# Crypto'Graph: Leveraging Privacy-Preserving Distributed Link Prediction for Robust Graph Learning

Sofiane Azogagh
Université du Québec à Montréal
Montréal, Canada

Zelma Aubin Birba
birba.zelma_aubin@courrier.uqam.ca
Université du Québec à Montréal
Montréal, Canada

Sébastien Gambs*
Université du Québec à Montréal
Montréal, Canada

Marc-Olivier Killijian†
Université du Québec à Montréal
Montréal, Canada

## ABSTRACT

Graphs are a widely used data structure for collecting and analyzing relational data. However, when the graph structure is distributed across several parties, its analysis is challenging. In particular, due to the sensitivity of the data each party might want to keep their partial knowledge of the graph private, while still be willing to collaborate with the other parties for tasks of mutual benefit, such as data curation or the removal of poisoned data. To address this challenge, we propose Crypto'Graph, an efficient protocol for privacy-preserving link prediction on distributed graphs. More precisely, it allows parties partially sharing a graph with distributed links to infer the likelihood of formation of new links in the future. Through the use of cryptographic primitives, Crypto'Graph is able to compute the likelihood of these new links on the joint network without revealing the structure of the private graph of each party, even though they know the number of nodes they have, since they share the same graph in terms of nodes but not the same links. Crypto'Graph improves on previous works by enabling the computation of a diverse set of similarity metrics in parallel without any additional cost. The use of Crypto'Graph is illustrated for defense against graph poisoning attacks, in which potential adversarial links are identified without compromising the privacy of the graphs of individual parties. The effectiveness of Crypto'Graph in mitigating graph poisoning attacks and achieving high prediction accuracy on a node classification task using graph neural networks is demonstrated through extensive experimentation on two real-world datasets.

## CCS CONCEPTS

• **Security and privacy** → *Data anonymization and sanitization*; *Intrusion/anomaly detection and malware mitigation*; *Distributed systems security*.

*supported by the Canada Research Chair program and a Discovery Grant from NSERC
†supported by a Discovery Grant from NSERC


*CODASPY '24, June 19–21, 2024, Porto, Portugal*

ACM ISBN 979-8-4007-0421-5/24/06…$15.00
https://doi.org/10.1145/3626232.3653257

## KEYWORDS

Security, privacy, secure multiparty computation, graph

**ACM Reference Format:**
Sofiane Azogagh, Zelma Aubin Birba, Sébastien Gambs, and Marc-Olivier Killijian. 2024. Crypto'Graph: Leveraging Privacy-Preserving Distributed Link Prediction for Robust Graph Learning . In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy (CODASPY '24), June 19–21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3626232.3653257

## 1 INTRODUCTION

In today's digital age, graphs have emerged as the predominant format for representing relational data, as they naturally capture both the relationships and structures inherent in such datasets. Indeed, from social networks [37] to biological systems [29], the interconnection of entities can be easily visualized and understood through graphs. However, as data becomes increasingly distributed, new challenges arise with respect to their analysis.

For example, in a scenario in which the structure of a graph is distributed across multiple parties (social network distributed across multiple servers, shared knowledge graph, etc), the objective might be to study this structure without any party disclosing the private details of their segment. Such an analysis could involve predicting potential future links [11, 20, 41, 42] or identifying malicious links that an adversary has introduced to compromise the graph's integrity [38, 39, 43]. As such attacks might happen without being noticed, it is crucial to act proactively to defend against them.

To solve these issues, we propose Crypto'Graph, a novel protocol for privacy-preserving link prediction on distributed graphs. To avoid privacy leakage, Crypto'Graph leverages cryptographic primitives such as Diffie-Hellman shared secrets and Private Set Intersection Cardinality (PSI-CA), which ensure that similarities used for link prediction can be computed on the joint network without exposing the specifics of the private individual graphs. Furthermore, Crypto'Graph can be used as a robust defense against graph poisoning attacks. More precisely, by predicting potential links without jeopardizing the confidential information of individual nodes, it can be used to effectively detect adversarial links, improving the quality of downstream graph learning tasks.

The main contributions of this paper are the following.

- We introduce Crypto'Graph, a new protocol for distributed privacy-preserving link prediction on graph data relying on

the similarity via the computation of the numbers of common neighbors. Crypto'Graph is more efficient, by several orders of magnitude, than state-of-the-art methods while making it possible to derive other metrics such as the Jaccard and Cosine similarity measures at no extra cost (*i.e.*, in parallel). Finally, unlike previous works, it can be used on the complete graph without compromising the confidentiality of the private individual graphs as we demonstrate by proving its security against graph reconstruction attacks.
- The applicability of Crypto'Graph is illustrated through a collaborative defense scenario against graph poisoning attacks. More precisely, we show how to leverage our protocol to privately derive link likelihood information in a distributed manner, enabling the different parties to identify adversarial links and remove them for a better utility on subsequent tasks, such as the training of graph neural networks. In addition, we show that the benefit of collaborating via our protocol varies according to the common knowledge between the participants, the amount of adversarial links introduced as well as the type of attack conducted. Nonetheless, experiments on real datasets demonstrate that it is almost always beneficial to cooperate even when the data of one party has not been poisoned. This encourages the use of our solution even as a preprocessing approach to clean the graph when no attack has been detected.

The outline of the paper is as follows. First, in Section 2, we review the related work on link prediction both in the centralized and distributed settings before introducing in Section 3 the background notions on link prediction, graph neural network and private set intersection, which are necessary to the understanding of our work. Afterwards, in Section 4 we describe Crypto'Graph, our protocol for privacy-preserving distributed link prediction before detailing how Crypto'Graph can be used to defend against graph poisoning in Section 5, in which we evaluate it on a real-world graph dataset.

## 2 RELATED WORK

Based on the structure of the graph and potential additional information (such as the values of node attributes), link prediction algorithms [11, 20, 41, 42] aim at inferring future potential links in dynamic networks. One classical method for predicting a link between two nodes consists in measuring the similarity of these nodes by leveraging their neighborhood structure and predict new links or not, based on the assumption that nodes that have common neighbors have more chance to develop future connections. This similarity can be computed on the structural information between nodes but also by considering their attributes. For instance, in research collaboration networks, Newman has shown that, in domains such as physics, the more coauthors two researchers have in common (*i.e.*, the more the have neighbors in common), the more they are likely to collaborate in the future [27]. In addition, he has also observed that a scientist taken at random is more likely to make new collaborations if he has many past ones, introducing the notion of preferential attachment. Other similarity measures such as the Jaccard similarity [17], the Cosine similarity [36] or the Adamic-Adar index [2] can also be used for link prediction.

Machine learning-based models have also been proposed to tackle the link prediction problem. For instance, Kashima and Abe have extracted topological features from the graph and used them to train a model for supervised link prediction [18]. Zhang and Chen have designed SEAL, a neural network-based architecture for capturing the link formation law on a graph [41]. In particular, they have proven that low order subgraphs, composed only of few hops-neighbors, are often sufficient to estimate high order metrics that reason on the whole graph. They have also proposed a framework for predicting new links using network-based heuristics applied on these subgraphs. Other link prediction methods also try to learn informative features for nodes while avoiding to explore the complete graph. For example, Perozzi and collaborators have used random walks in conjunction with the SkipGram model [23] (commonly used in natural language processing to vectorize words) to build representations of nodes based on samples of their neighborhoods [30]. The Node2vec embedding technique [15] also relies on a similar approach for neighborhood sampling in the graph to generate continuous node feature representations that can then be used to compare note to predict new links.

However, the aforementioned methods are focused on the centralised setting, in which there is a single graph in the hands of only one party. Nonetheless, subsequent works have proposed approaches inspired by the previous ones adapted to the multi-graph or distributed graph setting. More precisely, the multi-graph link prediction can be defined as the setting in which each party owns a graph with potentially different nodes and links. In contrast, the distributed graph setting assumes that the parties hold the same graph with respect to their nodes but not necessarily the same links. For instance, some works have proposed to allow members of a decentralised social network like Mastodon [1], to define privacy controls on their connections by specifying which of their friendships they are willing to disclose [42]. The service provider can then use this information to train a logistic regression model that privately makes friendship recommendations.

Another recent work has considered the distributed graph setting and enables multiple subgraph owners to make link predictions by computing similarity metrics in a secure manner [40]. More precisely by using secret sharing techniques, their approach can be used to privately aggregate the local similarity scores, thus allowing the different parties to make their decision based on the private aggregate. However, their approach can induce an accuracy loss in the prediction because it does not take into account the cross-party similarities (similarity between a node $x$ in one subgraph, and a node $y$ in another subgraph). Another recent approach [11] computes the common neighbors similarity measure using three instances of a Private Set Intersection (PSI) protocol, which we will detail later in Section 3.3. However, it is not clear how this protocol can be extended for the computation of other similarity measures and its computational cost is higher than our approach.

In this paper, we introduce a method that addresses the shortcomings of the two former previous works. More precisely, our protocol improves on the accuracy compared to [40] while allowing to compute a wide range of similarity measures and thus not being limited to common neighbors as in [11].

[1] https://joinmastodon.org

## 3 PRELIMINARIES

In this section, we review the notations we will be using, before introducing the preliminary notions of link prediction, graph neural network and private set intersection, that are necessary to the understanding of our work. The notations employed throughout the paper are presented in Table 1.

| | | |
|---|---|---|
| Graph | $\mathcal{V}$ | set of vertices (*i.e* nodes) |
| | $\mathcal{E}$ | set of edges (*i.e* links) |
| | $G = (\mathcal{V}, \mathcal{E})$ | graph of nodes $\mathcal{V}$ and links $\mathcal{E}$ |
| | $\Gamma(x) \subseteq \mathcal{V}$ | neighbors of $x$ in $G$ (*i.e*, the nodes in $\mathcal{V}$ that share a link with $x$) |
| GNN | $\mathcal{X} \in \mathbb{R}^{d \times \lvert\mathcal{V}\rvert}$ | feature matrix where $\mathcal{X}[i]$ is $d-$ dimensional feature vector of node $i$ |
| | $\mathcal{Y} \in \mathbb{R}^{\lvert\mathcal{V}\rvert}$ | labels of nodes where $\mathcal{Y}[i]$ is the label of node $i$ |
| | $G = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$ | extended graph (with features and labels) |
| | $\mathcal{A}$ | adjacency matrix of a graph |
| | $\tilde{A}$ | adjacency matrix with self-connections inserted (*i.e* $\tilde{A} = A + I_{\mathcal{V}}$) |
| | $\tilde{D}$ | degree matrix (*i.e*, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$) at a specific layer |
| Crypto | $p, q$ | large prime $p$ and integer $q$ such that $q$ divides $p-1$ |
| | $\mathbb{Z}_p$ | set of integers modulo p |
| | $\mathbb{G}_q$ | multiplicative group of order $q$ |
| | $g$ | generator of $\mathbb{G}_q$ |

**Table 1: Summary of the notations used in this paper.**

### 3.1 Link Prediction

Link prediction [20, 27] is a graph learning task that aims to infer the potential existence of links between nodes that are not currently connected. Link prediction has many possible applications, such as friend recommendation in a social network [20] or in healthcare for the the study of contacts for epidemic control purposes [4]. A lot of link prediction methods are based on the computation of the similarity between neighborhoods of pairs of nodes. More formally, considering two nodes $x, y \in \mathcal{V}$, we can compute their similarity in the following ways:

- **Common Neighbors:** The common neighbors similarity of $x$ and $y$ is defined as

$$\text{CN}(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

, which is the basically the size of the intersection between the two sets of neighbours.

- **Jaccard:** The Jaccard similarity between $x$ and $y$ is defined as

$$\text{J}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} = \frac{\text{CN}(x, y)}{|\Gamma(x) \cup \Gamma(y)|}$$

, which can be defined as the size of the intersection over the size of the union of the two sets of neighbours.

- **Cosine:** The cosine similarity between $x$ and $y$ is given by

$$\text{Cosine}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{|\Gamma(x)|} \times \sqrt{|\Gamma(y)|}} = \frac{\text{CN}(x, y)}{\sqrt{|\Gamma(x)|} \times \sqrt{|\Gamma(y)|}}$$

, which is basically the size of the intersection between the two sets of neighbours normalized by a function of their respective size.

In real-world scenarios, the graph might be distributed among different parties. For example, a distributed social network like Mastodon can be made of different instances, each representing the local relationships of their users as a graph that is part of the global network. Knowledge graphs collected by different entities are another example of distributed graph. To account for this and without loss of generality, we consider that the whole graph $G = (\mathcal{V}, \mathcal{E})$ is distributed among two parties $P_1$ and $P_2$ such that each party entirely knows $\mathcal{V}$ but only a fraction of $\mathcal{E}$. Such setting can be easily generalized to a more distributed setting (with more than two parties) as shown in [40]. Let $G_1(\mathcal{V}, \mathcal{E}_1)$ denote the graph of $P_1$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ the graph of $P_2$ such that $\mathcal{E}_1 \subseteq \mathcal{E}$ and $\mathcal{E}_2 \subseteq \mathcal{E}$. We consider the scenario in which $P_1$ and $P_2$ want to collaborate to predict the existence of a link in their respective graphs without revealing them due to confidentiality issues that may arise (*e.g.*, the graphs may represent personal relationships). A possible solution to predict a link between $x, y \in \mathcal{V}$ is for $P_1$ to privately share $\Gamma_1(x)$ and $\Gamma_1(y)$ with $P_2$, who in turn privately shares $\Gamma_2(x)$ and $\Gamma_2(y)$.

Subsequently, the link prediction primitive computes the similarity measure on the joined neighborhoods of $x$ and $y$ and outputs this measure to both $P_1$ and $P_2$. Finally, each party decides to predict a link based on this result and a threshold chosen independently as illustrated in Figure 1. The threshold is chosen to maximize the prediction of links between nodes that actually create links in the future, while minimizing false predictions.

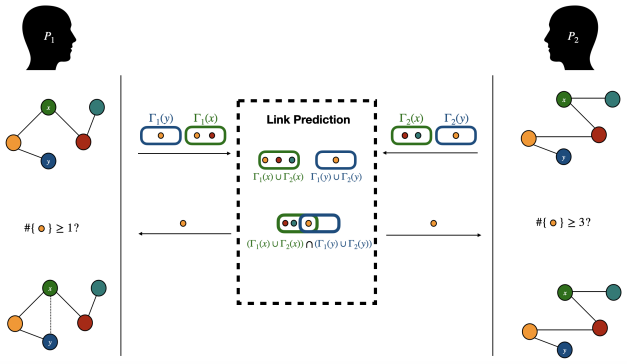



**Figure 1: An illustration of a collaboration between $P_1$ and $P_2$ to predict a link between the node $x$ (green) and $y$ (blue). Here, the link prediction primitive will output $\text{CN}(x, y)$ that each party $P_i$ will compare with their personal threshold, which is 1 for $P_1$ and 3 for $P_2$. Based on the result of this comparison, a link between $x$ and $y$ can be added or not, in one or both graphs $G_1$ and $G_2$.**

### 3.2 Graph Neural Networks

To take advantage of their structural information, deep learning approaches have been adapted to the graph data format with the advent of Graph Neural Networks (GNNs) [19, 34], which have made it possible to harness the structure and the node properties in graphs for various learning tasks. In a nutshell, most GNN models take as input a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$ and learn the appropriate features of each node, edge or subgraph to perform the predictions. Different families of approaches have been introduced over the years. For instance, representation learning algorithms [15, 30] are deep learning-based architectures that represent nodes, edges or subgraphs of a graph as a multidimensional vector. This vector can then be used for various prediction tasks, such as community identification (*i.e.*, a form of clustering) and link prediction.

In this work, we focus on Graph Convolutional Networks (GCNs), a specific type of network mimicking convolutions on images [19]. In GCNs, new node features are created with each node aggregating information from its neighbors at each layer. More precisely, we consider a semi-supervised node classification task that aims at predicting the labels of test nodes based on the features, edges and labels of a set of training nodes. Following the approach taken in [19, 43], we consider a two layer neural network given by the formula :

$$Z = \text{softmax}\left(\hat{A}\ \text{ReLu}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right),$$

in which $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, and $W^{(0)}, W^{(1)}$ are the trainable parameters of the network and $Z$ is the prediction of the network for a given node.

Similarly to classical deep learning models, GNNs have been shown to be vulnerable to adversarial attacks. More specifically for the setting considered here, this means that an adversary can poison the graph by injecting links or altering node features to influence the prediction made by the model [38, 43]. Some of the classical defenses against such poisoning attacks leverage link prediction techniques [38, 39], in which a likelihood score is computed for each edge, thus identifying unlikely links and considering them as being potentially adversarial (before cleaning them).

### 3.3 Private Set Intersection

A Private Set Intersection (PSI) protocol [14] allows two or more parties, each holding a private set of items to compute the intersection of their sets (*i.e.*, items that they have in common) without revealing any additional information. PSI has found many applications in the real world, such as private data mining [28], the analysis of genomics or medical data [6, 24, 33, 35] or even botnet detection [26]. Since it was first introduced, PSI has evolved in many subvariants of protocols according to the scenario in which it is applied. For example, if only one of the parties needs to obtain the intersection at the end of the computation, it is called a *one-way* PSI, while otherwise it is a *mutual* PSI. Another type of PSI protocol was developed according to the size of the set of each party. For instance, if the sizes of the private sets are similar, it is refer to as a *balanced* PSI while otherwise it is called an *unbalanced* one. More generally, a classification has recently been proposed in [25].

In other scenarios, the parties may want to know *how much* they share but not what exactly they have in common. This can be solved using a PSI-CA (standing for Private Set Intersection CArdinality), which cannot be directly instantiated from classical PSI protocols. Among the cryptographic building blocks that can be combined to develop a PSI-CA protocol, we can cite for instance homomorphic encryption, as seen in works like [9, 16, 21], oblivious transfer [13], generic public key techniques described in [10, 31], or even commutative encryption similar to the method outlined in [22]. In this latter approach, one party, $P_1$, initiates the protocol by sending its own set of items $X$ encrypted as $Enc_{PK_1}(X)$ to $P_2$. Afterwards $P_2$ shuffles the elements and sends them back to $P_1$ as $Enc_{PK_2}(Enc_{PK_1})(X)$. Additionally, $P_2$ sends its own set encrypted as $Enc_{PK_2}(Y)$. By using a commutative encryption scheme, $P_1$ can "delete" its corresponding key $PK_1$ from $Enc_{PK_2}(Enc_{PK_1}(X))$ to obtain $Enc_{PK_2}(X)$ and then compare the number of correspondence with $Enc_{PK_2}(Y)$. This approach has inspired a lot of subsequent PSI-CA including the one that we use in this paper which is based on [8].

## 4 CRYPTO'GRAPH

In this section, we present our protocol Crypto'Graph, which enables to perform link prediction on graph data distributed between two parties in a privacy-preserving manner. However, our protocol can easily be generalized to more than two parties as it has been done before in [40].

### 4.1 Description

We propose two close variants of our protocol : one that ensures maximum security via the refreshing of encryption keys at each prediction, and an optimization of this first protocol based on a caching mechanism.

*Basic protocol.* Crypto'Graph is close in spirit to the PSI functionality, in the sense that it privately computes the common neighbors of two nodes on a distributed graph. Let $G_1 = (\mathcal{V}, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}, \mathcal{E}_2)$ be the private subgraphs owned by the semi-honest parties $P_1$ and $P_2$. The main idea behind our approach is the following: by representing the union graph $G_1 \cup G_2$ as a data structure that masks the neighborhood of each node while keeping its cardinality, it is possible to count the common neighbors of any two nodes. To achieve this objective, we propose a solution based on the use of Diffie-Hellman shared secrets [12]. Our solution can be divided in two phases: an offline phase that can be precomputed before the beginning of the concrete protocol, and the online execution of the protocol on the precomputed data.

More precisely, for a link prediction performed between nodes $x$ and $y$, our protocol described in Figure 2 allows each party to represent the neighborhoods of the nodes as sets $\{g^{\alpha\beta x_i}\}_{i\in\{1,\dots,|\Gamma_k(x)|\}}$ and $\{g^{\alpha\beta y_i}\}_{i\in\{1,\dots,|\Gamma_k(y)|\}}$, with $k \in \{1, 2\}$ and in which $\alpha$ and $\beta$ are secrets randomly sampled respectively and independently by $P_1$ and $P_2$. Those secrets have the property that they can be used to compare the sets without disclosing the individual elements, which we leverage to compute the oblivious union of the neighbors of $x$ in both graphs, as well as for node $y$. Afterwards, we count the common elements of those sets to determine the size of the intersection.

One of the strengths of our method is that by computing the intermediary sets of neighbors of $x$ and $y$ separately, those results can be reused to compute the Jaccard similarity by dividing the common neighbors score with the size of the union of the neighbors of $x$ and $y$. More generally, our method allows for the computation of any similarity metric involving the sizes of the immediate neighborhood of two nodes [17, 27]. Figure 2 provides a graphic illustration of the Crypto'Graph protocol, in which the offline phase is represented in gray, and the online one in black.

*Optimization via caching.* As an additional contribution, we propose an optimization of the aforementioned algorithm based on a caching mechanism. More precisely, by keeping the same $\alpha$ and $\beta$ keys across predictions, for each node $x$ that has been involved in a previous prediction, we can reuse the encryptions of its neighbors $\{g^{\alpha\beta x_i}\}_{i\in\{1,\dots,|\Gamma(x)|}$ for all subsequent predictions involving $x$. A detailed analysis of this caching mechanism and its security are provided in Section 4.3.

## 4.2 Performance Results

To assess the performance of our protocol, we evaluate it on a real world graph commonly used in the literature for link prediction. Our experiments are conducted on two subgraphs of the Polblogs dataset from [32], which represents relationships between 1490 political blogging websites in the USA by depicting the websites as nodes of the graph, and hypertext links between them as edges in the graph. The original dataset contains a total of 16715 links.

The two subgraphs are obtained by sampling links from the original dataset. More concretely, we use a method inspired by the one in [40] to determine the membership of a link to one or both of the graphs : we choose two values $q_1, q_2$ in the interval $[0, 1]$, such that $q_1 \leq q_2$. Then, for each link in the original graph, we sample a random value $v$ in the same interval. If the sampled value lands in $[0, q_1]$, the link is attributed to $G_1$ while if $v \in ]q_1, q_2]$, the link is added to $G_2$ while otherwise the link is attributed to both the subgraphs. This generation of the distributed dataset enables to control the proportions of links owned by one or both of the graphs. The experiments described hereafter are made with $q_1$ and $q_2$ sets respectively to 0.3 and 0.6, which results in each subgraph solely owning 30% and sharing 40% of the graph with the other entity.

Our implementation is single-threaded and developed in C++, and for efficient exponentiation, we use the OpenSSL implementation of the *NIST P-256* [1] elliptic curve. This choice allows to achieve the typical 128-bit security level requirement in cryptographic protocols. Experiments are run on a desktop computer running the 20.04 LTS version of Ubuntu operating system with 64GB of memory, and 16 x 11th generation Intel i9@ 3.50 GHz cores. Since the protocol operates entirely on one machine, there is no network-related delay. In addition, our implementation uses the caching mechanism described previously. We compute the common neighbors heuristic for each pair of nodes in the graph. We also re-implemented the solution of [11] as described by its authors and present their performance next to ours in Table 2, with results obtained being consistent with those presented in their original paper. Three scenarios are considered : (1) predictions on all the node pairs in the graph (all vs all), (2) predictions between a single random node and all the other ones (one vs all) and (3) predictions between two random nodes (one vs one). These results show a drastic improvement of one to several orders of magnitude in both computing time and communication.

## 4.3 Security Model

*4.3.1 Security of the basic protocol.* In this subsection, we introduce the security model of our protocol and prove its security against a semi-honest adversary[2] under well-known cryptographic assumptions that we recall hereafter. In the following, we denote the security parameter as $\lambda$ and $n$ as the number of nodes.

*Definition 4.1 (Discrete Logarithm Assumption).* Let $\mathbb{G}$ be a cyclic group of generator $g$. The Discrete Logarithm Problem (DLP) is hard in $\mathbb{G}$ if, for every efficient algorithm $\mathcal{A}$, the following probability is a negligible function of $\lambda$ :

$$\mathbb{P}[\mathcal{A}(g, g^a) = a].$$

*Definition 4.2 (Decision Diffie-Hellman Assumption).* Let $\mathbb{G}$ be a cyclic group and $g$ be its generator. We assume that the bit-length of group size is $l$. The Decision Diffie-Hellman (DDH) problem is hard in $\mathbb{G}$ if, for every efficient algorithm $\mathcal{A}$, the following probability is a negligible function of $\lambda$:

$$|\mathbb{P}[x, y \leftarrow \{0,1\}^l : \mathcal{A}(g, g^x, g^y, g^{xy}) = 1] -$$

$$\mathbb{P}[x, y, z \leftarrow \{0,1\}^l : \mathcal{A}(g, g^x, g^y, g^z) = 1]|.$$

*Definition 4.3 (One-More-Diffie-Hellman Assumption).* Let $\mathbb{G}$ be a cyclic group of order $q$ and $g$ be its generator. The One-More-DH problem [7] is said to be $(\tau, t)$-hard if for every algorithm $\mathcal{A}$ that runs in time $t$ we have:

$$\mathbb{P}[\{(g_i, (g_i)^x)\}_{i=1,\dots,n+1} \leftarrow \mathcal{A}^{DH_x(.)}(g_1, \dots, g_m)] \leq \tau,$$

in which $m > n$ and $\mathcal{A}^{DH_x(.)}$ is the algorithm $\mathcal{A}$ with access to a "$DH_x(.)$" oracle. We assume that $\mathcal{A}$ can make at most $n$ queries to the $DH_x(.)$ oracle.

Theorem 4.4. *The security of the proposed protocol is ensured by the DLP and the One-More-DH problem if both parties reinitialise their keys $\alpha$ and $\beta$ after each instantiation of the protocol.*

Proof. Let $G = (\mathcal{V}, \mathcal{E})$ denote a graph shared between two parties $P_1$ and $P_2$. Let $x \in \mathcal{V}$ be a node in $G$ and lets denote $x_i$ the elements of $\Gamma_1(x)$ and $x'_i$ the elements of $\Gamma_2(x)$. During the protocol, $P_2$ obtains the elements from $P_1$ in the encrypted form $a_i = g^{x_i\alpha}$ that $P_2$ cannot decrypt without $\alpha$ because of the DLP. Afterwards, $P_2$ encrypts his own elements and sends them in the form $c_i = g^{x'_i\beta}$ before encrypting the elements of $P_1$ and sending them in the form $a'_i = g^{x_i\alpha\beta}$. From here, several scenarios are possible:

(1) If there are no elements in the intersection, then the elements of $P_2$ are protected by the hardness of DLP.
(2) If there is only one element in the intersection, for instance $x_j = x'_l$, $P_1$ can get $g^{x_j\alpha\beta} = g^{x'_l\alpha\beta}$ (by doing a modular exponentiation of $\alpha \in \mathbb{Z}_q$). The hardness of DLP implies that $P_1$ cannot find an algorithm $\mathcal{A}$ that runs in polynomial time to recover $x_j\alpha\beta$. Furthermore, even if we discard the

[2]The term semi-honest adversary refers to a participant of the protocol that does not deviate maliciously from it but tries to infer new knowledge about the inputs of other parties from the information it gathers.

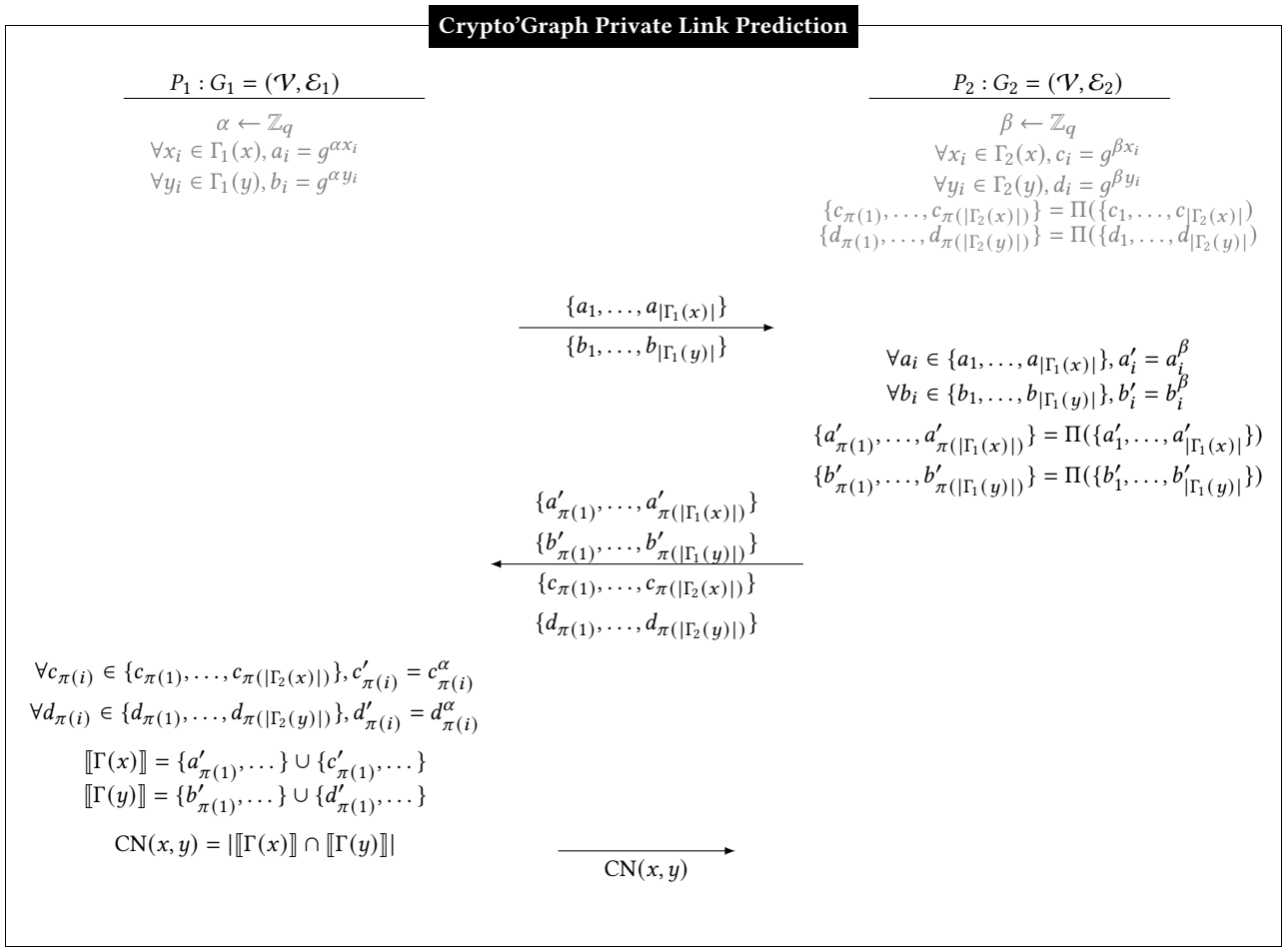



**Figure 2: A diagram of Crypto'Graph, our privacy-preserving link prediction protocol for two nodes. Both parties are assumed to share a known element $g \in \mathbb{G}_q$. In the offline part (gray), both parties generate a key ($\alpha$ for $P_1$ and $\beta$ for $P_2$) and then encrypt the neighbors of the nodes of $x$ and $y$ of their respective graphs. The online phase (black) begins with $P_1$ transmitting the encrypted nodes to $P_2$. Subsequently, $P_2$ proceeds by re-encrypting them using his own key $\beta$ before shuffling them at random and sending them accompanied of his own encrypted nodes. Hence, leveraging the commutativity of the encryption algorithm, $P_1$ can incorporate their key to the nodes of $P_2$. Finally once $P_1$ gets $|\Gamma(x)|$ and $|\Gamma(y)|$ by matching the ciphertexts, they can compute the similarity measure (here Common Neighbors, but it can also be Jaccard and Cosine similarities as well).**

| | | Offline time (ms) | | Online time (ms) | | Communication (MB) | |
|---|---|---|---|---|---|---|---|
| Topology | Protocol | $G_1$ | $G_2$ | $G_1$ | $G_2$ | $G_1$ | $G_2$ |
| all vs all | [11] | 2.82E+07 | 2.85E+07 | 1.42E+07 | 1.41E+07 | 1.02E+04 | 1.10E+04 |
| | **Ours** | **7.56E+02** | **7.23E+02** | **2.72E+05** | **7.87E+02** | **6.93E-01** | **1.43E+00** |
| all vs one | [11] | 2.49E+04 | 2.62E+04 | 1.26E+04 | 1.25E+04 | 8.96E+00 | 9.74E+00 |
| | **Ours** | **3.63E+02** | **3.78E+02** | **1.08E+04** | **7.70E+02** | **6.93E-01** | **1.43E+00** |
| one vs one | [11] | 1.07E+01 | 1.26E+01 | 5.32E+00 | 5.39E+00 | 3.84E-03 | 4.21E-03 |
| | **Ours** | **1.11E+00** | **1.19E+00** | **4.52E-01** | **3.58E-01** | **2.89E-04** | **6.98E-04** |

**Table 2: Online running time and communication performances of Crypto'Graph, compared to [11].**

DLP assumption, $x_j \alpha\beta$ is indistinguishable from a random element of $\mathbb{Z}_q$.

(3) If there are several elements in the intersection, say $x_{j_1} = x'_{l_1}, \ldots, x_{j_m} = x'_{l_m}$, $P_1$ can get $g^{x_{j_1}\beta}, \ldots, g^{x_{j_m}\beta}$ along with

$g_1 = g^{x_{j_1}}, \ldots, g_m = g^{x_{j_m}}$ matching the One-More-DH assumption. Indeed, this scenario arises due to the following reasoning: if $P_1$ possesses a $DH_x(.)$ oracle facilitating the retrieval of $m$ pairs, namely $(g_1, g_1^{\beta}), \ldots, (g_m, g_m^{\beta})$, it is infeasible for $P_1$ to devise an algorithm $\mathcal{A}$ that run in time $t$ and whose probability to recover an additional pair $(g_s, g_s^{\beta})$ where $1 \leq s \leq m$, is greater than $\tau$. The intuition behind this is to exploit $g_s$ in order to access the corresponding element within the intersection set. Thus, $P_2$'s privacy is ensured by the One-More-DH assumption.

□

We have demonstrated that the neighborhood's privacy of the two nodes is preserved during the execution of Crypto'Graph under cryptographic assumptions. One might wonder if this privacy guarantee could be compromised when we apply the protocol to all pairs of nodes. Indeed, an honest-but-curious adversary could attempt to infer information from the number of neighbors of the nodes that have been already considered during the protocol and thereby try to reconstruct the common graph.

Theorem 4.5. *The proposed protocol applied to all nodes is secure against a semi-honest adversary. In addition, the worst-case complexity to recover the entire graph is $O(2^n\sqrt{n})$, in which n is the number of the nodes of the graph.*

Proof. Let $G = (\mathcal{V}, \mathcal{E})$ denote a graph shared between two parties $P_1$ and $P_2$. We have seen that Crypto'Graph performs the PSI-CA described in Figure 2 on pairs of nodes, and the protocol can be instantiated multiple times to perform predictions on all the node pairs in the graph of $P_1$ denoted as $G_1$ and $P_2$ denoted as $G_2$. Focusing on $P_1$ performing a PSI-CA between $x$ and all $x_i \in \mathcal{V} \setminus \{x\}$, it does $n-1$ PSI-CA and sequentially receives $n-1$ outcomes ranging from 0 to $n-2$ (depending on the extent of shared nodes between $x$ and $x_i$ in $G_2$). To assess the extent of information that $P_1$ can deduce while executing the protocol, one possibility is to conduct a brute-force attack. This attack enables to derive an upper bound on the cost incurred by $P_1$ in reconstructing the merged graph $G$ or at least in determining the neighboring nodes of $x \in \mathcal{V}$. Thus, $P_1$ has $z_i = |\Gamma(x) \cap \Gamma(x_i)|$ for each $x_i \in \mathcal{V}/\{x\}$, which leads to $\binom{n-2}{z_i}$ ways of reconstructing the neighborhood of $x$ based on this single prediction. Consequently, the largest number of reconstructions after a single prediction is when $z_i = \lceil \frac{n-2}{2} \rceil$. Therefore in the worst case, $P_1$ has

$$\binom{n-2}{\lceil \frac{n-2}{2} \rceil}^{n-1}$$

possible ways of reconstructing the neighborhood of $x$ after preforming all the predictions, which can be bounded by $\frac{2^{n-2}}{\sqrt{n-2}}$ according to Stirling's approximation. As a consequence the worst-case complexity of the brute-force attack that aims at identifying the neighborhood of a specific node is $O(\frac{2^n}{\sqrt{n}})$. By extending this analysis to encompass all nodes, the resulting complexity is $O(2^n\sqrt{n})$. □

While this worst-case complexity is large enough to ensure the privacy of the distributed graph it does not take into account the structure of the graph. Indeed, we argue that the best case is when the graph to recover is either a clique (*i.e.*, a graph in which all nodes are connected) or the opposite (*i.e.*, a graph in which all nodes are completely disconnected). In such an improbable scenario, a single prediction is enough to reconstruct the neighborhoods of the two nodes involved in the prediction, as they have all the other nodes as common neighbors in the case of a clique, and no common neighbor in the case of a completely disconnected graph. The adversary removes the two nodes instantiated after each run of the protocol and then re-runs the protocol with another two nodes. By repeating this process $\lceil \frac{n-2}{2} \rceil$ times, the adversary achieves the complete reconstruction of the graph. However, let us underline the fact that cliques and anti-cliques are very specific types of graphs, which are not very interesting to model as graphs in real-life scenarios. To go beyond, some studies have shown that real-life graphs are often scale-invariant (*i.e.*, the distribution of node degrees follows the power law) [3, 5] implying that it is extremely rare in practice to have a clique to recover. Consequently, it may be more meaningful to consider an average-case complexity in this context in which we can assume a scale-invariant graph. Such a study goes beyond the scope of this paper and will be future work.

*4.3.2 Security of the caching-optimized protocol.* Our caching optimization is based on storing encryptions of previously seen nodes. For each node $x$ involved in a link prediction, party $P_1$ stores the final encryptions of its neighbors $\{g^{\alpha\beta x_i}\}_{i \in \{1,\ldots,|\Gamma(x)|\}}$. However, this optimization might also induce some additional leakage of the private graph, and give an advantage to $P_1$ in reconstructing the complete distributed graph. Imagine for instance that we would like to predict links between node $x$ and all the other nodes of the graph. If caching is used, $P_1$ will be storing encryptions of the neighbors of all the nodes in the graph. This means that $P_1$ is now able to additionally compute similarities between all pairs of nodes with the stored encryptions, not just between $x$ and the other nodes.

The same observation can be made from the partial encryptions $\{g^{\alpha x_i}\}_{i \in \{1,\ldots,|\Gamma_1(x)|\}}$ and $\{g^{\beta x_i}\}_{i \in \{1,\ldots,|\Gamma_2(x)|\}}$. By storing the partially encrypted neighbors from $P_2$, $P_1$ can compute the number of common neighbors of node pairs in $P_2$ that were not meant to be involved in the prediction. While this does not count as a leakage if the goal is to perform predictions between all node pairs (since the result of such an operation already gives the output of the previously explained attack), the problem still needs to be addressed for other use cases. To prevent this problem, refreshing the encryptions keys at each prediction is essential. We emphasize that this key changing process can be asymmetric : to prevent $P_1$ from computing the number of common neighbors of its nodes, $P_2$ can refresh its encryption key $\beta$ at each prediction without $P_1$ having to change theirs. This ensures that two encryptions $g^{\beta_1 x}, g^{\beta_2 x}$ of the same node $x$ under different keys $\beta_1, \beta_2$ are different, and therefore cannot be used to identify common neighbors across multiple predictions.

## 5 APPLICATION TO GRAPH SANITIZATION

Leveraging on our protocol, we can design a privacy-preserving defense mechanism against attempts to poison data in a GNN application. Hereafter, we provide experimental evidences of the effectiveness of this defense against state-of-the art graph attacks.

## 5.1 Description

Our approach acts as a preprocessing step by helping to privately clean the distributed graph before downstream learning tasks. The core idea of our approach is to identify suspicious or unlikely edges in the distributed graph based on the same approach as link prediction, which we refer to as *link removal*. These suspicious connections could be indicative of malicious intent as it has been shown in [38]. To obtain a likelihood score for each of the links on the joint network, we run Crypto'Graph for all the possible pairs of nodes in the network by computing the similarity measures introduced in Section 3.1. A threshold $t_i$ is then set by each party $i$, such that all links between pairs of nodes that have a similarity below $t_i$ are considered malicious and discarded from $G_i$.

## 5.2 Experimental evaluation

To assess the benefit of our collaborative approach over individual defense strategies, we evaluate its effectiveness against various types of attacks. We show how our distributed protocol can be leveraged to sanitize graphs before training graph neural networks for graph classification. We consider targeted attacks that aim at changing the class of specific nodes in the graph as well as global attacks that try to decrease the global classification accuracy over all the nodes. More precisely, we measure the performance of our defense against the IG-FGSM [38], Nettack [43] and Dice [44] attacks. The FGSM attack, a targeted attack traditionally applied to continuous image data, has been adapted to the discrete graph context by Wu and collaborators with the use of integrated gradient, hence the name IG-FGSM. Zugner and colleagues have proposed Nettack, another targeted attack using gradients to identify high-impact links and maliciously inject them in the neighborhood of an attacked node. Finally, Dice has been introduced in [44] as a baseline global attack, which simply randomly creates links between nodes belonging to different classes while removing links between nodes of the same class. Since Graph Convolutional Networks (GCNs) have been shown to perform quite well for node classification and given the usual two-layer network used in related papers [19, 43], we choose a Graph Convolutional Network with a single hidden layer as our learning architecture.

Figure 3 presents the experimental pipeline used in this section. Our experiments start by creating two subgraphs from the Polblogs dataset, as described in Section 4.2. Afterwards, we poison $G_1$ and $G_2$ using the previous attacks. Since Nettack and IG-FGSM are targeted attacks, we select 20 nodes that will undergo these attacks while this is not needed for Dice. Finally, we apply the three different defense strategies on the graphs obtained. Given graphs $G_{i\in\{1,2\}}$, we have identified the following key parameters that might influence the outcome of the defense mechanisms and subsequently the accuracy of the GCNs trained on sanitized data :

- **The thresholds of similarity $t_1, t_2$ for link removal**. These parameters directly impact the number of links removed during the sanitization. Indeed a high threshold might induce a high false positive rate whereas a low threshold could leave malicious links in the graphs (*i.e.*, leading to false negatives). Note that for party $i$ for the Jaccard and cosine similarities, $t_i \in [0, 1]$, and $t_i \in \mathbb{N}$ for the common neighbors similarity.

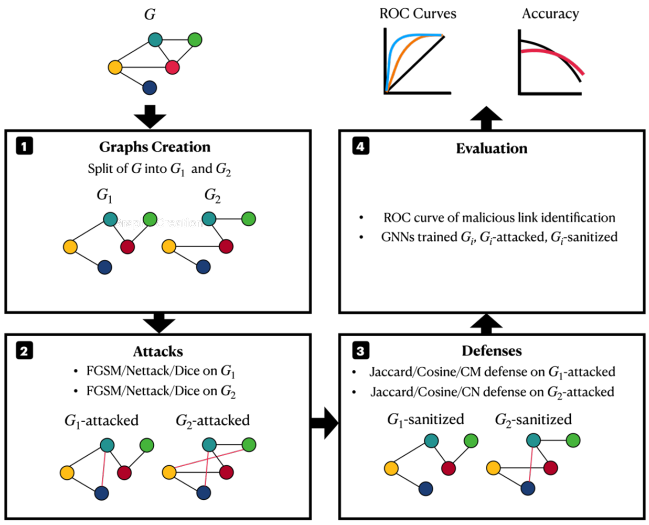



**Figure 3: Experimental pipeline for the application of Crypto'Graph to graph sanitization. At the end of the pipeline, we train simultaneously two GCNs on the different versions of $G_1$ and $G_2$ given through the pipeline.**

- **The common proportion of links *ppt* in the two graphs**. Since the graphs can have overlapping knowledge about the global network, our assumption is that the less they share, the more a collaborative defense is effective.
- **The perturbation rates $r_1$, $r_2$ of the attacks**. This factor influences how many malicious links are introduced by the attacks. More precisely, given the graph $G_i$, a perturbation rate of $r_i$ on a certain node $x$ means that the attack is allowed to add $r_i \times d(x)$ links to node $x$, in which $d(x)$ is the degree of node $x$. In contrast, a $r_i$-Dice attack on $G_i$ means that the attack modifies (adds or removes) exactly $r_i \times |\mathcal{E}_i|$ links in the entire graph. For this parameter, our hypothesis is that the more the graphs are attacked, the harder it becomes to recover from such perturbations.

We present two arguments that show the performance of our method for graph sanitization. First, we view our solution like a binary classifier, and study its statistical performance in identifying malicious links on a poisoned and distributed version of the Polblogs dataset. Then, we evaluate the performance of a node classification task on the same dataset in relation to different ranges of previous parameters. More precisely, we begin by exploring the impact of the similarity threshold, before evaluating the variation of the shared proportion of data and finishing with the experiments on the perturbation rate.

*5.2.1 Performance on Malicious Link Identification.* To demonstrate the protection of Crypto'Graph we start by giving an overview of the ability of our method to identify malicious links on a distributed graph. In this study, we set the common proportion of links *ppt* to 0.5, and the two graphs are attacked at rates $r_1 = r_2 = 0.5$ for IG-FGSM and Nettack, and $r_1 = r_2 = 0.1$ for Dice. We evaluate the performance of our solution with the aid of ROC curves showing the true positive rates and false positive rates obtained for the

different defense settings in their local and distributed versions, as shown in Figure 4. Each curve comes from the mean true positive rates and false positive rates for the two graphs.

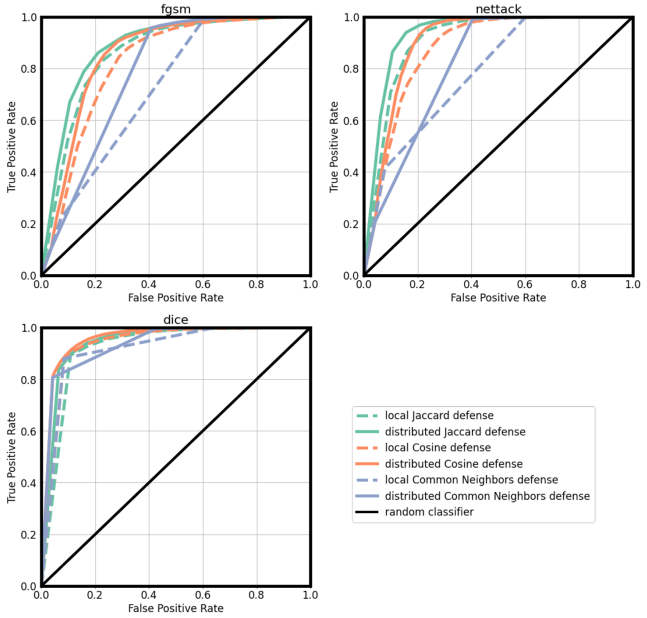



**Figure 4: Performance of Crypto'Graph for malicious link identification. The Jaccard, Cosine and Common Neighbors based defenses are applied to graphs attacked at rates $r_1 = r_2 = 0.5$ by the IG-FGSM and Nettack attacks, and $r_1 = r_2 = 0.1$ by the Dice attack.**

Those first results show the utility of our method in identifying malicious links on a distributed graph. We can clearly see how the distributed defenses outperform the local ones for most of the attacks. To further illustrate the improvement provided by our solution on a subsequent machine learning task, we explore in the following sections the impact of different settings of the previously identified parameters on the quality of our defense.

*5.2.2 Impact of Similarity Threshold for Defense.* In this section, we study the accuracy of GNNs trained on graphs after local and distributed defenses with different values of $t$. More precisely, in this series of experiments, the common proportion $ppt$ of links between $G_1$ and $G_2$ is set to 0.5 (*i.e.*, they both share 50% of the data and each own 25% of fresh and original data). The parameters are set to $r_1 = 0$ and $r_2 = 0.5$ for IG-FGSM and Nettack, while $r_1 = 0$ and $r_2 = 0.1$ for Dice. Figure 5 represents the evolution of the average accuracy of GNNs trained on $G_1$ and $G_2$ in three settings: when no attack and no defense have been performed, on the attacked graphs without defense and on the attacked and sanitized graphs with different defense mechanisms.

We summarize our findings on the impact of the similarity thresholds as follows:

- As expected, different thresholds lead to different graph qualities, which in turn induces varying performances for the trained GNNs. The same remark can be made for the similarity metric used for defense.
- The distributed defense mechanisms tend to be better than their local counterparts for most of the thresholds.
- In some situations, the defense mechanisms even allow for a better performance than on the clean graphs. We believe that this could be due to the fact that the defense removes the outliers from the data and that this helps for the GNNs tasks. This is an interesting finding that motivate the deployment of such defenses even when it is not clear if the graphs have been attacked.
- The Dice attack surprisingly improves the quality of the graphs. To understand this, one should remember that this attack is really simplistic, and thus might actually add absent but likely connections in the graphs, making them more useful for learning.

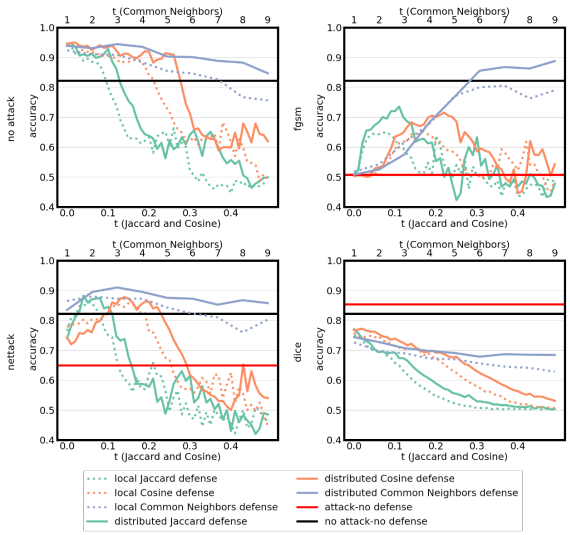



**Figure 5: Impact of the similarity threshold $t$ for link removal on the accuracy of GNNs trained on sanitized graphs. The Jaccard, Cosine and Common Neighbors based defenses are presented in the context of no attack, the IG-FGSM, Nettack as well as Dice attacks.**

*5.2.3 Impact of Shared Proportion of Links.* Since the global network can be distributed in many ways, we study the impact of the distribution of links over $G_1$ and $G_2$. Namely, we vary the proportion $ppt$ of common links owned by $G_1$ and $G_2$ such that $ppt$ ranges from 0 ($\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$) to 1 ($\mathcal{E}_1 = \mathcal{E}_2 = \mathcal{E}$). $t_1$ and $t_2$ are set to be the thresholds providing the best accuracy for each metric, $r_1 = 0$, $r_2 = 0.5$ for IG-FGSM and Nettack, $r_1 = 0$ and $r_2 = 0.1$ for Dice. As before, three settings are considered for the evaluation: no attack and no defense have been conducted, attack have been performed without subsequent defense and finally attack and defense have been deployed. This time again, the curves represent the average accuracy of the two models. From the results of this series of experiments presented in Figure 6, we derive the following observations:

- The distributed defense metrics stay consistently better than the local ones over the whole range of proportions.
- The quality of the local defenses grows with the proportion and gets near (sometimes exceeds) the performance of distributed defenses, meaning that the more each graph already shares links with the other, the more they can get from a local defense. This directly matches our initial assumption.
- Overall, the distributed defenses are more constant than the local ones across multiple instances of experimentation, which represents another advantage for them. Indeed in practice, one is more likely to choose a defense with a high and stable quality overtime.
- Again, the defense mechanisms (especially the distributed ones) make the graphs even better than if they were not attacked in the case of the IG-FGSM and Nettack attacks.
- As before, the Dice attack slightly improves the accuracy obtained on an attacked graph.

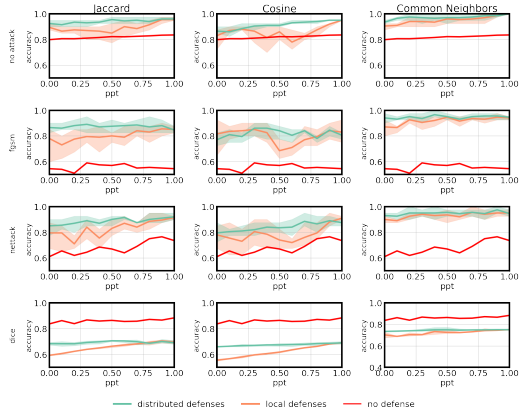



**Figure 6: Impact of the shared proportion (ppt) of links between $G_1$ and $G_2$ on the accuracy of the defense based on the different similarity metrics (Jaccard, Cosine and Common Neighbors), depending on the type of attack used (no attack, IG-FGSM, Nettack and Dice).**

*5.2.4 Impact of Perturbation Rates.* To demonstrate the performance of Crypto'Graph against different attack rates, we also evaluate the accuracy of GNNs trained on graphs sanitized after various forces of attack. To realize this, we study the combinations of three perturbation rates $r_i \in \{0, 0.5, 1\}$ for the IG-FGSM and Nettack attacks, and $r_i \in \{0, 0.1, 0.2\}$ for Dice. The common graph knowledge *ppt* is set to 0.5. Here, we show the accuracy gain of the distributed metrics over their local equivalents, with a positive margin meaning that the distributed metric is better that the local one, whereas a negative one favors the local metric. The results presented in Figure 7 lead us to the following conclusions:

- Overall, each graph can find a positive margin for each of the scenarios. This is especially important as the two graphs are not necessarily sanitized using the same defense metric, which leaves room for each graph owner to choose the metric that best suits them, or even combine several metrics for a better accuracy
- Often, the most attacked graph is the one with the lowest accuracy gain, which validates our assumption that high perturbation rates are more difficult to overcome.
- A perturbation rate of 1 is extreme, but in many of the reasonable scenarios, it appears that it is not too costly for the least attacked graph to cooperate and that it is always beneficial for the most attacked one to do so.

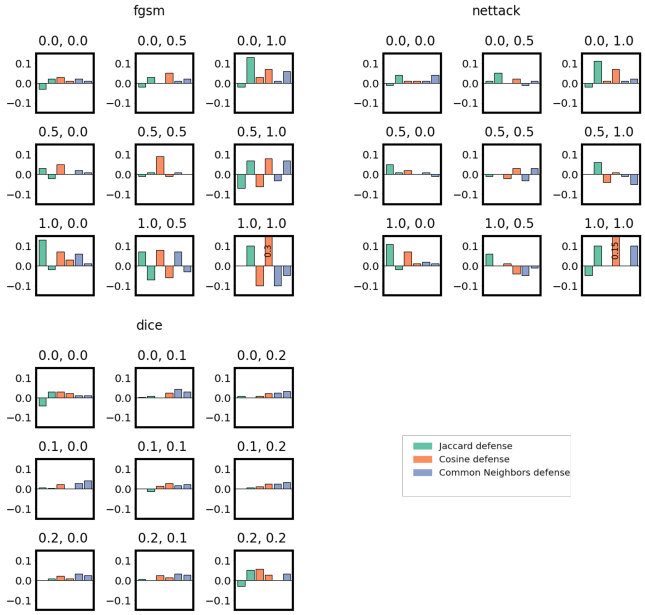
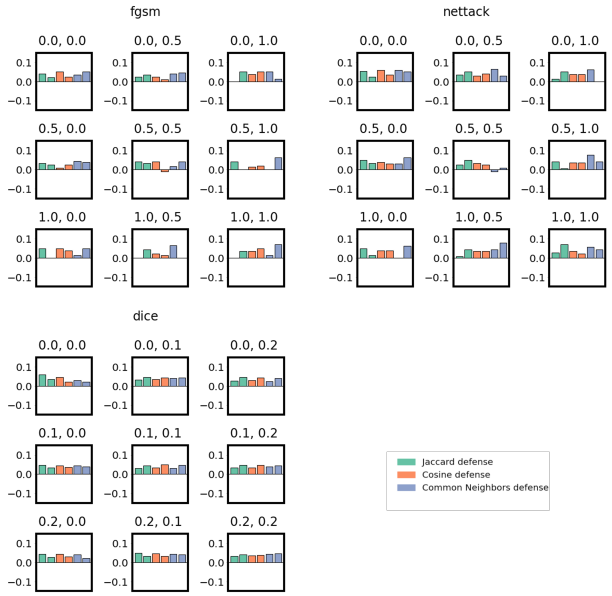



**Figure 7: Impact of the perturbation rate of the attacks on the accuracy of GNNs trained on sanitized $G_1$ and $G_2$ based on the Polblogs dataset. The amount of perturbation is denoted as $(r_1, r_2)$ on top of histograms. The first and second bar of each metric (*i.e,* each color) represents the accuracy gain for $G_1$ and $G_2$.**

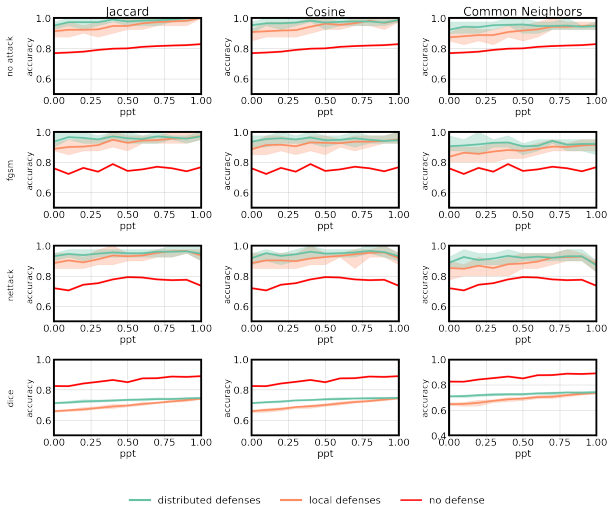
To support the genericity of our claim that using a distributed defense is better than a local one, we argue with the results obtained on the Cora dataset in Figure 8 that shows the same pattern as what we observed with Polblogs. We use a version of the dataset representing 2708 scientific papers as nodes, and 5278 citations as edges.

## 6 CONCLUSION

In this article, we have proposed Crypto'Graph, a protocol for privacy-preserving distributed link prediction. Crypto'Graph is more efficient than the other state-of-the-art methods both in terms of computation and communication, by one to several orders of magnitude, while reaching exactly the same utility. Additionally, our protocol is able to compute different similarity metrics, allowing for data owners to choose the best one according to their specific needs. We also demonstrate that Crypto'Graph is secure against external eavesdroppers and against honest-but-curious participants. Based on Crypto'Graph, we build a distributed defense mechanism against data poisoning in graph neural networks application scenarios. Our




(a) The impact of attacks' perturbation rates on GNN accuracy when trained on sanitized graphs $G_1$ and $G_2$.




(b) The impact of proportion of shared link between $G_1$ and $G_2$ with respect to the different similarity metrics across the attacks.

**Figure 8: Results on the Cora dataset.**

experiments show that this mechanism is effective to mitigate those attacks and can even be beneficial in the absence of attack. We also show that the more disjoint the data of the participants is, the more beneficial it is for them to cooperate via our distributed defense mechanism. Those benefits vary according to the power of the data poisoning attack. In reasonable attack scenarios, cooperation is a good strategy while it is a little more complex in extreme ones.

As a defense against the graph reconstruction attack presented above (which we have shown to be difficult to carry in the worst case in our security analysis), we consider as future works the application of methods like the occasional injection of dummy links in the graphs or an adaptation of differential privacy in our context. We would also like to propose a method for a private and efficient choice of the defense threshold, which we assume known by each of the parties in our current solution. In another direction, we would like to better characterize the security of our method by exploring more in-depth attack strategies and better estimating the difficulty of a brute-force graph reconstruction attack in the average case. Assuming malicious protocol participants and making our solution secure against such parties is also a future work that we consider. Finally, an interesting avenue would be to combine several similarity metrics to better counter graph poisoning attacks.

## ACKNOWLEDGMENTS

This work is supported by the DEEL Project CRDPJ 537462-18 funded by the National Science and Engineering Research Council of Canada (NSERC) and the Consortium for Research and Innovation in Aerospace in Québec (CRIAQ), together with its industrial partners Thales Canada inc, Bell Textron Canada Limited, CAE inc and Bombardier inc[3]. Sébastien Gambs is also supported by the Canada Research Chair program and a Discovery Grant from NSERC while Marc-Olivier Killijian is supported by a Discovery Grant from NSERC.

## REFERENCES

[1] *SEC 2: Recommended Elliptic Curve Domain Parameters*, 2010.
[2] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003. URL: https://www.sciencedirect.com/science/article/pii/S0378873303000091, doi:https://doi.org/10.1016/S0378-8733(03)00009-1.
[3] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.
[4] Dario Antweiler, David Sessler, Maxim Rossknecht, Benjamin Abb, Sebastian Ginzel, and Jörn Kohlhammer. Uncovering chains of infections through spatio-temporal and visual analysis of covid-19 contact traces. *Computers & Graphics*, 106:1–8, 2022.
[5] Igor Artico, I Smolyarenko, Veronica Vinciotti, and Ernst C Wit. How rare are power-law networks really? *Proceedings of the Royal Society A*, 476(2241):20190742, 2020.
[6] Md. Momin Al Aziz, Dima Alhadidi, and Noman Mohammed. Secure approximation of edit distance on genomic data. *BMC Medical Genomics*, 10, 07 2017. doi:10.1186/s12920-017-0279-9.
[7] Bellare, Namprempre, Pointcheval, and Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16:185–215, 2003.
[8] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *International Conference on Cryptology and Network Security*, pages 218–231. Springer, 2012.
[9] Sumit Kumar Debnath and Ratna Dutta. Provably secure fair mutual private set intersection cardinality utilizing bloom filter. In Kefei Chen, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology*, pages 505–525, Cham, 2017. Springer International Publishing.
[10] Sumit Kumar Debnath, Pantelimon Stănică, Tanmay Choudhury, and Nibedita Kundu. Post-quantum protocol for computing set intersection cardinality with linear complexity. *IET Information Security*, 14(6):661–669, 2020. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-ifs.2019.0315, arXiv:

[3] https://deel.quebec

https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-ifs.2019.0315, doi:https://doi.org/10.1049/iet-ifs.2019.0315.

[11] Didem Demirag, Mina Namazi, Erman Ayday, and Jeremy Clark. Privacy-preserving link prediction. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, and Nicola Dragoni, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 35–50, Cham, 2023. Springer International Publishing.

[12] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[13] Changyu Dong and Grigorios Loukides. Approximating private set union/intersection cardinality with logarithmic complexity. *IEEE Transactions on Information Forensics and Security*, 12(11):2792–2806, 2017. doi:10.1109/TIFS.2017.2721360.

[14] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 1–19, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[15] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2939672.2939754.

[16] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 370–389, 2020. doi:10.1109/EuroSP48549.2020.00031.

[17] Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 1901.

[18] Hisashi Kashima and Naoki Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 340–349, 2006. doi:10.1109/ICDM.2006.8.

[19] Thomas N. Kipf and Max Welling. 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*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[20] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/956863.956972.

[21] Dongxiao Liu, Jianbing Ni, Hongwei Li, Xiaodong Lin, and Xuemin Shen. Efficient and privacy-preserving ad conversion for v2x-assisted proximity marketing. In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 10–18, 2018. doi:10.1109/MASS.2018.00014.

[22] Siyi Lv, Jinhui Ye, Sijie Yin, Xiaochun Cheng, Chen Feng, Xiaoyan Liu, Rui Li, Zhaohui Li, Zheli Liu, and Li Zhou. Unbalanced private set intersection cardinality protocol with low communication cost. *Future Generation Computer Systems*, 102:1054–1061, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0167739X19316413, doi:https://doi.org/10.1016/j.future.2019.09.022.

[23] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL: http://arxiv.org/abs/1301.3781.

[24] Atsuko Miyaji, Kazuhisa Nakasho, and Shohei Nishida. Privacy-preserving integration of medical data: a practical multiparty private set intersection. *Journal of medical systems*, 41:1–10, 2017.

[25] Daniel Morales, Isaac Agudo, and Javier Lopez. Private set intersection: A systematic literature review. *Computer Science Review*, 49:100567, 2023. URL: https://www.sciencedirect.com/science/article/pii/S1574013723000345, doi:https://doi.org/10.1016/j.cosrev.2023.100567.

[26] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *19th USENIX Security Symposium (USENIX Security 10)*, Washington, DC, August 2010. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity10/botgrep-finding-p2p-bots-structured-graph-analysis.

[27] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64:025102, Jul 2001. doi:10.1103/PhysRevE.64.025102.

[28] Kenta Nomura, Yoshiaki Shiraishi, Masami Mohri, and Masakatu Morii. Secure association rule mining on vertically partitioned data using private-set intersection. *IEEE Access*, 8:144458–144467, 2020. doi:10.1109/ACCESS.2020.3014330.

[29] Georgios A Pavlopoulos, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G Bagos. Using graph theory to analyze biological networks. *BioData mining*, 4:1–27, 2011.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2623330.2623732.

[31] Amanda Cristina Davi Resende and Diego F. Aranha. Faster unbalanced private set intersection. In *Financial Cryptography*, volume 10957, pages 203–221. Springer, 2018.

[32] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL: https://networkrepository.com.

[33] Ou Ruan, Zihao Wang, Jing Mi, and Mingwu Zhang. New approach to set representation and practical private set-intersection protocols. *IEEE Access*, 7:64897–64906, 2019. doi:10.1109/ACCESS.2019.2917057.

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi:10.1109/TNN.2008.2005605.

[35] Liyan Shen, Xiaojun Chen, Dakui Wang, Binxing Fang, and Ye Dong. Efficient and private set intersection of human genomes. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 761–764, 2018. doi:10.1109/BIBM.2018.8621291.

[36] Amit Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2001.

[37] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, page 205–218, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1519065.1519089.

[38] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4816–4823. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi:10.24963/ijcai.2019/669.

[39] Xiaojun Xu, Hanzhang Wang, Alok Lal, Carl A. Gunter, and Bo Li. Edog: Adversarial edge detection for graph neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 291–305, 2023. doi:10.1109/SaTML54575.2023.00027.

[40] Hai-Feng Zhang, Xiao-Jing Ma, Jing Wang, Xingyi Zhang, Donghui Pan, and Kai Zhong. Privacy-preserving link prediction in multiple private networks. *IEEE Transactions on Computational Social Systems*, 10(2):538–550, 2023. doi:10.1109/TCSS.2022.3168010.

[41] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5171–5181, Red Hook, NY, USA, 2018. Curran Associates Inc.

[42] Yao Zheng, Bing Wang, Wenjing Lou, and Y. Thomas Hou. Privacy-preserving link prediction in decentralized online social networks. In Günther Pernul, Peter Y A Ryan, and Edgar Weippl, editors, *Computer Security – ESORICS 2015*, pages 61–80, Cham, 2015. Springer International Publishing.

[43] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2847–2856, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3219819.3220078.

[44] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=Bylnx209YX.