Link Inference Attacks in Vertical Federated Graph Learning

Oualid Zari, Chuan Xu, Javier Parra-Arnau, Ayşe Ünsal, Melek Önen

Link Inference Attacks in Vertical Federated Graph Learning

Oualid Zari
EURECOM
Biot, France
oualid.zari@eurecom.fr

Chuan Xu
Univ. Côte d'Azur, Inria, CNRS, I3S
Biot, France
chuan.xu@inria.fr

Javier Parra-Arnau
Universitat Politècnica de Catalunya
Barcelona, Spain
javier.parra@upc.edu

Ayşe Ünsal
EURECOM
Biot, France
ayse.unsal@eurecom.fr

Melek Önen
EURECOM
Biot, France
melek.onen@eurecom.fr

***Abstract*—Vertical Federated Graph Learning (VFGL) is a novel privacy-preserving technology that enables entities to collaborate on training Machine Learning (ML) models without exchanging their raw data. In VFGL, some of the entities hold a graph dataset capturing sensitive user relations, as in the case of social networks. This collaborative effort aims to leverage diverse features from each entity about shared users to enhance predictive models or recommendation systems, while safeguarding data privacy in the process. Despite these advantages, recent studies have revealed a critical vulnerability that appears in intermediate data representations, which may inadvertently expose link information in the graph. This work proposes a novel Link Inference Attack (LIA) that exploits gradients as a new source of link information leakage. Assuming a semi-honest adversary, we demonstrate through extensive experiments on seven real-world datasets that our LIA outperforms state-of-the-art attacks, achieving over 10% higher Area Under the Curve (AUC) in some instances, thereby highlighting a significant risk of link information leakage through gradients. Our attack's effectiveness primarily stems from label information embedded in gradients, as evidenced by comparison with a label-only LIA. We analytically derive our Label-based LIA's accuracy using graph characteristics, assessing target graph vulnerability. To address these vulnerabilities, we evaluate two types of defenses: edge perturbation based on differential privacy and a novel label perturbation approach, demonstrating that our proposed label perturbation defense is more effective against all attack types across all datasets examined, offering a more favorable privacy-utility trade-off. Our comprehensive analysis shows why LIAs are effective and identifies potential defenses, highlighting the need for further research to improve the security of VFGL systems against link information leakage.**

1. Introduction

Recent research results in AI technology have shown that graph data are becoming increasingly popular as they improve the performance and accuracy of machine learning models. According to Gartner[1], graph technologies will be used in 80% of data and analytics innovations.

The use of graph data in AI systems also introduces new concerns in terms of privacy and security. Recent works have shown that, if not appropriately protected, Graph Neural Networks (GNN) are exposed to the so-called *link inference attacks* (LIA) [He et al.(2021)], which aim to discover relations among graph nodes by identifying or inferring whether or not there exist edges between them. Such attacks can reveal sensitive information about the relationships or interactions between parties represented by nodes in the graph. For example, in a social network graph, an LIA could potentially reveal private connections between individuals, such as friendships, professional relationships, or even more sensitive associations.

In this paper, we propose to study link inference attacks within the federated learning setting. Introduced by McMahan et al.[McMahan et al.(2017)], federated learning allows multiple parties to collaboratively train machine learning models while keeping their data on local premises, thereby ensuring data privacy. To integrate the use of graph data with federated learning, federated graph learning can be adapted in two settings: the horizontal federated setting, where multiple parties collaborate to train a global model using their local graph datasets that share similar feature spaces but differ in samples, and the vertical federated graph learning (VFGL) setting, where each collaborating party holds different features of the same samples. For example, one party may possess a graph dataset, while another may have features about the samples without any associated graph topology. In VFGL, the parties utilize their local datasets, which may include graphs, to train local models that generate intermediate or latent representations of their data that are then sent to the server. The server combines these intermediate representations from the participating parties along with its own training labels to train its model.

1. Gartner: Top 10 Data and Analytics Trends for 2021

To illustrate a practical application of VFGL, consider a collaboration between a social network company and another company that shares the same user base. The social network holds a graph of user connections, while the other company holds user feature data. By using VFGL, they can collaboratively improve an ad recommendation system. The social network company provides the graph data, which captures user relationships, and the other company supplies additional user features. Through VFGL, they can leverage both graph structures and feature data without sharing raw data, thus preserving user privacy while enhancing the recommendation system's effectiveness.

This paper focuses on edge/link inference attacks in the VFGL setting. In this architecture, clients participating in the training of the model but not holding the actual graph data can become potential adversaries and infer some information about the edges/links in the graph from the available training information. This information ranges from the gradients that were shared among parties during each training epoch to the model output when queried. Such a distinction arises from the party that can perform these attacks, namely the querier, any participating client, or the server itself. While some of the attacks relying on the knowledge of classification output or intermediate results were already proposed and evaluated as in [Qiu et al.(2022)], we have spotted that the knowledge of gradients or training labels were not studied yet. We therefore propose to study the impact of such attacks and compare them with existing ones and observe that the knowledge of gradients results in a substantial privacy leakage of links in the graph. Interestingly, we identify that the reason why Gradient-based link inference attacks are powerful is mainly due to the correlation of the gradients with the labels of training nodes. We indeed observe that the impact of the knowledge of labels is sufficient to infer graph edges. We also explore how the different properties of the graph such as its density or the class diversity help improve Label-based LIA. For example, our analysis shows that when labels are uniformly distributed, LIAs reach the maximum accuracy. To address these vulnerabilities, we evaluate two types of defenses: edge-level perturbation using Lapgraph and a novel label perturbation approach. Our analysis reveals that Lapgraph is largely ineffective against the introduced attacks, while our proposed label perturbation defense demonstrates effectiveness against all attack types across all datasets examined, offering a more favorable privacy-utility trade-off at lower privacy budgets.

To summarize, this paper makes the following contributions to the field of Vertical Federated Graph Learning and privacy in graph data:

1) We construct a new LIA that exploits the gradients information during training.
2) We study and evaluate our attack using seven real-world datasets. In some datasets, our results demonstrate that our proposed Gradient-based LIA outperforms other forms of LIAs where the adversary is even stronger and, for example, knows the model's output predictions.
3) We identify that the successful performance of our proposed Gradient-based LIA is mainly due to the node label information embedded in the gradients. This facilitates the attack against certain graphs with specific characteristics.
4) We mount a new Label-based LIA to show its strong connection to our primarily proposed Gradient-based LIA. We also provide a theoretical study on how the success of label-based LIA is influenced by some graph properties. These include its homophily, density, and class diversity.
5) We evaluate two defense mechanisms against our proposed attacks: edge-level perturbation (Lapgraph) and a novel label perturbation approach. We demonstrate that our label perturbation defense is more effective in mitigating the attacks while maintaining a better privacy-utility trade-off compared to edge-level perturbation.

2. Background

2.1. VFL system

TABLE 1: Table of Notations

Notation	Description
$\mathcal{P}_\mathcal{G}$	Party owning the graph dataset
$\mathcal{P}_\mathcal{A}$	Party holding the separate feature set
$\mathcal{P}_\mathcal{Y}$	Party owning the training labels
$\mathcal{G}$	Target graph owned by $\mathcal{P}_\mathcal{G}$
$X_\mathcal{G}$	Features owned by party $\mathcal{P}_\mathcal{G}$
$H_\mathcal{G}$	Intermediate representation of $X_\mathcal{G}$
$G_\mathcal{G}$	The gradients sent to party $P_\mathcal{G}$
$X_\mathcal{A}$	Features owned by party $\mathcal{P}_\mathcal{A}$
$H_\mathcal{A}$	Intermediate representation of $X_\mathcal{A}$
P	Output prediction computed by $\mathcal{P}_\mathcal{Y}$
$G_\mathcal{A}$	The gradients sent to party $P_\mathcal{A}$
$\mathcal{Y}$	Training labels owned by party $\mathcal{P}_\mathcal{Y}$

In this study, we investigate a two-party VFL setting involving parties $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$, along with an active party $\mathcal{P}_\mathcal{Y}$. $\mathcal{P}_\mathcal{G}$ owns a graph dataset denoted as $\mathcal{D}_\mathcal{G} = (\mathcal{G}, X_\mathcal{G})$, while $\mathcal{P}_\mathcal{A}$ holds a separate feature set denoted as $X_\mathcal{A}$. The parties share a user space of N samples, implying that the graph $\mathcal{G}$ contains N nodes, each representing a user. Within this user space, $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$ manage user features of dimensions d_G and d_A respectively. They collaborate with $\mathcal{P}_\mathcal{Y}$, the party owning the classification labels $\mathcal{Y}$, to perform a supervised learning task.

Specifically, $\mathcal{P}_\mathcal{G}$ employs a Graph Neural Network (GNN) to transform $X_\mathcal{G}$ into its intermediate representation $H_\mathcal{G}$, while $\mathcal{P}_\mathcal{A}$ uses a deep neural network to transform $X_\mathcal{A}$ into its intermediate representation $H_\mathcal{A}$. $\mathcal{P}_\mathcal{Y}$ gathers these intermediate representations and trains a Deep Neural Network (DNN) to compute the output prediction P for classification.

The process of training involves computing the loss function $\mathcal{L}$, deriving the gradients with respect to the model

parameters, and then updating these parameters. The gradients are computed according to the following rule:

$$\nabla_{\theta_k}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_i \frac{\partial \mathcal{L}}{\partial H_{i,k}} \frac{\partial H_{i,k}}{\partial \theta_k} \tag{1}$$

where θ_k represents the model parameters, $H_{i,k}$ is the latent representation of the i^{th} sample computed by party $\mathcal{P}_k$, and $\frac{\partial \mathcal{L}}{\partial H_{i,k}}$ is the gradient of the loss function $\mathcal{L}$ with respect to $H_{i,k}$ for $k \in \{\mathcal{G}, \mathcal{A}\}$. The details of the VFL training protocol are outlined in Algorithm 4 in Appendix A.

2.2. Graph neural networks

Graph Neural Networks (GNNs) have emerged as a prominent machine learning architecture for leveraging graph data structures in predictive modeling. GNNs excel at modeling complex relationships between entities by utilizing the features of these entities and their connections within a graph. They have found applications across various domains, including social networks where users and their relationships are modeled as nodes and edges, respectively, to enhance recommendation systems [Sharma et al.(2022)]. GNNs demonstrate superior performance in a wide array of classification tasks, such as node classification, link prediction, and graph classification. This paper focuses exclusively on the node classification task, aiming to predict the classes of individual nodes based on their features and connections in the graph. Specifically, in a graph $G = (V, E)$, where V represents nodes and E denotes edges, the primary objective of a GNN is to learn the mapping from the nodes' feature space to an embedding space through the aggregation of neighboring features using a neural network. These embeddings are then utilized for downstream tasks such as node classification. In our VFL scenario, the node embeddings, referred to as intermediate representations, are transmitted to the server for further processing to generate final output predictions. Various GNN architectures exist, differing mainly in their aggregation rules.

In this paper, we primarily focus on the Graph Convolutional Network (GCN) architecture. Later, we demonstrate that our attacks can also be executed on other types of GNNs, including GraphSAGE and GAT. Therefore, we first introduce these architectures.

Graph Convolutional Networks (GCN). GCNs are a pivotal architecture in the field of GNNs, lauded for their ability to perform node-level prediction and facilitate learning on graph-structured data [Kipf and Welling(2017)]. GCNs adapt the convolutional operation from image processing to the domain of graphs, allowing the model to harness the graph's structural information as well as node features. The central operation within a GCN is the aggregation of neighboring node embeddings, which enables the propagation of informative signals across the graph's edges. The update rule for node embeddings at each layer in a GCN is detailed by the following formula:

$$H^{(k+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k)}W^{(k)}\right), \tag{2}$$

where $H^{(.)}$ represents the embeddings of the nodes at the corresponding layer with $H^{(0)}$ being the node features. $\tilde{A} = A + I$ denotes the adjacency matrix A of the graph with self-loops added via the identity matrix I; $\tilde{D}$ is the diagonal degree matrix corresponding to $\tilde{A}$; $W^{(k)}$ is the trainable weight matrix of the k-th layer, and σ is a non-linear activation function (e.g., ReLU). This recursive relation allows the GCN to effectively learn representations that integrate neighborhood embeddings, capturing both local structure and global topology of the graph.

GraphSAGE. GraphSAGE stands for Graph Sample and AggregatE, and addresses the scalability issue of GCN to large graphs by aggregating features from a sampled subset of a node's neighbors [Hamilton et al.(2017)]. The feature aggregation in GraphSAGE is formulated as follows:

$$h_u^{(k+1)} = \sigma\left(W^{(k)} \cdot \text{CONCAT}\left(h_u^{(k)}, \underset{v\in N(u)}{\text{AGGREGATE}}\left(h_v^{(k)}\right)\right)\right).$$

In this formula, $h_u^{(k)}$ represents the feature vector of node u at the k-th layer, $N(u)$ denotes the sampled neighbors of u, and $W^{(k)}$ is the trainable weight matrix at the k-th layer. The AGGREGATE function can be a mean, sum, or max operation, depending on the specific implementation of GraphSAGE. The CONCAT operation combines the node's current features with aggregated neighbor features before applying the non-linear activation function σ. This approach controls the dimensionality of the embeddings through the dimensions of the weight matrices $W^{(k)}$, ensuring consistent embedding sizes across layers.

Graph Attention Network (GAT). GAT, introduced in [Velickovic et al.(2017)], leverages a similar message-passing approach as GCN in Eq.(2) but with a distinct aggregation function. The aggregation function in GAT employs an attention mechanism, allowing the GAT to assign edge weights dynamically based on node features, as opposed to the static, degree-based weighting in GCN. Attention coefficients a_{ij}, derived from node features, prioritize information from certain neighbors during aggregation. The GAT's aggregation function is defined by:

$$h_u^{k'} = \sigma\left(\sum_{v\in N_u} a_{uv} W^k h_v^{k-1}\right),$$

where $h_u^{k'}$ is node u's embedding at the k-th layer, N_u is its set of neighbors, a_{uv} represents the attention coefficients, W^k is the weight matrix at the k-th layer, and σ is an activation function such as ReLU.

3. Related Work

3.1. Link stealing attacks

Link stealing attacks were first introduced in [He et al.(2021)], demonstrating a correlation between the output predictions of nodes and the structure of the graph's links, as a source of edge information leakage. Additionally, [He et al.(2021)] explored the use of node

feature similarities to infer the graph's connections. In subsequent work, [Duddu et al.(2020)] showed that it is possible to reconstruct graph edges by analyzing predictions based on node embeddings, which are trained to capture the graph structure. Another work [Chen et al.(2024)] employs a graph autoencoder method, where the adversary is a server that uses node embeddings to predict links, assuming partial knowledge of the target graph to train the graph autoencoder. This stronger assumption makes the attack more powerful but less practical. In contrast, our approach does not require prior knowledge of the target graph, offering more realistic and broadly applicable assumptions for real-world VFGL settings.

Recent literature has explored both passive and active link stealing attacks. Active attacks, such as altering node features [Wu et al.(2021)] or introducing new nodes [Zari et al.(2023)], can potentially lead to more significant edge privacy leakage but require stronger assumptions about adversary capabilities. Our work, like other passive attacks, focuses on more realistic scenarios adhering to standard federated learning protocols. We exploit information leakage from model gradients and labels without requiring additional adversarial capabilities or partial knowledge of the target graph, making our approach more applicable in real-world VFGL settings. In the VFL setting, the first LIA was introduced in [Qiu et al.(2022)]. This research indicated that edge information could be revealed through the intermediate representations and output predictions of nodes. Multiple attack strategies were developed based on these leaks, depending on the adversary's knowledge of the target graph. However, the study did not address potential leakages from gradients or training labels of the nodes.

3.2. Privacy attacks in federated learning

A significant number of attacks have been developed, with a predominant focus on the horizontal federated learning (HFL) setting, compared to the VFL setting. Notably, in the case of VFL, one of the first developed attacks is the label-inference attack, wherein adversaries leverage gradients to infer the labels of nodes [Fu et al.(2022)], [Li et al.(2021)], [Zou et al.(2022)], [Sun et al.(2022)]. Beside label inference attacks to VFL, feature inference attacks have also been extensively studied in [Luo et al.(2020)], [Ye et al.(2022)]. In [Ye et al.(2022)], the authors showed that it is possible to reconstruct binary features of participating clients based on the intermediate representations of the samples, if the features dimension is not exceedingly large. In [Luo et al.(2020)], the authors demonstrated the feasibility of the reconstruction of features based on model predictions.

In contrast, the HFL setting has seen extensive study of gradient-based attacks, largely because gradients often represent the sole form of information exchange between participating parties. For instance, it has been shown that gradients can reveal the membership of training data, as highlighted in [Nasr et al.(2019)]. Furthermore, gradient-inversion attacks aim to reconstruct inputs from gradients, an attack that has been effectively mounted to computer vision domain [Zhu et al.(2019)]. Another study in [Melis et al.(2018)], which reveals that gradients can inadvertently disclose attributes unrelated to the primary machine learning task. Specifically, the authors showed that gradients could expose sensitive attributes, such as gender or race.

In this paper, we explore edge privacy leakage in gradients through the introduction of a new LIA. We aim to deepen the understanding of edge privacy vulnerabilities within federated learning environments.

4. Link Inference attacks

In this section, we define five different adversary models which mainly differ with respect to their knowledge whereas the goal always remains the same, that is, discovering edges in a graph. In all these models, we consider an honest but curious adversary who adheres to the training protocol without altering any of the communicated messages or the data. We assume a minimal setting with three parties: the server (the active party), who owns the training labels; the client, who has the graph with some features; and another client, who has information about other features. The victim is the client who owns the graph dataset. The other client and the active server can each be considered a potential adversary, who tries to infer graph edges with the information that is available to them.

4.1. Environment

We consider a VFL setting with two clients and one server. As illustrated in figure1, the two clients $\mathcal{P}_{\mathcal{G}}$ and $\mathcal{P}_{\mathcal{A}}$ contribute to the trained model and send the intermediate model representations to the server $\mathcal{P}_{\mathcal{Y}}$ who further computes and sends the gradients. $\mathcal{P}_{\mathcal{G}}$ is the party who holds the graph and is considered as the victim party whereas $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{Y}}$ can launch different LIAs which we study in the next section.

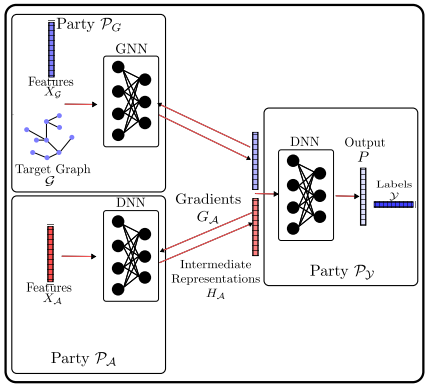


Figure 1: Schematic representation of a VFL setting with two clients and one server

4.2. Adversary's knowledge

Model gradients: The adversary, a participating client, receives gradients $G_{\mathcal{A}}$ for the cut layer of their model. This layer is the output of the adversary's model, interfacing with layers owned by other parties. The adversary obtains the gradient of each node in the batch at every training step.
Intermediate Representation: A participating client knows its intermediate representations $H_{\mathcal{A}}$ of its own features about the nodes.
Features: The adversary, a participating client, uses their features $X_{\mathcal{A}}$ to construct the attack. Features can correlate to graph edges: nodes are more likely to share similar features if they are neighbors in the target graph.
Training labels: The server, as an adversary, possesses the training labels of the graph nodes.
Prediction output: The server, as an adversary, has access to the final output prediction layer and uses the output prediction P scores to conduct the attack.

Although the Gradient-based and Intermediate representation-based LIAs are initially designed for participating clients, the server also has the necessary knowledge to perform these attacks. This is because the server dispatches gradients to the clients and receives the intermediate representations of the nodes from them. Table 2 summarizes the knowledge of different adversaries (i.e., server and client) in our VFGL setting.

TABLE 2: Adversary's knowledge in VFGL.

Party	Features	Labels	Gradients	Inter-Rep.	Pred. Output
Client	✓	✗	✓	✓	✗
Server	✗	✓	✓	✓	✓

While intermediate representations and prediction output were already exploited in [Qiu et al.(2022)] and features in [He et al.(2021)], gradient and label information have not been studied in the context of the VFGL setting. In the next section, we describe the actual attacks exploiting this information.

4.3. Link inference attacks - Description

In this section, we elaborate on the design of LIAs which differ with respect to the knowledge of the adversary. We first define our proposed LIAs; Gradient-based and Label-based LIAs, and then remind the other already existing attacks that we consider as baseline attacks.

4.3.1. Gradient-based LIA. In this attack, the adversary, acting as an FL client, exploits the received gradients $G_{\mathcal{A}}$ of node samples to infer links in the target graph $\mathcal{G}$. The adversary computes the cosine similarity between pairs of sample gradients and compares this score against a threshold value τ to determine if a link exists. Algorithm 1 details our methodology. We use cosine similarity due to its effectiveness in measuring neural network sample similarity [Charpiat et al.(2019), Corollary 2] and its common use in gradient-based FL attacks [Geiping et al.(2020)]. Empirically, it outperforms Euclidean and Chebyshev distances in our attack. For threshold selection, if the adversary can estimate the graph's density d, they could select the top $d\%$ most similar node pairs as connected, using the least similar among these as the decision threshold [Wu et al.(2021)]. Alternatively, the adversary might use a public partial graph or a similar graph in the same domain to estimate the threshold.

To evaluate our attack's effectiveness comprehensively, we primarily use the AUC metric, following previous works [Zari et al.(2023)], [Qiu et al.(2022)], [Salem et al.(2018)], [He et al.(2021)]. This metric assesses performance across various thresholds, providing a threshold-independent evaluation. For comparison with the label-based attack, which does not rely on a threshold, we compute the accuracy of our threshold-dependent attacks at the threshold yielding the highest F1 score. This approach balances precision and recall, offering detailed insights into each attack strategy's effectiveness while accounting for the practical challenges of threshold selection in real-world scenarios.

Algorithm 1 Gradient-based LIA

Require: Gradients $G_{\mathcal{A}}$ and threshold τ
Ensure: Inferred graph $\hat{\mathcal{G}}$
1: Adversary $\mathcal{A}$ receives gradients $G_{\mathcal{A}}$ from the server.
2: $\mathcal{A}$ initializes an empty graph $\hat{\mathcal{G}}$.
3: **for** each pair of samples (i, j) **do**
4: $S \leftarrow$ Cosine similarity$(G_{\mathcal{A}}^{(i)}, G_{\mathcal{A}}^{(j)})$.
5: **if** $S > \tau$ **then**
6: $\mathcal{P}_{\mathcal{A}}$ decides there is a link between (i,j) in graph $\hat{\mathcal{G}}$.

4.3.2. Label-based LIA. Unlike the Gradient-based LIA, where the adversary exploits the received gradients of the samples, in Label-based LIA, the adversary leverages the training labels $\mathcal{Y}$ of the samples to infer their links in the target graph $\mathcal{G}$. This method assumes that nodes with the same labels are more likely to be connected. More specifically, the adversary compares the labels of each pair of node samples to determine if they match. If the labels of two samples are identical ($\mathcal{Y}_i = \mathcal{Y}_j$), the adversary $\mathcal{A}$ infers that there is a link between these two samples in the target graph $\mathcal{G}$. This attack method is captured in Algorithm 2. It is important to note that the Label-based LIA requires different knowledge compared to the Gradient-based LIA, specifically access to training labels. While this assumption might seem strong, it is not unrealistic in certain scenarios. For instance, in our VFGL protocol, the server has access to the labels to perform the final model training. The Label-based LIA was introduced to provide deeper insights into the Gradient-based LIA's effectiveness, highlighting the complementary nature of the two methods in understanding link inference attacks in VFGL.

4.3.3. Baseline LIAs. Similar to the Gradient-based LIA, in Baseline LIAs, the adversary computes the cosine similarity between the observations of samples. These observations

Algorithm 2 Label-based LIA

Require: Labels $\mathcal{Y}$
Ensure: Inferred graph $\hat{\mathcal{G}}$
1: $\mathcal{A}$ initializes an empty graph $\hat{\mathcal{G}}$.
2: **for** each pair of samples (i, j) **do**
3: **if** $\mathcal{Y}_i == \mathcal{Y}_j$ **then**
4: $\mathcal{P}_{\mathcal{A}}$ decides there is a link between (i,j) in graph $\hat{\mathcal{G}}$.

could consist of either intermediate representations or outputs as already studied in [Qiu et al.(2022)], or features as investigated in [He et al.(2021)]. The aim is to use these similarities to infer links in the target graph $\mathcal{G}$. Similar to Gradient-based attacks, the effectiveness of such attacks is evaluated using the AUC metric across a range of potential threshold values. The details of the baseline attacks are outlined in Algorithm 3.

Algorithm 3 Baseline LIA

Require: Observations $\mathcal{O}$ based on attack type (intermediate representations, outputs, or features) and threshold τ
Ensure: Inferred graph $\hat{\mathcal{G}}$
1: Adversary $\mathcal{A}$ collects observations depending on the specific attack focus.
2: $\mathcal{A}$ initializes an empty graph $\hat{\mathcal{G}}$.
3: **for** each pair of samples (i, j) **do**
4: $S \leftarrow$ Cosine similarity$(\mathcal{O}_i, \mathcal{O}_j)$.
5: **if** $S > \tau$ **then**
6: $\mathcal{P}_{\mathcal{A}}$ decides there is a link between (i, j) in graph $\hat{\mathcal{G}}$.

4.4. Analytical results for LIAs

In this section, we first provide theoretical guarantees for the performance of Label-based LIA (Algorithm 2), given some common statistics of the graph dataset such as density d, homophily ratio h [Zhu et al.(2020)] and the number of classes C. Following this, we demonstrate a special case, where the Baseline LIA (Algorithm 3) with prediction outputs as observation (called Output-based LIA below) achieves the same link inferences as Label-based LIA. Note that these two attacks can only be executed by the server (Sect. 4.2) as a participating client lacks access to the required corresponding knowledge. Lastly, we illustrate a toy example showing the close relationship between the performance of Gradient-based LIA (Algorithm 1) and Label-based LIA (Algorithm 2).

4.4.1. Performance of Label-based LIA. Label-based LIA (Algorithm 2) infers a link when the two nodes share the same label. First, we can see that the success of Label-based LIA hinges on the correlation between links and the labels of nodes within a graph. In some real-world networks, it is often observed that adjacent nodes share similar labels (e.g., papers within the same research domain are likely to be cited together). This correlation can be quantified by the edge homophily ratio of a graph [Zhu et al.(2020)], which is the fraction of edges with nodes of the same label (see formal definition in 1). However, it is important to note that the homophily ratio only aids in measuring the success of inference on existing links. When inferring non-existing links, the diversity of class labels in a graph plays a crucial role in performance. If all nodes in the graph belong to one specific class, our LIA incorrectly predicts all non-existing links as existing ones. Thus, we introduce a metric $1 - \sum_{c=1}^{C} \alpha_c^2$, where α_c represents the proportion of nodes with label c, and C is the number of label classes, to measure this diversity. We observe that when the distribution of nodes across classes is uniform ($\alpha_c = \frac{1}{C}$), this metric reaches its maximum $\left(1 - \frac{1}{C}\right)$, signaling high class diversity. Conversely, when all the nodes are concentrated in one class t ($\alpha_t = 1$, $\alpha_i = 0$ for $i \neq t$), the metric reaches its minimum, zero. In balanced datasets where labels are uniformly distributed, a dataset with a higher number of classes is considered more diverse than one with fewer classes, even though both are balanced. This is because the diversity metric $1 - \frac{1}{C}$ directly reflects the impact of the number of classes on the perceived diversity; more classes mean a higher potential for diversity within the dataset, given that the distribution remains uniform across these classes. Due to this behavior, we refer to this metric as "class diversity" in our evaluations, highlighting its importance in accurately inferring non-existing links within a graph. With the knowledge of the aforementioned homophily ratio and the class diversity metric for label classes in the graph, we can derive the exact accuracy of Label-based LIA (Theorem 1) given the graph size and its density (i.e., the ratio of the number of edges in the graph to the maximum possible number of edges). The proofs are in Appendix B.

Definition 1 (Homophily Ratio [Zhu et al.(2020)])**.** *The homophily ratio of a graph quantifies the likelihood that adjacent nodes in the graph share the same label. Formally, the homophily ratio h can be expressed as:*

$$h = \frac{|\{(v, w) : (v, w) \in E \wedge \mathcal{Y}_v = \mathcal{Y}_w\}|}{|E|}, \tag{3}$$

where E denotes the set of edges in the graph, v and w represent nodes, and $\mathcal{Y}_u$ and $\mathcal{Y}_w$ are their respective class labels.

Theorem 1 (Accuracy of Label-based Link Inference Attack)**.** *For a graph G with N nodes, exhibiting homophily ratio h, density d, and C distinct label classes, let $\alpha_c = \frac{N_c}{N}$ represent the proportion of nodes with label c. The accuracy Acc of the Label-based link inference attack can be computed as:*

$$Acc = 2hd - d + \frac{N}{N-1}(1 - \sum_{c=1}^{C} \alpha_c^2) \tag{4}$$

When labels are uniformly distributed across nodes ($\alpha_c = \frac{1}{C}$), the accuracy of Label-based LIA reaches its maximum:

$$Acc \leq 2hd - d + \frac{N}{N-1}(1 - \frac{1}{C}) \tag{5}$$

4.4.2. Performance of Prediction Output-based LIA. When a model is well-trained, its prediction output highly correlates with the true label. Consequently, the performance of Output-based LIA is related to the performance of Label-based LIA. Here, we aim to establish a direct equivalence between these two attacks in the scenario where a GNN achieves perfect prediction for every data sample. Since Output-based LIA infers the links based on the cosine similarity of prediction outputs, a critical question arises: What is the minimum prediction score that models should assign to the true label in the output, ensuring that the cosine similarity of outputs for nodes with identical labels is distinguishable from those with different labels?

We demonstrate that in the case of binary classification, and where the loss function is log loss, such as cross-entropy, the condition for equivalence between the two types of attacks can be met under specific constraints on the training samples' loss. This equivalence is encapsulated in the following theorem:

Theorem 2. *For a Graph Neural Network (GNN) trained with cross-entropy loss for a binary classification task and with each training sample's loss l_i satisfying $l_i \leq -\log\left(\frac{3-\sqrt{3}}{2}\right)$, the output-based LIA with threshold $\tau = \frac{\sqrt{3}}{2}$ infers the same graph as Label-based LIA.*

The proofs are moved to Appendix C. In addition, we validate this theorem through empirical analysis, detailed in Appendix D.Briefly, we use the Cora dataset and a two-layer GCN, demonstrating that pairs of samples with loss below the threshold result in matching link predictions for both attack methods.

4.4.3. Performance of Gradient-based LIA. From Figure 1, we can see that although a participating client lacks access to the prediction output P and the labels $\mathcal{Y}$, the gradient $G_{\mathcal{A}}$ encodes this information in its calculation. Previous research on centralized graph learning demonstrates that gradients can leak label information [Fu et al.(2022)], [Li et al.(2021)]. Here, we illustrate a toy example showing that the cosine similarity of the gradients considered in our Gradient-based LIA is heavily dependent on the labels rather than the prediction output. Consequently, the performance of our Gradient-based LIA should closely align with that of Label-based LIA, which will be further confirmed by experiments in Section 5.3.

The toy example is a binary classification task with log loss, where the DNN owned by the Party $\mathcal{P}_{\mathcal{Y}}$ is a fully connected layer with weights $\mathbf{A}$ and bias $\mathbf{b}$, followed by a sigmoid function σ. Let $H_{\mathcal{A}}$ and $H_{\mathcal{G}}$ be the intermediate representations of party $\mathcal{P}_{\mathcal{A}}$ and of party $\mathcal{P}_{\mathcal{G}}$ respectively. We denote the H^i the i-th row of matrix H and $\mathcal{Y}_i$ the label of sample i. For every sample i, the prediction output $\hat{\mathcal{Y}}_i$ is

$$\hat{\mathcal{Y}}_i = \sigma(\mathbf{A}[H^i_{\mathcal{G}} \;\; H^i_{\mathcal{A}}] + \mathbf{b}).$$

We have the log loss $\mathcal{L}$ which is

$$\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}_i = -\sum_{i=1}^{N} \mathcal{Y}_i \log \hat{\mathcal{Y}}_i + (1-\mathcal{Y}_i)\log(1-\hat{\mathcal{Y}}_i).$$

The gradient of sample i is

$$G^i_{\mathcal{A}} = \frac{\partial L}{\partial H^i_{\mathcal{A}}} = \frac{\partial L_i}{\partial H^i_{\mathcal{A}}} = (\hat{\mathcal{Y}}_i - \mathcal{Y}_i)\mathbf{A}', \tag{6}$$

where $\mathbf{A}'$ are the weights in $\mathbf{A}$ that apply to $H^i_{\mathcal{A}}$.

According to Eq. 6, we can see that since $\forall i, 0 \leq \hat{\mathcal{Y}}_i \leq 1$, regardless of the prediction outputs, if two samples i and j are of different labels, the cosine similarity of $G^i_{\mathcal{A}}$ and $G^j_{\mathcal{A}}$ is negative. Conversely, if the samples belong to the same label, the cosine similarity is positive. In other words, even if the prediction outputs $\hat{\mathcal{Y}}_i = \hat{\mathcal{Y}}_j$, if these two samples are of different labels, our Gradient-based LIA will infer no link between them. Therefore, our Gradient-based LIA should be highly related to the Label-based LIA, which is less dependent on the gradient dimension and the features dimension. This contrasts with the other two possible attacks for the participating client, namely Baseline LIA with observations of intermediate representations and features, whose performance are inherently related to the observation dimension.

5. Evaluation

5.1. Overview

First, we outline our experimental setup, detailing datasets, GNN architectures, training protocols, and performance metrics in Section 5.2). We then present our experimental results (Section 5.3), highlighting the effectiveness of our proposed attacks compared to baselines and confirming the analytical studies from Section 4.4. Our main ablation studies (Section 5.4) focus on the gradient-based LIA's effectiveness over training time and the impact of the adversary-controlled feature ratio, comparing it with Intermediate representation-based and Feature-based attacks. Additional ablation studies are provided in Appendix E, which include analyses on the impact of GNN architecture, the number of parties, and model complexity on gradient-based LIA performance, offering further insights into our attack's robustness under various conditions.

5.2. Experimental setup

5.2.1. GNN Model architecture and learning setting. Our study follows the model architecture established in the baseline [Qiu et al.(2022)]. The server's model, also called the top model, is designed as a DNN featuring two fully connected layers activated by ReLU functions. To determine the vulnerability of the target/victim party's GNN's architecture to the attack, we utilize GCN, GraphSAGE, and GAT architectures.The GNNs' hop count are set to 2. The GNN implementations are derived from a publicly

TABLE 3: Datasets statistics.

Dataset	Nodes	Edges	Features	Classes	Density(%)
Photo	7650	119081	745	8	0.41
Cora	2708	5278	1433	7	0.14
Computer	13752	245861	767	10	0.26
Citeseer	3327	4552	3703	6	0.08
Twitch-DE	9498	157887	128	2	0.35
Twitch-EN	7126	38887	128	2	0.15
Twitch-FR	6551	115941	128	2	0.54

accessible code[2]. Similarly, the model of the other client, who can be a potential adversary, comprises a DNN with two fully connected layers employing ReLU activation functions. The bottom models, i.e., the two FL clients' models, are set to encode input features into a 16-dimensional latent space as a standard configuration, where the first layer maps first into half the input dimension. Note that, by default, the adversary controls 50% of the features.

We follow the same training protocol in baseline [Qiu et al.(2022)], where the VFGL models are trained over 300 epochs, by using a learning rate of 0.001 and by setting the regularization parameters to 0.001. The loss function of choice is cross-entropy, and model parameters are updated using the Adam optimizer. All experiments are conducted under identical settings but with different random seeds, repeated five times to calculate and report the average and standard deviation values of the performance metrics.

5.2.2. Datasets. We utilize seven public datasets for our analysis as Cora [Kipf and Welling(2017)], Citeseer [Kipf and Welling(2017)], Amazon Computers (Computer) [Shchur et al.(2018)], Amazon Photos (Photo) [Shchur et al.(2018)], and Twitch-(FR, DE, and EN) [Rozemberczki and Sarkar(2021)] datasets. These are widely recognized as benchmark datasets for evaluating the performance and privacy aspects of GNNs [Qiu et al.(2022)], [Zari et al.(2023)], [He et al.(2021)], [Wu et al.(2021)].

The Citeseer and Cora datasets are citation networks, where nodes and edges respectively correspond to publications and citations among these publications. Node features consist of the declared keywords in the publications, and class labels represent the research fields of the corresponding publications. The Amazon Computers and Amazon Photos datasets are parts of the Amazon co-purchase graph, where nodes and edges respectively represent products and the actual two products are frequently bought together. Node features are bag-of-words representations of the corresponding product reviews, and class labels categorize the product types. The Twitch datasets are social network datasets that depict the followership connections between users on the Twitch streaming platform. Node features include users' preferred games, location, and streaming habits, while class labels indicate whether a streamer uses explicit language.

2. https://pytorch-geometric.readthedocs.io/

5.2.3. Evaluation Metrics. AUC. The performance of the attack is evaluated by using the area under the ROC curve (AUC). The AUC provides a comprehensive measure of the attack's performance across various decision thresholds, highlighting its threshold-independent nature. An AUC value of 0.5 means that the attack performance is equivalent to random guessing and hence the adversary has no power, as opposed to the case where the AUC approaches 1, the attack becomes successful in inferring graph links.

Accuracy. Accuracy is used to measure the performance of label-based LIA, which uniquely does not require a decision threshold for link prediction unlike the other attacks. For a comparative analysis of label-based LIA against our introduced threshold-dependent attacks, accuracy is assessed at the threshold with the highest F1 score. This approach ensures a balance between precision and recall, providing a detailed insight into the effectiveness of each attack strategy.

5.3. Performances of LIAs.

We first conduct all the attacks across all the datasets in the scenario where the Party $\mathcal{P}_{\mathcal{A}}$ has owned 50% of the features, i.e., the size of $X_{\mathcal{A}}$ is equivalent to the size of $X_{\mathcal{G}}$. We also report the maximum accuracy achieved over the training epochs for time-dependent attacks, including Gradient-based, Intermediate Representation-based, and Output-based LIAs. The accuracy of the LIAs are shown in Table 4. Note that participating clients can only conduct Gradient-based, Intermediate-Representation-based and Features-based LIA as mentioned in Section 4.2.

Label-based LIA. First, we can observe from Table 4 that the performance of Label-based LIA is positively correlated with the homophily ratio h and the class diversity $1 - \sum_{c=1}^{C} \alpha_c^2$, which confirms the analytical results in Sect. 4.4.1. In fact, due to the low density of the datasets (Table 3), the performance of the attack is primarily influenced by the class diversity metric (the third term in Eq. 4). Note that variability in results observed across the 5 trials stems from the random partitioning of nodes into training and testing sets (Sect. 5.2.2). Here we report the average metrics for homophily ratio and the class diversity.

Prediction Output based LIA. For the datasets Photo, Cora, Computer, and Citeseer, the output-based LIA demonstrates similar performance compared to the Label-based LIA where the difference is within 2 percentage points (p.p.). The reason behind this, as explained in Section 4.4.2, is that since the model, being well-trained, yields prediction outputs that strongly align with the true labels. For Twitch datasets, since the model is less pertinent (as also observed in [Wu et al.(2021)], [Zari et al.(2023)]), the performance displays a weaker correlation with the Label-based attack. Moreover, the links within Twitch datasets (followership connections) are primarily associated with features (such as users' streaming habits) rather than labels (indicating whether a streamer uses explicit languages). Therefore, the

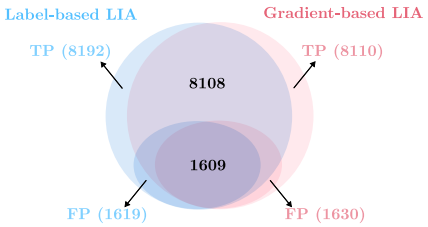


Figure 2: Overlap of predictions between gradient-based and label-based LIAs in Photo dataset.

prediction output, which inherently relies on these features, surpasses the performance of the Label-based attack.

Gradient-based LIA VS. Label-based. In Sect. 4.4.3, we presented a simplified example where the DNN architecture of Party $\mathcal{P}_{\mathcal{Y}}$ consists of only one fully connected layer with one neuron as output. This example was used to illustrate that the effectiveness of Gradient-based LIA is strongly influenced by the inherent embedding of label information into the gradients, resulting in predictions that closely resemble those produced by Label-based LIA. Our experimental results, as depicted in Table 4, confirm that this observation still remains applicable in a more intricate DNN scenario, characterized by two fully-connected layers, each followed by a ReLU activation function, when the dataset is with high homophily ratio (>0.7) and class diversity (>0.7). The accuracy difference between Gradient-based and Label-based attacks is within 1.7 p.p.. To further investigate this observation, we examine the predictions of both attacks, focusing on True Positives (TP) and False Positives (FP). Specifically, we conduct a comparative analysis on a subgraph of the Amazon-Photo dataset, comprising 10000 positive pairs (unlinked pairs) and 10000 negative pairs (linked pairs). The experiment (Figure 2) reveals a significant overlap in link predictions between the two attacks. In the case of label-based LIA, out of 9811 link predictions, 8192 are correctly identified (TP), and 1619 pairs of nodes are incorrectly identified as linked (FP). Similarly, gradient-based LIA results in 9740 link predictions, with 8110 correctly identified (TP) and 1630 pairs incorrectly identified as linked (FP). The TPs of both attacks overlap by 8108 pairs, and the FPs overlap by 1609 pairs. This overlap suggests that the success of gradient-based LIA is not coincidental but rather indicative of its reliance on label information for link prediction. Given the high overlap in predictions and equivalent performance of the two attacks, Label-based LIA (the attack performance of which can be analytically derived in Eq. 6) can serve as a proxy for estimating an adversary's potential performance using gradients to infer links in the target graph.

Gradient-based LIA VS. Prediction Output-based. For datasets with lower homophily ratios and class diversity, our gradient-based LIA performs similarly to the Prediction Output LIA (accuracy differences within 0.3 p.p.). While not significantly improving accuracy, our attack offers greater practicality. The output prediction-based attack requires server-level access to predictions, whereas our gradient-based attack adheres to strict VFL protocol without additional assumptions, as shown in Algorithm 4.

Gradient-based LIA VS. Intermediate-representation and feature-based. Recall that for a participating client adversary, the attack can be only executed based on the gradients $G_{\mathcal{A}}$, intermediate representations $H_{\mathcal{A}}$, and the features $X_{\mathcal{A}}$. In the table, we highlighted the best attack performance among these three attacks for every dataset. We can see that the Gradient-based LIA outperforms the other two attacks in 5 out of 7 datasets. Particularly for the Photo, Cora, and Computer datasets, the attack performance is enhanced by at least 10 p.p.. This improvement can be attributed to the fact that the links in these datasets are more label-related than feature-related (see Section 5.2.2 for more further details on datasets). Although Citeseer and Cora are both citation datasets, Citeseer possesses more than twice the number of features compared to Cora (Table 3). Therefore, the Feature-based LIA performance is close to the Gradient-based LIA performance, given that Citeseer exhibits a unique property where the links are both label-related and feature-related. For the other Twitch datasets, where the links are primarily associated with features, the Gradient-based LIA still demonstrates comparable performance. Additionally, as discussed in Section 4.4.3, one advantage of the gradient-based attack is its independence w.r.t. the number of features possessed by the adversary, while the other two attacks are significantly influenced by this factor. In our subsequent ablation study (Section 5.4.2), we will demonstrate that the Gradient-based attack yields even greater advantages when the feature ratio is decreased from 50% to 10%.
Overall, the Gradient-based LIA demonstrates comparable results in both label-related and feature-related scenarios (all ragne of h), while Intermediate-representation, Feature-based and Label-based attacks fail in one of these scenarios. It also outperforms Prediction output LIA by 0.75 p.p. in average even though Prediction output LIA requires a strong adversary. In real-world applications, the adversary may lack prior knowledge regarding whether the dataset is more label-related or feature-related. Therefore, the Gradient-based attack stands out as the optimal choice for the adversary.

5.4. Ablation study

5.4.1. Impact of training epochs on gradient-based LIA performance.. In this study, we aim to asses the impact of the training epochs on the performance of our gradient-based LIA. We conduct a parallel analysis on the Intermediate representation-based LIA baseline, noting that both attacks can be mounted on the client side by party $P_{\mathcal{A}}$ and are inherently time-dependent. We specifically investigate how the learning epochs affect the AUC of these two attack strategies, executing them at each training epoch. According to Figure 3, our gradient-based LIA outperforms in the initial epochs, though its efficiency decreases as training progresses. This pattern likely emerges because the gradients in

TABLE 4: Accuracy of link inference attacks across datasets using GCN architecture. Bold numbers indicate highest accuracy among client-side attacks (first three columns). Participating clients can only conduct these three attacks.

Datasets	h	Class diversity	Attack Methods				
			Gradients	Inter-Reps	Features	Labels	Prediction output
Photo	0.83	0.84	**82.12 ± 1.32**	67.61 ± 2.76	63.58 ± 0.42	83.76 ± 0.10	81.80 ± 0.68
Cora	0.81	0.82	**81.71 ± 0.21**	65.77 ± 1.19	71.34 ± 1.95	81.74 ± 0.15	80.14 ± 0.58
Computer	0.78	0.79	**78.23 ± 1.26**	66.46 ± 0.40	63.75 ± 0.86	79.35 ± 0.05	78.82 ± 0.70
Citeseer	0.74	0.82	**82.76 ± 0.38**	73.53 ± 2.58	82.65 ± 0.70	82.14 ± 0.02	79.64 ± 0.64
Twitch-DE	0.64	0.48	**58.76 ± 0.22**	55.35 ± 1.47	56.76 ± 0.47	48.02 ± 0.03	58.61 ± 0.01
Twitch-EN	0.60	0.50	53.28 ± 0.25	**54.63 ± 0.79**	54.37 ± 0.80	49.65 ± 0.10	53.51 ± 0.44
Twitch-FR	0.55	0.47	50.89 ± 0.66	50.16 ± 1.09	**56.04 ± 0.45**	46.68 ± 0.20	49.72 ± 0.51

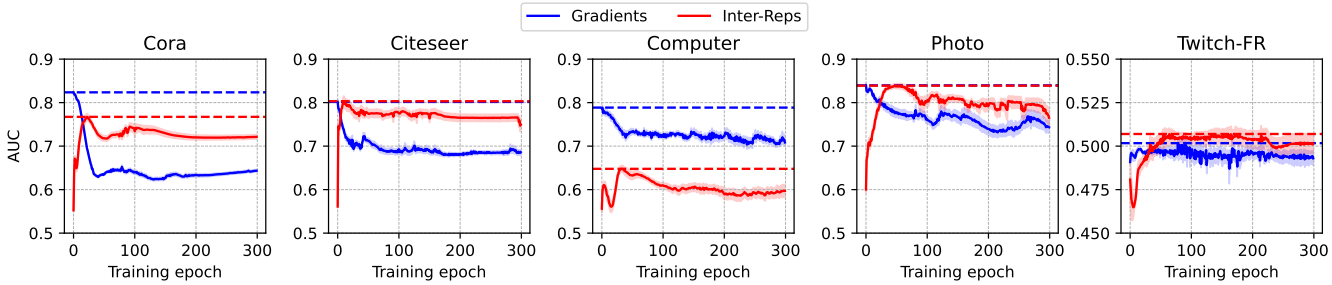


Figure 3: Evolution of the AUC for Gradient-based LIA (blue) and Intermediate-representation LIA (red) Over Time. The horizontal dashed lines indicate the maximum AUC achieved by the two attacks. Attacks were conducted at each training epoch, across five runs, with the mean and standard deviation of the AUCs reported.

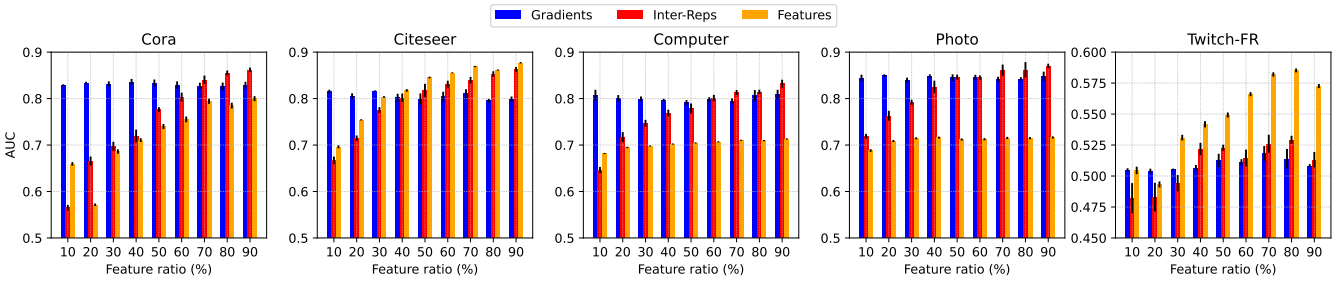


Figure 4: Comparison of AUC between Gradient-based LIA (blue), Intermediate Representations-based LIA (red), and Feature-based LIA (orange) across different feature ratios of the adversary. The maximum AUC achieved by both attacks during training is reported, alongside the mean and standard deviation of these AUCs across five runs.

the initial epochs are more informative, gradually becoming less distinguishable as the model nears convergence. In particular, as the model approaches convergence, the gradients of all nodes start to concentrate around zero, complicating the adversary's task of differentiating between connected and non-connected node pairs, as depicted in Figure 5.

On the other hand, the Intermediate representation-based LIA shows modest performance in the first training epochs, with its performance reaching its maximum after the early training epochs. This pattern is attributed to the fact that, during the early epochs, the intermediate representations do not capture the features of nodes due to the adversary's model lack of optimization. Thus, the Intermediate representation-based LIA reaches peak efficiency in the later stages of training. In contrast, our gradient-based LIA achieves its highest performance in the training's early epochs.

5.4.2. Impact of feature ratio owned by the adversary.. In this study, we examine the impact of the proportion of features owned by the adversary $P_{\mathcal{A}}$ on the performance of our gradient-based LIA, intermediate representation LIA, and the feature-based LIA. For this purpose, we analyze the success of the attack across varying ratios of adversary-owned features $X_{\mathcal{A}}$, which aids in comparing the attacks and their dependence on the number of features the adversary possesses. As illustrated in Figure 4, we report the peak AUC over time achieved by the attacks across a range of feature ownership ratios, adjusting from the default condition where the adversary controls 50% of the features, ranging from 10% to 90% of the entire set. As shown

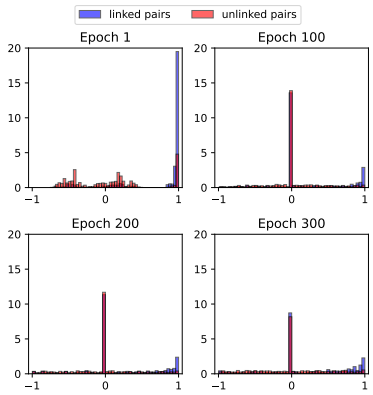


Figure 5: Distribution of the cosine similarities of gradients among linked and unlinked pairs at different training epochs.

in Figure 4, the gradient-based LIA outperforms the intermediate representation-based and feature-based LIAs when less than half of the features are owned by the adversary, except for the Twitch-FR dataset. This is mainly due to the low homophily and class diversity, while the features are more representative of the links in the graph. However, the intermediate representation-based LIA performs better at higher feature ratios. This phenomenon is attributed to the richness of intermediate representations when a larger number of features is available, leveraging the correlation between features and the connections within the target graph. This also accounts for the proportional increase in the intermediate representation-based LIA's AUC with the number of features the adversary controls. Unlike the gradient-based LIA, whose AUC remains relatively unaffected by changes in the number of adversary-owned features, this stability stems from the fact that gradients rely not only on the adversary's intermediate representations, which are impacted by the features owned, but also on the labels of the training nodes as discussed in section 4.4.3. The same phenomenon of feature size dependence on the performance of feature-based LIA is observed, except for the Photo and Computer datasets. We think this may be attributed to the sparsity of the features compared to other datasets. In the Photo and Computer datasets, the feature sparsity is relatively higher than in the other datasets as indicated in [Qiu et al.(2022)], which means that even 10% of the features is sufficient to reach the maximum performance in inferring the links of these datasets.

6. Defense

In this section, we evaluate potential defenses against the LIAs introduced earlier. We consider two types of defenses: edge perturbation and label perturbation. The edge perturbation aims to obscure the graph structure, while the label perturbation addresses the core issue of label leakage. For edge perturbation, we use Lapgraph [Wu et al.(2021)], a differential privacy (DP) mechanism that guarantees edge-level DP by adding noise to the graph structure. For label perturbation, we develop a novel approach using quadratic optimization to strategically obfuscate labels. In our vertical federated learning setting, the edge-level perturbation (Lapgraph) is implemented by the (honest) client, which owns the graph structure, while the label perturbation is applied by the (honest) server, which possesses the labels. In the following subsections, we will analyze how these distinct defenses impact each of the previously introduced attacks.

6.1. Lapgraph

Lapgraph [Wu et al.(2021)] is a defense mechanism that applies differential privacy at the edge level by perturbing the adjacency matrix of the graph. It adds Laplace noise to the adjacency matrix, controlled by a privacy parameter ε, followed by a binarization process. This effectively alters the graph structure by potentially adding fake edges and removing real ones. In our implementation, we vary ε from 1 to 10 to demonstrate the defense's effectiveness across different privacy levels. Figure 6 shows the results of Lapgraph defense on the Computer, Photo, Cora, and Citeseer datasets, along with a random guessing baseline for test accuracy of the model. From the results, we can see that the Lapgraph defense does not effectively mitigate the attacks' effectiveness across all datasets, particularly for our introduced gradient-based attack. The gradient-based attack remains highly resilient, showing minimal to no reduction in accuracy across different ε values. This resilience can be attributed to the fact that the gradient information, which is closely tied to the label information, remains largely unaffected by the graph structure perturbation.

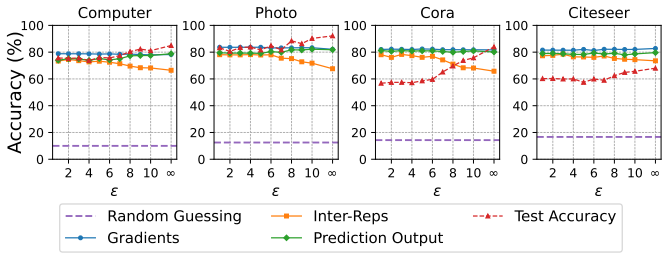


Figure 6: The performance of LIAs under Lapgraph defense. $\varepsilon = \infty$ represents no defense; lower ε values indicate stronger privacy protection.

The prediction output attack follows a similar pattern to the gradient-based attack, with only a marginal decrease in effectiveness across all datasets. On Cora at $\varepsilon = 6$, its accuracy is 80.96%, compared to the baseline of 80.14% without any defense. Interestingly, the intermediate representation attack exhibits an unexpected inverse behavior across all datasets, showing increased performance as ε decreases. For Cora at $\varepsilon = 6$, its accuracy rises to 76.89% from the baseline of 65.77%. We speculate that this counterintuitive result occurs because as the graph edges become noisier, the server model pays more attention to the adversary model than to the victim GNN model. This shift leads to the adversary model

becoming a stronger learner about the labels and features of the nodes, hence making its intermediate representation a better signal for the edges of the graph, thus increasing the attack's accuracy. This phenomenon aligns with observations in [Qiu et al.(2022)], where perturbation of the victim party's intermediate representations as a defense mechanism inadvertently resulted in strengthening the intermediate representation-based attack of the adversary model.

It is important to note that the label-based attack remains unaffected by this defense. This is because Lapgraph only perturbs the graph structure and does not alter the label information, which is the primary source of information for this attack. The same goes for features-based attack, as Lapgraph does not modify the node features.

While the defense shows minimal impact on the attacks' effectiveness, there is a significant decrease in test accuracy as ε decreases, indicating a substantial utility loss across all datasets. On Cora at $\varepsilon = 6$, the test accuracy drops to 59.66% from the baseline of 83.97%. This underscores the limitation of edge perturbation as a defense mechanism against our proposed attacks, which primarily exploit label information leakage rather than graph structural properties.

6.2. Label perturbation

Label perturbation is a defense mechanism that directly addresses the issue of label leakage by strategically obfuscating a portion of the labels. This approach operates with a budget B, representing the percentage of labels to be changed. We implement this defense by formulating a quadratic optimization problem to find the optimal class proportions that minimize the attack accuracy derived in our theorem 1, subject to the budget constraint. The label perturbation defense is implemented in two steps: label proportion optimization (Algorithm 5) and label redistribution (Algorithm 6). These algorithms are outlined in Appendix F.

Figure 6 shows the results of our label perturbation defense on the Computer, Photo, Cora, and Citeseer datasets, along with a random guessing baseline for test accuracy of the model. We observe that the label perturbation defense demonstrates significant effectiveness in mitigating various attacks across all datasets. As the budget increases from 0.05 to 0.90, we see a substantial decrease in the accuracy of all attacks, albeit at different rates.

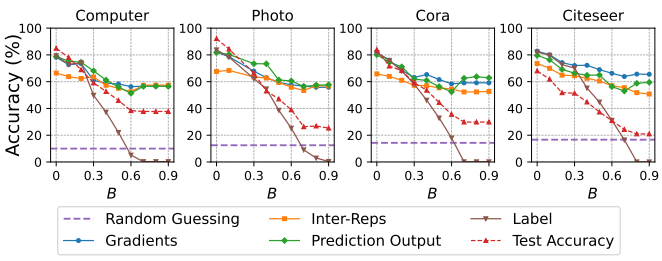


Figure 7: The performance of LIAs under label perturbation defense. $B = 0$ represents no defense; higher B values indicate stronger protection.

The label-based attack is the most affected across all datasets, with its accuracy on Cora dropping from 79.14% at a 0.05 budget to near-zero (0.14%) at a 0.90 budget, compared to the baseline of 81.74% without defense. This dramatic reduction is expected as the defense directly targets the label information that this attack relies on. The gradient-based attack also shows a significant decrease in accuracy across all datasets. On Cora, it drops from 79.03% at a 0.05 budget to 58.93% at a 0.90 budget, a substantial reduction from the baseline of 81.71%. This substantial reduction reflects the strong correlation between gradient information and labels. The prediction output attack and inter-representation attack show more resilience to the label perturbation defense across all datasets. On Cora, the prediction output attack's accuracy decreases from 77.41% to 64.18%, while the inter-representation attack's accuracy drops from 66.77% to 51.90% as the budget increases from 0.05 to 0.90. These are still significant reductions from their baselines of 80.14% and 65.77% respectively. Comparing these results to Lapgraph at points where the utility (test accuracy) is similar, we see that label perturbation defense tends to be more effective across all datasets. For instance, on Cora at a budget of 0.30, label perturbation achieves a test accuracy of 59.12%, which is comparable to Lapgraph's 59.66% at $\varepsilon = 6$. At these points, the gradient-based attack accuracy is 63.14% for label perturbation, compared to 82.24% for Lapgraph. Similarly, the inter-representation attack accuracy is 57.21% for label perturbation, but 76.89% for Lapgraph. The label perturbation defense does impact the system's utility across all datasets, with the test accuracy on Cora decreasing from 73.87% at a 0.05 budget to 29.79% at a 0.90 budget, compared to the baseline of 83.97%. However, its privacy-utility trade-off appears more favorable compared to Lapgraph, especially at lower privacy levels where the utility loss is less severe. In conclusion, these results suggest that addressing the fundamental issue of label leakage through targeted perturbation is generally more effective than graph structure alterations in defending against LIAs, particularly when considering the privacy-utility trade-off.

7. Conclusion

In this study, we have introduced gradient-based and label-based link inference attacks on Vertical Federated Learning (VFL). We have demonstrated that gradients in VFL can reveal link information, primarily due to the embedded label information within the gradients. To explore their relation, we have developed a label-based link inference attack. Our comprehensive evaluation has confirmed that both attack methods perform similarly, supporting our hypothesis. Our findings have also indicated that these attacks surpass alternative link inference attacks that rely on model predictions and intermediate representations. To counter these vulnerabilities, we proposed a label perturbation defense that shows promise in mitigating these attacks across all datasets. While effective, future work is needed to further improve the privacy-utility trade-off in VFGL systems.

Acknowledgments

Oualid Zari acknowledges support from the 3IA program. Chuan Xu's research received support from Groupe La Poste, sponsor of the Inria Foundation, within the FedMalin Inria Challenge framework, and was partly funded by the European Network of Excellence dAIEDGE under Grant Agreement Nr. 101120726. Javier Parra-Arnau acknowledges his "Ramón y Cajal" fellowship (ref. RYC2021-034256-I) funded by the Spanish Ministry of Science and Innovation and the European Union – "NextGeneration EU"/PRTR. This research was supported by Spanish Government projects: "DISCOVERY" - PID2023-148716OB-C32, "COMPROMISE" - PID2020-113795RB-C31, and "MOBILYTICS" - TED2021-129782B-I00, funded by MICIU/AEI/10.13039/501100011033, and the "NextGeneration EU/PRTR." Support also came from the Generalitat de Catalunya under AGAUR grant "2021 SGR 01413."

References

[Charpiat et al.(2019)] Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. 2019. Input Similarity from the Neural Network Perspective. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc.

[Chen et al.(2024)] Jinyin Chen, Minying Ma, Haonan Ma, Haibin Zheng, and Jian Zhang. 2024. An Empirical Evaluation of the Data Leakage in Federated Graph Learning. *IEEE Transactions on Network Science and Engineering* 11, 2 (2024), 1605–1618.

[Duddu et al.(2020)] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying Privacy Leakage in Graph Embedding. *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (2020).

[Fu et al.(2022)] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Junfeng Zhou, Alex X. Liu, and Ting Wang. 2022. Label Inference Attacks Against Vertical Federated Learning. In *USENIX Security Symposium*.

[Geiping et al.(2020)] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients - How easy is it to break privacy in federated learning?. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 16937–16947.

[Hamilton et al.(2017)] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Neural Information Processing Systems*.

[He et al.(2016)] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[He et al.(2021)] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *30th USENIX Security Symposium (USENIX Security 21)*. 2669–2686.

[Kipf and Welling(2017)] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[Li et al.(2021)] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2021. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504* (2021).

[Luo et al.(2020)] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2020. Feature Inference Attack on Model Predictions in Vertical Federated Learning. *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2020), 181–192.

[McMahan et al.(2017)] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282.

[Melis et al.(2018)] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2018. Exploiting Unintended Feature Leakage in Collaborative Learning. *2019 IEEE Symposium on Security and Privacy (SP)* (2018), 691–706.

[Nasr et al.(2019)] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 739–753.

[Qiu et al.(2022)] Pengyu Qiu, Xuhong Zhang, Shouling Ji, Tianyu Du, Yuwen Pu, Jun Zhou, and Ting Wang. 2022. Your Labels Are Selling You Out: Relation Leaks in Vertical Federated Learning. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–16.

[Rozemberczki and Sarkar(2021)] Benedek Rozemberczki and Rik Sarkar. 2021. Twitch Gamers: a Dataset for Evaluating Proximity Preserving and Structural Role-based Node Embeddings. *CoRR* abs/2101.03091 (2021). arXiv:2101.03091

[Salem et al.(2018)] A. Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. 2018. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. *ArXiv* abs/1806.01246 (2018).

[Sharma et al.(2022)] Kartik Sharma, Yeon-Chang Lee, Siva Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. 2022. A Survey of Graph Neural Networks for Social Recommender Systems. *ArXiv* abs/2212.04481 (2022).

[Shchur et al.(2018)] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *ArXiv* abs/1811.05868 (2018).

[Sun et al.(2022)] Jiankai Sun, Xin Yang, Yuanshun Yao, and Chong Wang. 2022. Label Leakage and Protection from Forward Embedding in Vertical Federated Learning. *ArXiv* abs/2203.01451 (2022).

[Velickovic et al.(2017)] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. 2017. Graph Attention Networks. *ArXiv* abs/1710.10903 (2017).

[Wu et al.(2021)] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2021. LINKTELLER: Recovering Private Edges from Graph Neural Networks via Influence Analysis. *2022 IEEE Symposium on Security and Privacy (SP)* (2021), 2005–2024.

[Ye et al.(2022)] Peng Ye, Zhifeng Jiang, Wei Wang, Bo Li, and Baochun Li. 2022. Feature Reconstruction Attacks and Countermeasures of DNN training in Vertical Federated Learning. *ArXiv* abs/2210.06771 (2022).

[Zari et al.(2023)] Oualid Zari, Javier Parra-Arnau, Ayşe Ünsal, and Melek Önen. 2023. Node Injection Link Stealing Attack. *ArXiv* abs/2307.13548 (2023).

[Zhu et al.(2020)] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 653, 12 pages.

[Zhu et al.(2019)] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).

[Zou et al.(2022)] Tianyuan Zou, Yang Liu, Yan Kang, Wenhan Liu, Yuanqin He, Zhi qian Yi, Qian Yang, and Ya-Qin Zhang. 2022. Defending Batch-Level Label Inference and Replacement Attacks in Vertical Federated Learning. *IEEE Transactions on Big Data* (2022).

Appendix A. VFL training protocol

Algorithm 4 Two-Party Vertical Federated Learning

Require: learning rates $\eta_{\mathcal{G}}$ and $\eta_{\mathcal{A}}$
Ensure: Trained model parameters $\theta_{\mathcal{G}}$, $\theta_{\mathcal{A}}$, ψ
Parties $\mathcal{P}_{\mathcal{G}}$, $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{Y}}$ initialize $\theta_{\mathcal{G}}$, $\theta_{\mathcal{A}}$, ψ.
for each training iteration $t = 1, 2, \ldots$ **do**
  In **parallel** do the following:
    Party $\mathcal{P}_{\mathcal{G}}$:
      Computes $H_{\mathcal{G}} = GNN(X_{\mathcal{G}}, \theta_{\mathcal{G}})$
      Sends $H_{\mathcal{G}}$ to party $\mathcal{P}_{\mathcal{Y}}$
    Party $\mathcal{P}_{\mathcal{A}}$:
      Computes $H_{\mathcal{A}} = DNN(X_{\mathcal{A}}, \theta_{\mathcal{A}})$
      Sends $H_{\mathcal{A}}$ to party $\mathcal{P}_{\mathcal{Y}}$
  End
  Party $\mathcal{P}_{\mathcal{Y}}$ computes the prediction output $P_{\mathcal{Y}} = DNN((X_{\mathcal{G}}, X_{\mathcal{A}}), \psi)$
  $\mathcal{P}_{\mathcal{Y}}$ updates $\psi^{t+1} = \psi^t - \eta_{\mathcal{G}} \frac{\partial \mathcal{L}}{\partial \psi}$
  $\mathcal{P}_{\mathcal{Y}}$ computes and sends the gradients $G_{\mathcal{G}} = \frac{\partial \mathcal{L}}{\partial H_{\mathcal{G}}}$ and $G_{\mathcal{A}} = \frac{\partial \mathcal{L}}{\partial H_{\mathcal{A}}}$ to $\mathcal{P}_{\mathcal{G}}$ and $\mathcal{P}_{\mathcal{A}}$, respectively.
  In **parallel** do the following:
    Party $\mathcal{P}_{\mathcal{G}}$:
      Computes $\nabla_{\theta_{\mathcal{G}}} \mathcal{L}$ with Equation 1
      Updates $\theta_{\mathcal{G}}^{t+1} = \theta_{\mathcal{G}}^t - \eta_{\mathcal{G}} \nabla_{\theta_{\mathcal{G}}} \mathcal{L}$
    Party $\mathcal{P}_{\mathcal{A}}$:
      Computes $\nabla_{\theta_{\mathcal{A}}} \mathcal{L}$ with Equation 1
      Updates $\theta_{\mathcal{A}}^{t+1} = \theta_{\mathcal{A}}^t - \eta_{\mathcal{A}} \nabla_{\theta_{\mathcal{A}}} \mathcal{L}$
  End

Appendix B. Proof of theorem 1

B.1. Notations

Consider a graph G consisting of N nodes and a set of edges E, characterized by a homophily ratio h, density d, and C distinct label classes. Let $\alpha_c = \frac{N_c}{N}$ denote the fraction of nodes that have the label c. We use the binary indicator e_{ij} to specify whether nodes i and j are connected ($e_{ij} = 1$) or not ($e_{ij} = 0$). The label of node i is represented by y_i. The density d of the graph G is defined by the equation:

$$d = \frac{|E|}{\frac{N(N-1)}{2}}, \tag{7}$$

where $|E|$ is the number of edges in the set E.

B.2. Accuracy of the label-based LIA

The accuracy Acc of the label-based LIA can be expressed in terms of the confusion matrix elements:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN}. \tag{8}$$

The different terms in Acc are defined in below:
True Positives (TP): The number of correctly predicted edges that exist between nodes with the same label; i.e. nodes that are connected and have the same labels, given by:

$$TP = |\{(i,j) \mid e_{ij} = 1 \wedge y_i = y_j\}|. \tag{9}$$

True Negatives (TN): The number of correctly predicted non-edges between nodes with different labels; nodes that are not connected/neighbors and do not share the same label, computed as:

$$TN = |\{(i,j) \mid e_{ij} = 0 \wedge y_i \neq y_j\}|. \tag{10}$$

False Positives (FP): The number of incorrectly predicted non-edges for pairs of nodes with the same label; nodes that are not connected but have the same labels.

$$FP = |\{(i,j) \mid e_{ij} = 0 \wedge y_i = y_j\}|. \tag{11}$$

False Negatives (FN): The number of incorrectly predicted edges between nodes with different labels.

$$FN = |\{(i,j) \mid e_{ij} = 1 \wedge y_i \neq y_j\}|. \tag{12}$$

We know that the sum of all the metrics is equal to the total number of possible edges in the graph; $TP + FP + TN + FN = \frac{N(N-1)}{2}$, Hence:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\frac{N(N-1)}{2}}. \tag{13}$$

Expressing TP in terms of homophily, based on Definition 1, yields to:

$$TP = h|E|. \tag{14}$$

For TN, we obtain:

$$\begin{aligned} TN &= |\{(i,j) \mid e_{ij} = 0 \wedge y_i \neq y_j\}| \\ &= |\{(i,j) \mid e_{ij} = 0\}| - |\{(i,j) \mid e_{ij} = 0 \wedge y_i = y_j\}| \\ &= \frac{N(N-1)}{2} - |E| - FP. \end{aligned} \tag{15}$$

Here, $|\{(i,j) \mid e_{ij} = 0\}|$ represents the number of non-edges that is equal to the total number of possible edges in the graph ($\frac{N(N-1)}{2}$) minus the number of existing edges ($|E|$) whereas the second term corresponds to FP. Developing the term FP, we obtain the following:

$$\begin{aligned} FP &= |\{(i,j) \mid e_{ij} = 0 \wedge y_i = y_j\}| \\ &= |\{(i,j) \mid y_i = y_j\}| - |\{(i,j) \mid e_{ij} = 1 \wedge y_i = y_j\}| \\ &= |\{(i,j) \mid y_i = y_j\}| - TP \\ &= |\{(i,j) \mid y_i = y_j\}| - h.|E|. \end{aligned}$$

The first term counts combinations of nodes pairs with the same label, expressed as:

$$|\{(i,j) \mid y_i = y_j\}| = \sum_{c=1}^{C} \binom{N_c}{2},$$

Where N_c is the number of nodes with label c, and C the number of labels. Defining $\alpha_c = \frac{N_c}{N}$ as the proportion of

nodes with label c, and noting $\sum_{c=1}^{C} \alpha_c = 1$, we derive:

$$FP = \sum_{c=1}^{C} \frac{\alpha_c N(\alpha_c N - 1)}{2} - h.|E| = \frac{N^2}{2} \sum_{c=1}^{C} \alpha_c^2 - \frac{N}{2} - h.|E|. \quad (16)$$

Substituting the expression for FP (16) into TN (15), we obtain:

$$TN = \frac{N(N-1)}{2} - |E| - FP = \frac{N(N-1)}{2} - |E| - \left(\frac{N^2}{2} \sum_{c=1}^{C} \alpha_c^2 - \frac{N}{2} - h.|E| \right). \quad (17)$$

Substituting the expressions for TP (14) and TN (17) into the accuracy (13) and applying the graph density definition (7), we deduce the accuracy of the attack as follows:

$$\begin{aligned} Acc &= \frac{TP + TN}{\frac{N(N-1)}{2}} \\ &= \frac{h.|E| + \frac{N(N-1)}{2} - |E| - \left(\frac{N^2}{2} \sum_{c=1}^{C} \alpha_c^2 - \frac{N}{2} - h.|E| \right)}{\frac{N(N-1)}{2}} \\ &= \frac{4h|E|}{N(N-1)} + 1 - \frac{2|E|}{N(N-1)} - \frac{N}{N-1} \sum_{c=1}^{C} \alpha_c^2 + \frac{1}{N-1} \\ &= 2h.d - d + \frac{N}{N-1}(1 - \sum_{c=1}^{C} \alpha_c^2). \end{aligned}$$

To establish the upper bound of accuracy, we apply the lower bound $\frac{1}{C} \leq \sum_{c=1}^{C} \alpha_c^2$, which is a direct application of the $L_1 - L_2$ norm inequality:

$$C \sum_{c=1}^{C} \alpha_c^2 \geq \sum_{c=1}^{C} \alpha_c \qquad \sum_{c=1}^{C} \alpha_c^2 \geq \frac{1}{C}$$

Finally, we prove that the upper bound on the accurracy Acc, using the lowed bound above is:

$$Acc \leq 2hd - d + \frac{N}{N-1}(1 - \frac{1}{C})$$

This upper bound is attained when $\alpha_c = \frac{1}{C}$, corresponding to a uniform distribution of labels across nodes. This concludes our proof, demonstrating the derived upper bound for the accuracy of the attack.

Appendix C. Proof of theorem 2

The essence of this proof is to demonstrate the specific conditions required to prevent the overlap of cosine similarities between prediction vectors of nodes with the same label and those with different labels. Therefore, we establish the minimum probability assigned to the true label of nodes that ensure the cosine similarity between any pair of prediction vectors with the same labels is greater than those with different labels. For a binary classification task involving two classes, c_1 and c_2. Let θ represent the minimum probability assigned to the true label of a node, we have:

- $P^1_{max} = [1, 0]$: the maximum posterior probability for class c_1,
- $P^1_{min} = [\theta, 1-\theta]$: the minimum θ posterior probability for class c_1,
- $P^2_{min} = [1-\theta, \theta]$: the minimum θ posterior probability for class c_2.

The goal is to find a value of θ. This value should ensure that the cosine similarity between P^1_{max} and P^1_{min} is greater than the cosine similarity between P^1_{min} and P^2_{min}. Here, P^1_{max} and P^1_{min} represent the prediction vectors for nodes within the same class, yielding the minimum cosine similarity for nodes with the same label. Conversely, P^1_{min} and P^2_{min} represent nodes with different labels, yielding the maximum cosine similarity for nodes with different labels. To solve this problem, We need to find θ such that the cosine similarities are equal as a boundary condition, and then ensure θ is set such that it satisfies the inequality in practice.

The cosine similarity between two vectors A and B is given by $\text{Cos}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$.

For the pair (P^1_{max}, P^1_{min}), the cosine similarity is:

$$Cos(P^1_{max}, P^1_{min}) = \frac{1 \cdot \theta + 0 \cdot (1-\theta)}{\sqrt{1^2 + 0^2}\sqrt{\theta^2 + (1-\theta)^2}} = \frac{\theta}{\sqrt{\theta^2 + (1-\theta)^2}}. \quad (18)$$

For the pair (P^1_{min}, P^2_{min}), the cosine similarity is:

$$Cos(P^1_{min}, P^2_{min}) = \frac{\theta \cdot (1-\theta) + (1-\theta) \cdot \theta}{\sqrt{\theta^2 + (1-\theta)^2}\sqrt{(1-\theta)^2 + \theta^2}} = \frac{2\theta(1-\theta)}{\theta^2 + (1-\theta)^2}. \quad (19)$$

To find θ such that the similarity conditions are equal, which represents the boundary condition for our inequality, we equate the two cosine similarities (18, 19):

$$\frac{\theta}{\sqrt{\theta^2 + (1-\theta)^2}} = \frac{2\theta(1-\theta)}{\theta^2 + (1-\theta)^2}.$$

Solving this equation yields $\theta = 0$ and $\theta = \frac{3}{2} - \frac{\sqrt{3}}{2}$. The solution $\theta = 0$ is not practical for our purpose as it does not represent a valid probability for class prediction. Thus, we consider the solution $\theta = \frac{3}{2} - \frac{\sqrt{3}}{2}$, which signifies the minimum probability that must be assigned to the true class of a node to satisfy our initial condition.

For a model trained with cross-entropy loss, the loss of a sample is given by $l_i = -\log(p_t)$, where p_t is the

probability assigned to the true label of the node. Therefore, based on our finding, the condition on θ translates to a condition on the sample loss as:

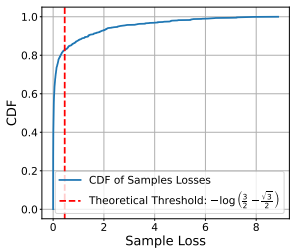
$$l_i \leq -\log\left(\frac{3}{2} - \frac{\sqrt{3}}{2}\right)$$

This result establishes a threshold for the probability assigned to the true class, ensuring that inter-class prediction similarities are always greater than intra-class prediction similarities, rendering the decision of output-based LIA equivalent to that of label-based LIA. To achieve identical inference outcomes between output-based and label-based LIA, we incorporate our solution θ into one of the cosine similarity equations (18, 19). This substitution yields a decision threshold $\tau = \frac{\sqrt{3}}{2}$, ensuring both LIA methods infer equivalent graphs.

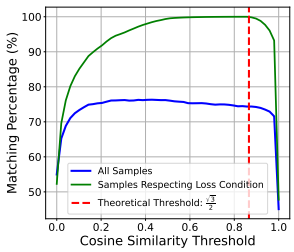
Appendix D. Validation of theorem 2 through empirical analysis

In this experiment, we aim to validate Theorem 2, which states that for a GNN trained with cross-entropy loss for a binary classification task, the output-based LIA with a cosine similarity threshold $\tau = \frac{\sqrt{3}}{2}$ infers the same graph as the label-based LIA, provided that each training sample's loss l_i satisfies $l_i \leq -\log\left(\frac{3-\sqrt{3}}{2}\right)$. We use the Cora dataset, focusing on a subgraph containing the two most populous classes. A two-layer GCN is defined with an input GCN layer of 16 hidden units, followed by a ReLU activation, dropout, and an output GCN layer producing two units. The model is trained using the Adam optimizer with a learning rate of 0.01 and weight decay of 5e-4, minimizing the cross-entropy loss over 200 epochs.

First, we compute and plot the CDF of the cross-entropy sample losses, highlighting the theoretical condition $-\log\left(\frac{3-\sqrt{3}}{2}\right)$. We then perform the output-based attack and the label-based attack for all samples and for samples respecting the loss condition. By varying the cosine similarity threshold, we plot the matching percentage link predictions between the two attacks, marking the theoretical cosine similarity threshold $\frac{\sqrt{3}}{2}$. The results show that the matching percentage is 100% when only samples respecting the loss condition are considered. These experimental results validate the theorem by demonstrating that pairs of samples with cross-entropy loss below the theoretical threshold have the same link predictions of the output-based and label-based attacks, confirming that the cosine similarity threshold can infer the same graph structure as the label-based approach when the sample loss condition is met.



(a) CDF of sample losses



(b) Sensitivity analysis of cosine similarity threshold

Figure 8: Analysis of sample losses and cosine similarity threshold

TABLE 5: Accuracy of gradient-based link inference attack on different GNN architectures

Dataset	GAT	GCN	GraphSAGE
Photo	84.27 ± 0.62	82.12 ± 1.32	82.34 ± 1.31
Cora	82.23 ± 0.97	81.71 ± 0.21	81.40 ± 0.65
Computer	79.38 ± 0.39	78.23 ± 1.26	78.06 ± 1.32
Citeseer	83.40 ± 1.15	82.76 ± 0.38	82.35 ± 0.15

Appendix E. Ablation Studies

E.1. Impact of GNN's architecture on the performance of gradient-based LIA

We study the influence of the architecture of the attacked GNN on the accuracy of our gradient-based LIA. For the architectures under examination, we have included GAT and GraphSAGE, as they are some of the most well-known architectures used in the literature. GATs are distinguishable by their ability to assign varying levels of importance to nodes in a neighborhood, using attention mechanisms. As observed in our results (see Table 5), GATs generally exhibit higher values across various datasets, indicating a greater susceptibility to LIAs. This slight increased vulnerability may be attributed to the attention mechanism in GATs that learns the attention coefficients of the edges (See Eq 2.2), which may results in a slight memorization of the edges of the target graph. GraphSAGE utilizes a neighborhood sampling and aggregation approach to generate node embeddings. This method, as reflected in the table, generally shows lower values compared to GAT, suggesting reduced susceptibility to LIAs. The fixed-size neighborhood sampling employed by GraphSAGE potentially obscures some links between nodes, providing a form of obfuscation against such attacks. The performance of GCNs, which falls between GAT and GraphSAGE in our study, suggests a moderate level of susceptibility to LIAs. It indicates that while GCNs do learn node connections, they neither reveal as much detail about the links in the graph as GATs nor obscure connections as GraphSAGE.

E.2. Impact of the number of parties on the performance of gradient-based LIA

In this study, we analyze the sensitivity of the number of parties on the performance of our gradient-based attack in varying multi-party settings. We extend our VFGL protocol by adding more benign participants that contribute features, while maintaining one adversary and one victim party. We adopt the same multiparty protocol utilized in the baseline study, fixing the adversary's feature ratio at 20%. The remaining features are evenly distributed among the benign participants and the target victim. We vary the number of parties from 2 to 5 to ensure each participant has at least 20% of the features. From Figure 9, we observe that the performance of our gradient-based attack is independent of the number of parties. This is primarily because the attack exploits the label information embedded in the gradients, regardless of the number of clients participating in the VFGL setting. The label-based attack's performance remains constant across different numbers of clients. This is because in our VFGL setting (Figure 1), the server owns the labels, so the available label information is unchanged regardless of the number of participating parties.

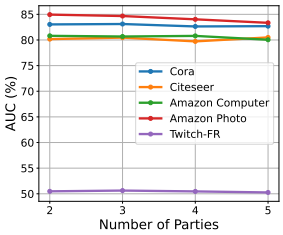


Figure 9: Performance analysis with varying number of parties. The results show the AUC values across different datasets as the number of parties changes from 2 to 5.

E.3. Impact of model complexity on the performance of gradient-based LIA

We investigate the influence of model complexity on our gradient-based LIA's accuracy using ResNet-like architectures [He et al.(2016)] for both adversary's and server's models. Our results (Table 6) on the Cora dataset show the attack maintains high accuracy across different complexity levels, from a simple 2-layer neural network (81.71% ± 0.21%) to architectures with multiple residual blocks (81.26% ± 0.18% with four blocks). This robustness can be attributed to two factors: complex models' gradients still carry sufficient graph structure information, and the attack leverages label information, which remains present in gradients regardless of model complexity.

Appendix F. Label perturbation algorithm

We present the label perturbation defense algorithm in two parts: optimization (Algorithm 5) and redistribution (Algorithm 6). Given the accuracy formula for Label-based LIA (Equation 4), our experimental observations (Table 4) show that terms involving homophily ratio (h) and density (d) are negligible due to the low density of typical graphs in our scenarios. We can thus approximate the accuracy as:

$$Acc \approx \frac{N}{N-1}(1-\sum_{c=1}^{C}\alpha_c^2) \tag{20}$$

To minimize this approximated accuracy and reduce the effectiveness of the Label-based LIA, our objective becomes maximizing $\sum_{c=1}^{C}\alpha_c^2$. Algorithm 5 optimizes label proportions to maximize $\sum_{c=1}^{C}\alpha_c^2$ within the given budget B using quadratic programming. Algorithm 6 then redistributes labels to match these optimized proportions through iterative balancing between classes.

TABLE 6: Accuracy of gradient-based LIA on ResNet-like architectures (Cora dataset)

Number of Residual Blocks	Accuracy (%)
Baseline	81.71 ± 0.21
1	81.70 ± 0.21
2	81.54 ± 0.19
3	81.47 ± 0.23
4	81.26 ± 0.18

Algorithm 5 Label Proportion Optimization

Require: Initial label proportions α_c^{init}, budget B
Ensure: Optimized label proportions α_c^*
Solve the following optimization problem:
$\max_{\alpha_c} \sum_{c=1}^{C}\alpha_c^2$
subject to:
$\sum_{c=1}^{C}\alpha_c = 1$
$\sum_{c=1}^{C}\max(0, \alpha_c^{init} - \alpha_c) \leq B$
$\alpha_c \geq 0, \forall c \in \{1, \ldots, C\}$
return α_c^*

Algorithm 6 Label Redistribution

Require: Current labels y, optimized proportions α^*
Ensure: Obfuscated labels y'
Calculate target number of nodes for each class based on α^*
Determine classes needing more nodes (C^+) and fewer nodes (C^-)
for each class $c_t \in C^+$ **do**
  for each class $c_s \in C^-$ **do**
    Flip labels from c_s to c_t until:
      - c_t has enough nodes, or
      - c_s has no more nodes to give
return y'