max}$ then
6:          Go to step 20
7:      end if
8:      for each head $k$ in $K$ do
9:          e(v,u) $\leftarrow (W * H_v)^{(T)}.W * H_u$ // v,u are connected vertices
10:         $a_{vu} \leftarrow$ softmax(e(v,u)) // Normalize attention coefficients
11:         $H_k^{(v)} \leftarrow \sum_u a_{vu} * H_{k-1}^{(u)}$ // node embeddings
12:         $H^{(v)} \leftarrow$ mean($H_k^{(v)}$)
13:     end for
14:     $Z \leftarrow$ mean($H^{(v)}$) //graph-level embedding
15:     $Y \leftarrow$ softmax($Z * W_{out}$),where $W_{out}$ is the weight matrix for the
        output layer
16:     $Loss \leftarrow -\frac{1}{N}\sum_{i=1}^{N} Y_i * \log \bar{Y_i} + (1 - Y_i) * \log(1 - \bar{Y_i})$ //Calcu-
        late loss
17:     $g \leftarrow \nabla_W * Loss$ //Calculate gradient
18:     g' $\leftarrow$ g/max(1, $\frac{\|g\|_2}{C}$) //Clip gradient
19:     $\bar{g} \leftarrow \frac{1}{N}$(g'+Noise(0,$\sigma^2 C^2 I$)) //Adding noise
20:     $W_{new} \leftarrow W - \alpha(\bar{g})$ // Update weights
21: end for
22: $\bar{W} \leftarrow W_{new}$
```

B. Learning using DP-DotGAT

The DP-DotGAT algorithm takes DPMal, features, maximum budget, learning rate, noise scale, leak probability sample size, and gradient norm bound as inputs and produces local weights and privacy cost (ε, δ) as output. Initially, H(0) is initialized as X. Next, the weights are randomly assigned. The privacy cost is calculated using the DP-SGD algorithm [26] in step 4. In step 9, the attention coefficients for each edge (v,u) in the graph are computed, followed by the application of row-wise Softmax to normalize them in step 10. In step 11, the normalized attention coefficients are utilized to compute a set of weighted feature vectors for each node at attention head k. These vectors are then concatenated in step 12 through a simple average operation. The resulting output is averaged across all nodes in the graph to obtain a graph-level embedding in step 14. The loss factor is computed in step 16 based on the difference between the predicted and actual output, and gradients are subsequently computed in step 17. In step 18, the gradients are clipped based on the gradient norm bound factor. Following that, the clipped gradients are perturbed by adding Gaussian noise in step 19. Subsequently, in step 20, the weights of the model are updated based on the noisy gradients represented by the variable $\bar{g}$. This weight update process is iterated until the privacy cost exceeds $\varepsilon_{\max}$.

VI. Experiment Setup

These experiments are conducted using the Python programming language. Experiments use the Androguard tool to extract the FCG from an Android application. Additionally, PyTorch is utilized for constructing the models. After thorough analysis and evaluation, it is determined that a noise multiplier of 0.5 and an L2 clipping norm of 6 are the optimal parameters for the DP-DotGAT model. During this decision-making process, it is observed that setting the noise multiplier above 0.5 results in poor model performance. Conversely, reducing the noise multiplier below 0.5 led to exponential growth in the privacy budget. With the selected parameters, the model is evaluated using four different privacy budget values ($\varepsilon = 5, 10, 15, 20$). Furthermore, to compare the overall performance of the DP-DotGAT model, the DotGAT (without DP) model is used as a baseline. Table I presents the various parameters used in the experiment.

TABLE I
Model Parameters.

Parameters	DotGAT	DP-DotGAT
Input	247	247
Hidden dimension	3	3
Learning rate	0.0125	0.0125
Optimizer	SGD	SGD
Number of heads	2	2
Weight decay	0.0001	0.0001
L2 norm clipping	-	6
Noise multiplier	-	0.5

VII. Performance Evaluation

The impact of the privacy budget on accuracy is illustrated in Fig. 3. Our analysis shows that the DotGAT model achieves a convergence accuracy of 90.04%. In the case of DP-DotGAT, increasing the privacy budget from 5 to 20 leads to a corresponding accuracy improvement from 70.44% to 92.22%. However, it is important to note that this increase in accuracy comes at the cost of reduced privacy for the model. Furthermore, we observed that the accuracy of the DP-DotGAT model reaches 80% when the privacy parameter ε is set to 12. This indicates that the model can achieve a reasonably high level of accuracy while maintaining satisfactory privacy protection. Interestingly, at a privacy parameter of $\varepsilon = 18$, the accuracy of the DP-DotGAT model surpasses that of the baseline DotGAT model. This behavior can be attributed to the regularization effect of the noise introduced by differential privacy. Therefore, our approach demonstrates that a privacy budget ranging from above 12 to below 18 effectively satisfies privacy requirements while maintaining a satisfactory level of model performance.

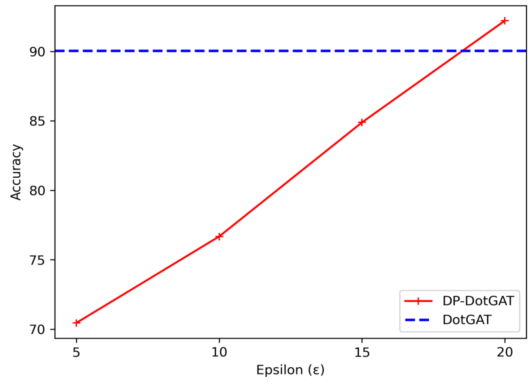

Fig. 3. Accuracy vs privacy budget ε.

Fig. 4 and Fig. 5 present the ROC curves and PR curves for the DP-DotGAT model with various privacy budget settings. The configuration with $\varepsilon = 20$ demonstrates the highest performance, achieving an impressive AUC value of 0.922 and AP value of 0.891. However, this performance gain comes at the cost of significant privacy leakage, indicating a compromise in preserving sensitive information. The DotGAT model shows the second-highest performance, with an AUC of 0.904 and AP of 0.861. Conversely, the setting with $\varepsilon = 5$ yields the lowest performance among the tested configurations, highlighting the inverse relationship between the privacy budget and model performance. These findings emphasize the importance of carefully considering the privacy-performance trade-off when selecting a privacy budget for the DP-DotGAT model.

Fig. 6 and Fig. 7 illustrate the precision and F1 score of both the DotGAT and DP-DotGAT models. As the privacy budget increases, the precision and F1 score of DP-DotGAT exhibit an upward trend. For DP-DotGAT, precision ranges from 71.05% to 91.88%, and F1 scores vary from 70.90% to 92.47% across different privacy budgets (ε). In comparison, the DotGAT model achieves a precision of 88% and an F1 score of 91%, without explicitly considering privacy protection. Remarkably, DP-DotGAT demonstrates superior precision and F1 score compared to DotGAT for privacy budgets after $\varepsilon = 15$ and $\varepsilon = 18$. DP-DotGAT demonstrates enhanced precision and F1 score compared to DotGAT for certain privacy budgets, it is essential to consider the presence of potential privacy vulnerabilities. These vulnerabilities can potentially lead to the disclosure of sensitive information, posing risks to the privacy of individuals.

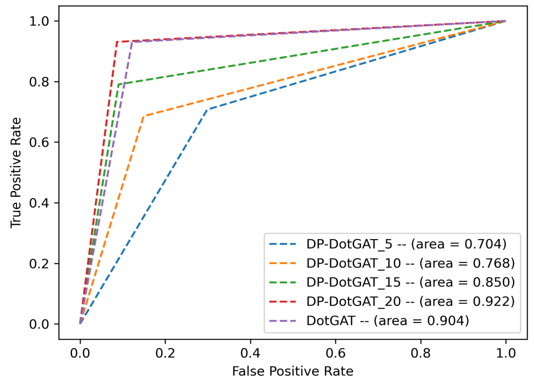

Fig. 4. ROC curve comparison for different ε

.

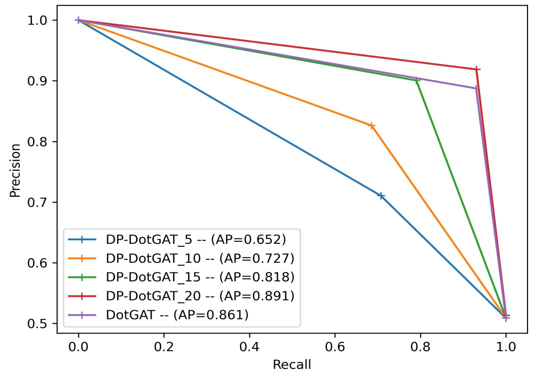

Fig. 5. PR curve comparison for different ε

.

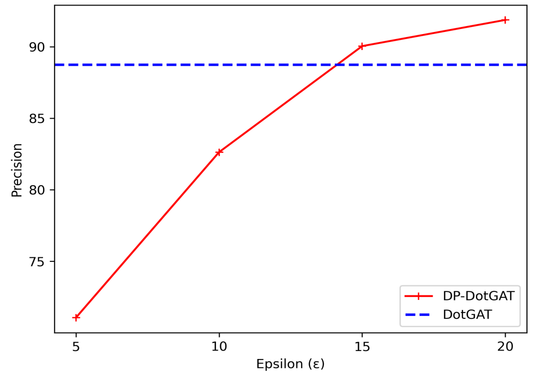

Fig. 6. Precision vs Privacy budget.

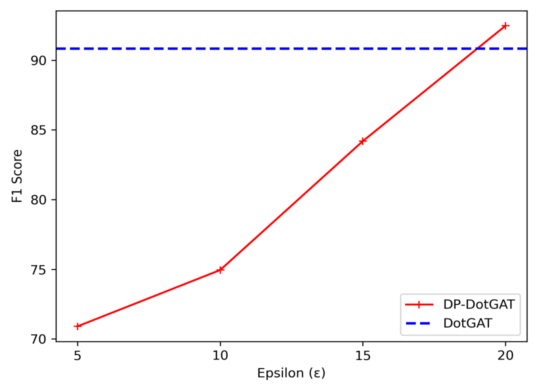

Fig. 7. F1 score vs Privacy budget.

In summary, the analysis highlights that while DP-DotGAT outperforms DotGAT in terms of performance under specific privacy budgets, the existence of privacy vulnerabilities cannot be ignored. Therefore, careful budget setting is crucial when applying DP-DotGAT to application graph data in order to strike a balance between privacy and performance.

VIII. Conclusion

The study provides a privacy-preserving technique to mitigate MIA in an FL setting by integrating the DP-SGD algorithm with the DotGAT model. To introduce randomness, Gaussian noise is added at each local update of the model. Further, the findings indicate that increasing the privacy budget in the DP-DotGAT model enhances accuracy but reduces privacy. Furthermore, choosing a privacy budget ε between 12 and 18 offers a reasonable balance of accuracy and privacy in the DP-DotGAT model. Moreover, our approach overcomes the limitation of solely focusing on individual data points by considering the cascade structure of function calls within the applications. The findings contribute to advancing privacy-preserving techniques in FL and highlight the importance of considering contextual information for improved privacy protection.

References

[1] S. Bharati, M. R. H. Mondal, P. Podder, and V. B. S. Prasath, “Federated learning: Applications, challenges and future directions,” Int. J. Hybrid Intell. Syst., vol. 18, no. 1–2, pp. 19–35, 2022, doi: 10.3233/his-220006.

[2] A. Chaudhuri, A. Nandi, and B. Pradhan, “A Dynamic Weighted Federated Learning for Android Malware Classification,” pp. 1–11, 2022, [Online]. Available: http://arxiv.org/abs/2211.12874.

[3] Y. Chen, Y. Gui, H. Lin, W. Gan, and Y. Wu, “Federated Learning Attacks and Defenses: A Survey,” Proc. - 2022 IEEE Int. Conf. Big Data, Big Data 2022, pp. 4256–4265, 2022, doi: 10.1109/BigData55660.2022.10020431.

[4] A. Hatamizadeh et al., “Do Gradient Inversion Attacks Make Federated Learning Unsafe?,” IEEE Trans. Med. Imaging, pp. 1–1, 2023, doi: 10.1109/tmi.2023.3239391.

[5] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang, “Challenges and Approaches for Mitigating Byzantine Attacks in Federated Learning,” pp. 1–8, 2021, doi: 10.1109/TrustCom56396.2022.00030.

[6] C. Dwork, “Differential privacy,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 4052 LNCS, pp. 1–12, 2006, doi: 10.1007/11787006_1.

[7] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage A. Segal and Karn Seth, “Cryptology ePrint Archive: Report 2017/281 - Practical Secure Aggregation for Privacy-Preserving Machine Learning,” vol. 5, no. 1, pp. 7–22, 2019, [Online]. Available: https://eprint.iacr.org/2017/281.

[8] D. Yu, H. Zhang, W. Chen, J. Yin, and T. Y. Liu, “How Does Data Augmentation Affect Privacy in Machine Learning?,” 35th AAAI Conf. Artif. Intell. AAAI 2021, vol. 12B, no. Mi, pp. 10746–10753, 2021, doi: 10.1609/aaai.v35i12.17284.

[9] C. Pauling, M. Gimson, M. Qaid, A. Kida, and B. Halak, “A Tutorial on Adversarial Learning Attacks and Countermeasures,” 2022, [Online]. Available: http://arxiv.org/abs/2202.10377.

[10] A. Alkhulaifi, F. Alsahli, and I. Ahmad, “Knowledge distillation in deep learning and its applications,” PeerJ Comput. Sci., vol. 7, pp. 1–24, 2021, doi: 10.7717/peerj-cs.474.

[11] M. Aitsam, “Differential Privacy Made Easy,” pp. 1–7, 2023, doi: 10.1109/etecte55893.2022.10007322.

[12] M. Abadi et al., “Deep learning with differential privacy,” Proc. ACM Conf. Comput. Commun. Secur., vol. 24-28-Octo, no. Ccs, pp. 308–318, 2016, doi: 10.1145/2976749.2978318.

[13] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” 6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc., pp. 1–14, 2018.

[14] P. Kairouz, S. Oh, and P. Viswanath, “The Composition Theorem for Differential Privacy,” IEEE Trans. Inf. Theory, vol. 63, no. 6, pp. 4037–4049, 2017, doi: 10.1109/TIT.2017.2685505.

[15] A. Ziller, D. Usynin, R. Braren, M. Makowski, D. Rueckert, and G. Kaissis, “Medical imaging deep learning with differential privacy,” Sci. Rep., vol. 11, no. 1, pp. 1–8, 2021, doi: 10.1038/s41598-021-93030-0.

[16] M. Adnan, S. Kalra, J. C. Cresswell, G. W. Taylor, and H. R. Tizhoosh, “Federated learning and differential privacy for medical image analysis,” Sci. Rep., vol. 12, no. 1, pp. 1–10, 2022, doi: 10.1038/s41598-022-05539-7.

[17] L. Fang, B. Du, and C. Wu, “Differentially private recommender system with variational autoencoders,” Knowledge-Based Syst., vol. 250, p. 109044, 2022, doi: 10.1016/j.knosys.2022.109044.

[18] M. Hardt, K. Ligett, and F. McSherry, “A simple and practical algorithm for differentially private data release,” Adv. Neural Inf. Process. Syst., vol. 3, pp. 2339–2347, 2012.

[19] B. Balle and Y.-X. Wang, “Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising BT - International Conference on Machine Learning,” Int. Conf. Mach. Learn., pp. 403–412, 2018.

[20] L. Li et al., “AndroZoo++: Collecting Millions of Android Apps and Their Metadata for the Research Community,” 2017, [Online]. Available: http://arxiv.org/abs/1709.05281.

[21] “MalDroid 2020 — Datasets — Research — Canadian Institute for Cybersecurity — UNB.” https://www.unb.ca/cic/datasets/maldroid-2020.html (accessed Jul. 01, 2023).

[22] K. V. Vinayaka and C. D. Jaidhar, “Android Malware Detection using Function Call Graph with Graph Convolutional Networks,” ICSCCC 2021 - Int. Conf. Secur. Cyber Comput. Commun., pp. 279–287, 2021, doi: 10.1109/ICSCCC51823.2021.9478141.

[23] D. Zou et al., “IntDroid: Android Malware Detection Based on API Intimacy Analysis,” ACM Trans. Softw. Eng. Methodol., vol. 30, no. 3, 2021, doi: 10.1145/3442588.

[24] B. J. Kang, S. Y. Yerima, K. McLaughlin, and S. Sezer, “N-opcode analysis for android malware classification and categorization,” 2016 Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2016, pp. 13–14, 2016, doi: 10.1109/CyberSecPODS.2016.7502343.

[25] D. Kim and A. Oh, “How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision,” pp. 1–25, 2022, [Online]. Available: http://arxiv.org/abs/2204.04879.

[26] I. Mironov, “Rényi Differential Privacy,” Proc. - IEEE Comput. Secur. Found. Symp., pp. 263–275, 2017, doi: 10.1109/CSF.2017.11.