

A User-Centric Mechanism for Sequentially Releasing Graph Datasets under Blowfish Privacy

ELIE CHICHA, TICKET Lab., Antonine University, Hadat-Baabda, Lebanon and LIUPPA Lab., University of Pau & Pays Adour, Anglet, France

BECHARA AL BOUNA, TICKET Lab., Antonine University, Hadat-Baabda, Lebanon

MOHAMED NASSAR, Computer Science Department, American University of Beirut, Lebanon

RICHARD CHBEIR, LIUPPA Lab., University of Pau & Pays Adour, Anglet, France

RAMZI A. HARATY, Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon

MOURAD OUSSALAH, Center for Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering, University of Oulu, FI90014, Finland

DJAMAL BENSLIMANE, Université Claude Bernard Lyon 1, LIRIS, Villeurbanne, France

MANSOUR NASER ALRAJA, Department of Management Information Systems, College of Commerce and Business Administration, Dhofar University, Oman

In this article, we present a privacy-preserving technique for user-centric multi-release graphs. Our technique consists of sequentially releasing anonymized versions of these graphs under Blowfish Privacy. To do so, we introduce a graph model that is augmented with a time dimension and sampled at discrete time steps. We show that the direct application of state-of-the-art privacy-preserving Differential Private techniques is weak against background knowledge attacker models. We present different scenarios where randomizing separate releases independently is vulnerable to correlation attacks. Our method is inspired by Differential Privacy (DP) and its extension Blowfish Privacy (BP). To validate it, we show its effectiveness as well as its utility by experimental simulations.

The authors acknowledge the National Council for Scientific Research of Lebanon (CNRS-L) and Univ. Pau & Pays Adour, UPPA - E2S, LIUPPA for granting a doctoral fellowship to Elie Chicha. This research work is also supported by the Lebanese American University - Beirut, the Research Council (TRC), Sultanate of Oman (Block Fund-Research Grant).

Authors' addresses: E. Chicha, TICKET Lab., Antonine University, Hadat-Baabda, Lebanon and LIUPPA Lab., University of Pau & Pays Adour, Anglet, France; email: elie.chicha@ua.edu.lb; B. A. Bouna, TICKET Lab., Antonine University, Hadat-Baabda, Lebanon; email: bechara.albouna@ua.edu.lb; M. Nassar, Computer Science Department, American University of Beirut, Lebanon; email: mn115@aub.edu.lb; R. Chbeir, LIUPPA Lab., University of Pau & Pays Adour, Anglet, France; email: richard.chbeir@univ-pau.fr; R. A. Haraty, Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon; email: rharaty@lau.edu.lb; M. Oussalah, Center for Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering, University of Oulu, FI90014, Finland; email: Mourad.Oussalah@oulu.fi; D. Benslimane, Université Claude Bernard Lyon 1, LIRIS, Villeurbanne, France; email: djamal.benslimane@univ-lyon1.fr; M. N. Alraja, Department of Management Information Systems, College of Commerce and Business Administration, Dhofar University, Oman; email: malraja@du.edu.om.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

© 2021 Association for Computing Machinery.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; **Social aspects of security and privacy**; **Privacy protections**;

Additional Key Words and Phrases: Call detail records, graph anonymization, differential privacy, Blowfish Privacy, multi-release datasets

ACM Reference format:

Elie Chicha, Bechara Al Bouna, Mohamed Nassar, Richard Chbeir, Ramzi A. Haraty, Mourad Oussalah, Djamel Benslimane, and Mansour Naser Alraja. 2021. A User-Centric Mechanism for Sequentially Releasing Graph Datasets under Blowfish Privacy. *ACM Trans. Internet Technol.* 21, 1, Article 20, 25 pages.

1 INTRODUCTION

Nowadays, data sharing is growing in popularity. It is shaping innovative business models and creating new marketplaces. Ad-based business models as Social network companies (Facebook, Twitter, Instagram, etc.) need user-centric designs to grow. Even companies that do not rely totally on an ad-based model may also benefit from their users' data to make extra money like telecommunications corporations, video-on-demand service providers, and so on. Then, a user-centric design benefits from its users and their data to make profits. While it is bringing benefits to data providers, at the same time, it is putting a significant risk on the privacy of the data owners. This risk appears because the shared data contains sensitive information and needs to be appropriately protected. Several privacy breach scenarios have been widely cited, notably, the leaks of the medical records of the governor of Massachusetts [37, 52] and the AOL search data in 2006 [4], which are typical examples of privacy breaches caused by inappropriate protection of data.

Several techniques were proposed in the literature to protect various types of data. In k -anonymity [49], the values of the quasi-identifier attributes of the tuples are suppressed or generalized until each tuple is identical with at least $(k - 1)$ other tuples on their quasi-identifier attributes. In l -diversity [38], a group of tuples is considered l -diverse if it contains at least l "well represented" values for the sensitive attribute. A table is l -diverse if every group is l -diverse. In t -closeness [31], the distribution of sensitive attributes in any group is close to its distribution in the full population. The distance between the distribution in a group and the population distribution should not exceed a distance of t . Differential Privacy (DP) [43] provides a mathematically provable guarantee that, whether or not, an individual's private information is included in the input of any DP algorithm, the output will lead to the same assumption about this individual's private information. In an attempt to fortify an individual's privacy, DP [16] has been proposed and has since garnered much attention among the privacy policymakers. DP is a privacy definition that aims to ensure a tradeoff between privacy and utility by adding a small amount of noise enough to hide the adding or dropping of an individual from the database.

Current DP mechanisms have been applied on a variety of data structures such as images [13], location data [3, 24, 56], set-valued data [10, 17], relational data [28], and graph-based datasets [2, 32, 48, 55] (representing for instance social networks interactions and call detail records). In these graphs, vertices represent individuals, sometimes annotated with meta-information. Edges represent interactions among users and can be labeled as well. Applying the DP mechanism on graph datasets is done by adding noise to the edges, as in References [7, 20, 21, 25, 45, 48, 54] or to the nodes [11, 14, 26] to prevent identity and link disclosure while keeping the dataset suitable for analysis. For example, in Figure 1, if a background knowledge determines three people calling each other; thus, we can identify them in the graph as the triangle (1-2-4). With some sensitive data, we can recognize each of them. However, by adding some noisy edges, we increase the uncertainty and make it harder for the adversary to relate their background knowledge to the triangle (1-2-4).

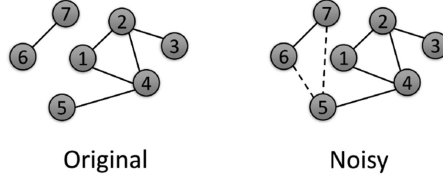


Fig. 1. Original and edge-Differentially Private Graphs.

Table 1. Sensitive Data Associated to the Nodes

Vertex	Age	Gender	Marital Status	Political View
1	30–40	Male	Divorced	Democratic
2	10–20	Male	Single	Republican
4	20–30	Female	Married	Republican

by creating another one (5-6-7). By releasing noisy data, the aim of DP is preventing adversary with a background knowledge from breaching the privacy of individuals that exists or not in a database. However, Kifer et al. [27] have proved that an adversary with a background knowledge can still disclose sensitive information from the data that induces correlations across tuples even if DP is applied.

This article shows a new user-centric design that relies on sharing or publishing subsequent graph versions representing interactions between individuals at subsequent time frames instead of just one graph. The limitation of DP proved by Kifer et al. [27] can appear in this kind of data. Since anonymizing each release independently will not be sufficient, an attacker can still infer information about the individuals by combining the anonymized released graphs together. Here, we show how a DP mechanism can be extended to address the multi-release graph anonymization.

To illustrate this scenario in an example, let us consider that we record the communications inside our university community for some statistics and social studies. Let us say that the graphs in Figure 2 are the records of three days: G_1 for the first day, G_2 for the second, and G_3 for the third. However, by sharing these graphs as they are, a real threat privacy could arise even if no identifiers are associated with the vertices. For example, if the shared graphs fall into the hands of an adversary possessing a background knowledge such as as Doctor Bob and his two students, Alice and John, who all call each other every day, by intersecting the three graphs G_1 , G_2 , and G_3 as in Figure 2 and by projecting the background knowledge to this intersection, it becomes easy to know that vertices 1, 2, and 4 represent Doctor Bob and his two students.

If the sensitive data associated with these nodes are as mentioned in Table 1, then, it is effortless to guess that the vertices 1, 2, and 4 represent Bob, John, and Alice, respectively. In this way, the adversary can discover the three individuals' marital status and political view, which presents a serious privacy breach. The situation does not improve in Figure 3 despite applying a DP algorithm to add edges in each graph. These noisy edges cannot change the fact that just one triangle appears in the Intersection graph. If we even use a privacy-preserving technique to protect the users' identity in the three graphs, then the adversary can still connect the persons in his/her background knowledge with nodes 1, 2, and 4. Here, we can define the background knowledge as a list of tuples representing interactions between users and labeled with timestamps. $(u_i, u_j, [t_s, t_e])$ indicates that user i has communicated with user j in the time frame $[t_s, t_e]$. For example: (Bob, Alice, Day1); (Bob, John, Day1); and so on. This background knowledge could be improved in a way that every tuple represents the interactions between many users instead of just two, which can be projected on the graph as a subgraph instead of just an edge: $((u_i, u_j), (u_i, u_z), (u_j, u_z), [t_s, t_e])$ indicating that users



Fig. 2. Graphs at disjoint time frame.

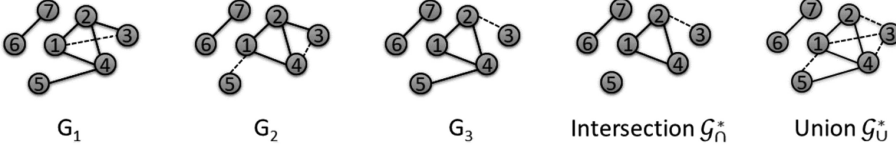


Fig. 3. Graphs anonymized by an edge-DP mechanism.

i, j and z have interacted with each other in the time frame $[t_s, t_e]$. For example, ((Bob, Alice), (Bob, John), (Alice, John), Day1).

We intend to solve a paradox, since we need the data to still carry out valid information about the population represented by the graphs without exposing individuals' privacy. We can define the utility of the released data as one or more statistical measures that the data user can compute with a certain degree of confidence. We propose here a new anonymization technique for a dataset composed of sequential dependent released graphs based on a relaxed version of Blowfish Privacy (BP) [23]. Our goal is to propose a mechanism that protects the correlated communications between users in sequentially released graphs. We can identify these communications as subgraphs. The subgraphs' privacy relies on the fact that the mechanism's output is independent of the original existence or absence of these particular subgraphs. Our mechanism applies reliable protection for these subgraphs by manipulating their existence or absence in each graph. We notice that implementing these manipulations is not simple and requires a fine-tuning about the subgraphs' privacy. We prove that our mechanism is under a relaxed version of BP [23]. BP has two intentions: from the privacy side, it presents a solution to correlated data where DP fails to protect data. In contrast, from the utility side, it allows to unveil more about the individuals without risking their privacy. In our solution, we are applying the non-interactive approach of BP. Thus, the BP mechanism returns a noisy data set instead of a query result. The BP mechanism will generate the noisy version of the released graphs [41] under a privacy guarantee called (ϵ, δ) -BP.

The rest of the article is organized as follows. In Section 3, we present the main related work in the literature. Section 2 introduces DP and BP backgrounds. In Section 4, we discuss our data model. We propose a new anonymization mechanism in Section 5, and we discuss its privacy guarantee. In Section 5.1, we also discuss how to implement this mechanism in the graphs and how the implementation affects privacy. Then, we present our algorithm to apply the mechanism and the implementation in Section 6. Section 7 presents conducted experiments to validate our approach. Finally, Section 8 concludes the article and draws some future directions.

2 BACKGROUND

2.1 Differential Privacy

Before discussing our mechanism, we introduce some basic notions of DP. Dwork et al. [16] define DP as a privacy mechanism for a curator holding a data of individuals in a database D , where each row represents the data of a single individual, to provide a sanitized database that allows statistical analysis and simultaneously protecting the individual rows. Formally speaking, a

randomized mechanism M with domain $\mathbb{N}^{|\chi|}$ is (ϵ, δ) -Differential Private if for all $S \subset \text{Range}(M)$ and for all $D, D' \in \mathbb{N}^{|\chi|}$ such that $\|D - D'\|_1 \leq 1$:

$$\Pr[M(D) \in S] \leq e^\epsilon \cdot \Pr[M(D') \in S] + \delta, \quad (1)$$

where D and D' are two databases of records from a universe χ , which differ by just one row at most and e^ϵ is always greater or equal to 1 ($\epsilon \geq 0$). DP promises that the probability of harm for an individual to participate in a survey, for example, is not more significant than the probability of harm if she does not. By choosing $\delta = 0$, we ensure that, for every run, the randomized mechanism (M) returns the same results for D and D' with roughly the same probability. In other words, the probabilities that the output of $M(D)$ is in the range S and the probability that $M(D')$ is in the range S too are very close. This closeness is managed by a privacy factor ϵ . We can list two types of DP applied to graphs:

- Node-based: requires insensitivity to the addition or removal of any vertex (and all its edges) from the graph.
- Edge-based: requires insensitivity to the addition or deletion of any edge from the graph.

In our work, we create two probabilities represented in Lemma 1, and we prove that the added noise based on these probabilities respects the DP inequality.

2.2 Blowfish Privacy

BP is a generalization of DP. It relies on the same inequality but has a privacy policy P as an extension of DP. $P = (\mathcal{T}, G_S, I_Q)$, where \mathcal{T} denotes the domain of all possible tuples, $G_S = (V_S, e_S)$ is a discriminative secret graph with $V_S \subseteq \mathcal{T}$ and I_Q denotes the set of databases that are possible under the constraints Q . Let us say that in Figure 3 we have released noisy graphs that appear as if some professors or students are communicating from inside the university during some holidays where working is strictly forbidden, for tradition or religious reasons, or during a strike of the University Teachers Union. Then, releasing these graphs (showing calls between these phones in one of these days) may pose a problem for employees, students, or the university. Therefore, any Call Detail Records (CDR) graph that respects the privacy policy $P = (\mathcal{T}, G_S, I_Q)$ should not show any call between phones on these specific days.

While in DP, the discriminative secret graph G_S is always a complete graph, BP may allow the public to distinguish between specific tuples. Besides, DP does not have any constraint Q . Since BP is a strong privacy definition, a relaxation is needed in many cases to prove that a mechanism is relaxed Blowfish Private. The data owner provides the upper bound of this relaxation. Before releasing the graphs, we have to ensure that the noise is enough that the relaxation does not surpass the upper bound. The data owners also provide any background knowledge that they possess about their database. The service takes this knowledge as constraints and checks if the noisy graphs respect them. Therefore, in our study, any mechanism that does not consider these two types of constraints, does not respect BP. We will prove in Lemma 2 that our proposed mechanism respects the BP inequality, and we explain the BP privacy policy adopted by our mechanism in 1.

3 RELATED WORK

In this section, we outline some techniques and algorithms dedicated to protect the graph datasets. We divide them into four categories: (1) identity and link disclosure [15, 57, 60], (2) dK -graph generation model [12, 48, 54], (3) platforms and programming languages [40, 45, 46], and, finally, (4) static [32, 42], and dynamic graphs anonymization [5, 6, 57].

3.1 Identity and Link Disclosure

Graph data disclosure can be divided into three categories [34]: (1) *Identity*: the identity of an individual associated with a node is revealed; (2) *Link*: the sensitive relationships between two individuals are disclosed; and (3) *Content*: the sensitive data associated with each node is compromised. We can list three privacy definitions to encounter identity disclosure:

- *k*-Candidate Anonymity [22, 57]: An anonymized graph satisfies *k*-candidate anonymity if, for a given structural query, no individual can be identified with a probability higher than $\frac{1}{k}$.
- *k*-Degree Anonymity [35]: An anonymized graph satisfies *k*-degree anonymity if every node in the graph has the same degree with at least $(k - 1)$ other nodes.
- *k*-Neighborhood Anonymity [61]: A node is *k*-anonymous in a graph if there are at least $(k - 1)$ other nodes such that the subgraphs constructed by the neighbors of each node are all isomorphic. A graph satisfies *k*-neighborhood anonymity if all the nodes are *k*-anonymous as defined above.

Other approaches can also be listed for the link disclosure. In the Link Re-identification [60], edges are classified as either sensitive or observed. The goal is to minimize the probability of predicting sensitive edges based on the observed edges while keeping the number of observational edges removed small enough to preserve the utility. Another approach is Privacy-Preserving Link Analysis [15] to enable link analysis in dynamic graphs. The algorithms address privacy concerns by applying encryption. For Random Perturbation for Private Relationship Protection [57], two randomization techniques were studied. The first is based on adding then deleting edges randomly. The second one relies on switching two edges, e.g., deleting the two edges (v_1, v_2) and (v_3, v_4) , then adding the two edges (v_1, v_3) and (v_2, v_4) , where v_1, v_2, v_3 , and v_4 are nodes in the graph. The content disclosure is a significant problem, but, to the best of our knowledge, the literature does not consider the impact of graph structure on this category of disclosure.

It is worth noting that, in contradiction with anonymization techniques, many works proposed de-anonymization techniques. For example, Data for Development (D4D) is an innovation challenge for societal development. In Reference [50], the authors study one of the D4D datasets where the anonymization strategy is to disconnect users' ego nets up to depth 2. The ego nets are published to conceal the overall graph structure. The authors successfully propose two de-anonymization techniques for identifying 1-hop and 2-hop nodes in these ego nets. Furthermore, automated social graph de-anonymization using adversarial machine learning is proposed in Reference [51].

3.2 *dK*-graph Generation Model

Another type of DP mechanisms is based on the *dK*-graph generation model. In these mechanisms, various parameters are derived from the original graph. DP is applied on these parameters to create noisy versions, and, finally, new derived graphs are generated using a generation model. Chen et al. [12] present a method for publishing graphs under DP. They rely on a community-preserving generative model called CAGM. This model profits from some properties from the community as parameters to generate the graphs. Some differential private methods are applied to these properties to create the noisy parameters of CAGM. Another example is in References [48, 54], where the authors propose an edge-DP graph generation mechanism relying on the *dK*-graph model. The mechanism generates a noisy graph based on a set of properties in the single static original graph. Thus, this mechanism cannot ensure the privacy of individuals in dynamic or multi-released graphs.

3.3 Programming Languages and Platforms for Queries

Programming languages and platforms listed in this subsection provide the possibility of creating queries that guarantee noisy results respecting the DP requirements. McSherry [40] represents the Privacy Integrated Queries (PINQ) platform for differentially private data analysis, a privacy-preserving version of Language Integrated Queries (LINQ). This platform adopts the interactive approach; thus, the outputs are the noisy results of queries and not a noisy or synopsis dataset.

The wPINQ platform [46] generalizes PINQ to deal with weighted datasets. It improves the results on graph analysis and introduces new generalization by counting triangles with given degrees, for example. Note that in an interactive approach, a privacy budget always limits the number of possible queries. In contrast, in a non-interactive approach, the analyst can apply unlimited queries on the noisy graph. In Reference [45], the authors benefit from wPINQ to extract noisy properties of the secret graph then generate a seed synthetic graph that fits these properties by relying on Marko Chain Monte Carlo. The properties extraction is composed of two steps: Rescaling the resulting records' weights of the query, then aggregating the weighted secret records, adding noise, and exposing results. Nevertheless, nothing indicates that wPINQ is designed to protect correlated data like sequentially released graphs.

3.4 Anonymization of Static Graphs

Static graphs are those that are fixed once created. A plethora of work addresses the anonymization of static graphs. For example, Nguyen et al. [42] categorize the state-of-the-art into two main groups: direct publication schemes and model-based publication schemes. The authors introduce a new scheme, so-called Top-m-Filter (TmF), and improve an existing technique called EdgeFlip. Both of them exhibit consistent behavior with increasing privacy budgets while the model-based publication schemes do not. Li et al. [32] propose a graph-based framework for privacy-preserving data publication. They define how to transform a dataset to a graph $G = U \cup V \cup S, E$, where U , V , and S are the sets of vertices representing the users or ids, the quasi-identifiers, and the secrets, respectively. Edges E indicate a semantic relation (connecting two users) or correlation relation (connecting two vertices from different types). The authors propose a framework to preserve privacy against background knowledge based on Anonymity Operators for the addition and deletion of vertices and edges, and a graph partition to apply these operators on a member of subgraphs instead of the whole graph. This work does not consider the multi-released graph scenario, thus does not guarantee that their framework is robust against correlated data from multiple versions of a graph.

3.5 Anonymization of Dynamic Graphs

Dynamic graphs are those subjected to changes in their structures or the weights of their edges. Less work follows the direction of dynamic graphs. The authors in [36] rely on DP for anonymizing dynamic social networks. Their approach, named Dynamic Differential Privacy Algorithm (DDPA), adds Laplacian noise to edge weights. DDPA tracks the edge weight information across the graph iterations and adds the privacy protection budget. However, it does not consider altering the graph topology. Very similar to our work is Reference [53]. In this article, social network graphs representing a time series of the corresponding social network's evolution are anonymized to a sequence of sanitized graphs released for further analysis. We share the same view that naively applying the existing approaches to each time-series graph will breach privacy purposes. However, our assumptions are more restricted, since we assume that the attacker has external background knowledge about the graphs.

Li et al. [33] propose a privacy model k^m where k indicates the privacy level and m is a time period that an adversary can monitor a victim to collect the attack knowledge. A distributed algorithm is provided that adds nodes to the graph, then generates the noisy version. The distributed greedy merge noise node algorithm reduces the number of nodes added under satisfying the anonymous model. Qiuyang et al. [47] propose a dynamic algorithm that satisfies DP and protect social networks against attacks based on semantic information. They classify the original graph into several subgraphs according to some characteristics of nodes. The graph is represented as an adjacency matrix. Quad-tree is used to divide the dense area of each subgraph. DP noise is added to the tree's leaf nodes, and finally, the adjacency matrix is reconstructed and published.

Yue et al. [59] proposed local and global anonymity functions and a framework called APRI to apply sequential online anonymization on a set of graphs. They anonymize the degree of the node of the current graph locally, then they compare, via APRI using a Kolmogorov–Smirnov Test, the distribution of node degrees of the current graph, and the set of previously anonymized graphs. When the difference is equal or greater than a given threshold, they restart the anonymization process for this graph. They use the global anonymity function to ensure the similarity in the distribution of node degrees between all the anonymized graphs.

Mcwan et al. [39] propose a clustering algorithm to group at least k nodes into k clusters based on their connectivity and anonymize each cluster for every instance of the graph. The algorithm supports the addition of nodes in new instances. Each cluster contains the nodes with close connectivity, and these nodes in the anonymized instance of the graph are assigned with the same label. Also, Yu et al. [58] propose a grouping mechanism for the nodes based on their properties. The mechanism guarantees that without a background knowledge, an attacker's probability of identifying a node involved in any edge is at most $\frac{1}{k}$. Also, the probability that an attacker identifies an edge between two nodes is at most $\frac{1}{k}$. Therefore, the goal of the mechanism is to protect edges against attacks without background knowledge dynamically.

All these mechanisms for sequentially released graphs focus on the properties of nodes, especially degrees of nodes. Thus, two of them [39, 59] might protect dynamic graphs against a background knowledge. However, all of them do not aim to protect the dynamic graphs against the type of background knowledge containing information about the connections or the relation between two or more individuals. Returning to the example provided in the Section 1, the background knowledge could be that Alice and John call their Doctor Bob daily, especially in exams and project submissions. They do not usually communicate in summer, but we also know that Alice's birthday is on August 6 and Doctor Bob's birthday is on August 24. We doubt that it is possible to prove that the mechanisms proposed to deal with the background knowledge of nodes could deal with an adversary having our type of background knowledge and trying to project it into the graphs by searching for three nodes communicating in the time of academic year especially in the periods of exams and projects. Then, a period of no communication in summer interrupted by calls on August 6 and 24. Thus, our work is different from others by addressing the problem of facing this type of background knowledge and preserving an acceptable level of utility in the graph's released instances.

4 PRELIMINARY DEFINITIONS

Definition 1. A simple graph G is undirected and defined in a given time frame by a set of vertices and a set of edges: $G(V, E, [t_i, t_{i+1}])$ where:

- V is the set of vertices $V \subset \mathcal{V}$ representing all the users in the time frame $[t_i, t_{i+1}]$ (While \mathcal{V} represents all the vertices in all the versions).

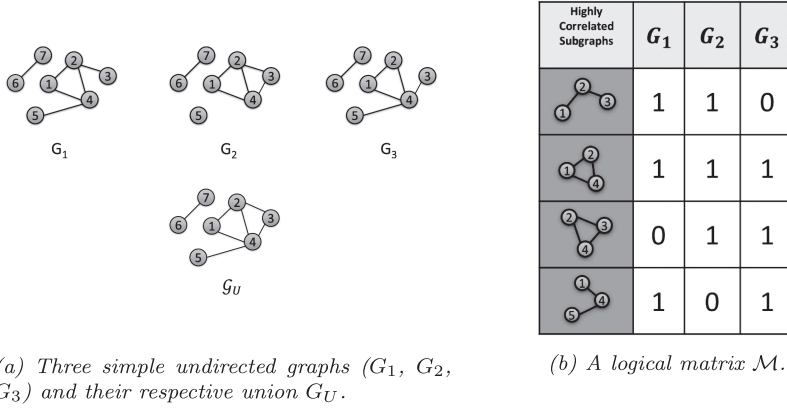


Fig. 4. An example of the Union graph \mathcal{G}_U and logical matrix.

- E is the set of edges $E \subset V \times V$ representing interactions between the users in V during the time frame $[t_i, t_{i+1}]$. We consider only two states: interaction and no interaction. An edge is assigned to two vertices (representing two individuals) if they are interacting during the i th time frame.
- $[t_i, t_{i+1}]$ is the i th time interval ($i = 0, 1, \dots, N$)

The definition can be used for the anonymized graph $G^*(V^*, E^*, [t_i, t_{i+1}])$.

For example, the first graph in Figure 4(a) is a simple undirected graph with a set of seven vertices, a set of six edges, and a time interval defined as Day 1.

Definition 2. Let \mathcal{G} be a set of graphs to be released in distinct time frames, $\mathcal{G} = \{G_1, \dots, G_n\}$. Let SG be a subgraph of \mathcal{G} where $\exists G \in \mathcal{G}$ such that $SG \subseteq G$. We define the logical matrix \mathcal{M} whose row and column indices indicate the elements of SG and \mathcal{G} . $\mathcal{M}[SG_i][G_j] = \mathcal{M}_{ij}$ represents the status of subgraph SG_i in the graph G_j , where

$$\mathcal{M}_{ij} = \begin{cases} 0 & \text{if } SG_i \not\subseteq G_j \\ 1 & \text{if } SG_i \subseteq G_j \end{cases}. \quad (2)$$

In what follows, we will use \mathcal{M}_{ij} to denote $\mathcal{M}[SG_i][G_j]$.

Several subgraphs were sampled from the Union graph in Figure 4(a) and were represented as the rows of the matrix in Figure 4(b). The columns of the matrix represent the graphs of the three days, and each element represents the existence of the subgraph in the graph.

The notations in this article are summarized in Table 2.

5 BP MECHANISM FOR SEQUENTIALLY RELEASING GRAPH DATASETS

DP provides reliable privacy in a single graph [11, 25]. However, it has severe limitations when graphs are sequentially released [27] in different time frames. The problem appears when the background knowledge can be linked to correlated or sensitive subgraphs in \mathcal{G} . Hence, the adversary will tie his/her background knowledge to a complete subgraph instead of separated edges or vertices in \mathcal{G} . Here, we propose a new layer of privacy-preserving approach. It is neither edge-DP nor node-DP, considering that we are not protecting separated edges or vertices but a combination of them in the form of subgraphs. We propose a robust privacy-preserving operation on the logical matrix \mathcal{M} represented by the graphs in \mathcal{G} and the correlated (or sensitive) sampled subgraphs.

Table 2. Notations and Descriptions

Symbol	Description	Symbol	Description
\mathcal{T}	Domain of all possible tuples	G_S	Discriminative secret graph
\mathcal{G}_\cup	Union graph of the set of graphs \mathcal{G}	G	Undirected simple graph
V	Set of nodes in G	E	Set of edges in G
$[t_i, t_{i+1}]$	i^{th} time interval	G^*	Anonymized version of G
\mathcal{G}	Set of graphs to be released	$\mathcal{M}^{n \times m}$	Logical matrix of size $n \times m$
\mathcal{M}^*	Anonymized version of \mathcal{M}	SG	Sampled subgraph from \mathcal{G}
f	Operation on logical matrix \mathcal{M}	S	Range of outputs
p, q	Probabilities	g	Operation on a set of graphs \mathcal{G}
δ'	Rate of subgraph in \mathcal{G} that does not reflect their values in \mathcal{M}^*	\mathcal{I}_Q	Set of databases that are possible under the constraints Q

1	1	0		1	1	0		1	1	0		1	1	0
1	1	1		1	1	1		0	1	1		1	1	0
0	1	0		0	1	0		0	0	1		0	1	0
1	0	1		1	1	1		1	1	1		1	0	1
M				M'				M''				M'''		

Fig. 5. Matrix M and three of its possible neighbors.

In this way, we can determine the noise as many subgraphs are suppressed or added from/to the graphs in \mathcal{G} .

Before we elaborate more about our privacy-preserving mechanism, we define first how two logical matrices M and M' can be considered as neighbors.

Definition 3 (Neighboring Matrices). We say that two matrices M and M' are neighbors if:

- (1) $M, M' \in \mathcal{I}_Q$, which means that both respect the constraints mentioned in Section 2.2.
- (2) M, M' differ by just one binary element: $\exists! i \in \{0, \dots, n\}, j \in \{0, \dots, m\} \mid M_{ij} \neq M'_{ij}$ where n is the number of graphs in the dataset and m is the number of sampled subgraphs.

For example, in Figure 5, we check the neighboring between M and the three other matrices. If we have the following constraint Q that “No column in any published matrix could be formed by just 1s,” then M and M' could not be neighbors, because the second column of M' violates the constraint, then $M' \notin \mathcal{I}_Q$. M and M'' are not neighbors, because they differ by more than one element, while M and M''' are neighbors, because both of them respects the constraint and they differ by just one element.

Figure 6 shows the input and the output of the privacy-preserving operation applied to the logical matrix M .

LEMMA 1. Let f be an operation on the logical matrix M , we say that $f(M)$ is Differential Private (or f is ϵ -DP) if it respects the inequality:

$$\Pr[f(M) = S] \leq e^\epsilon \times \Pr[f(M') = S],$$

where M and M' are two neighbors matrices, S is one possible output of the operation f , ϵ is a privacy parameter.

The output of $f(\mathcal{M})$ is a matrix \mathcal{M}^* related to \mathcal{M} by the equation

$$f(\mathcal{M}_{ij}) = \mathcal{M}_{ij}^* = \begin{cases} \overline{\mathcal{M}_{ij}} & \text{with probability } q = \frac{1}{e^\epsilon + 1} \\ \mathcal{M}_{ij} & \text{with probability } p = 1 - q = \frac{e^\epsilon}{e^\epsilon + 1} \end{cases}, \quad (3)$$

where $\overline{\mathcal{M}_{ij}}$ is the opposed binary value of \mathcal{M}_{ij} and $p > q$.

PROOF. Let us say that we have two neighboring matrices \mathcal{M} and \mathcal{M}' that differ by only one element $\mathcal{M}_{ij} \neq \mathcal{M}'_{ij}$. Let \mathcal{S} be an output matrix of the operation f .

$$\begin{aligned} \frac{Pr[f(\mathcal{M}) = \mathcal{S}]}{Pr[f(\mathcal{M}') = \mathcal{S}]} &= \frac{Pr[\mathcal{M}_{11} \rightarrow \mathcal{S}_{11}] \times \dots \times Pr[\mathcal{M}_{nm} \rightarrow \mathcal{S}_{nm}]}{Pr[\mathcal{M}'_{11} \rightarrow \mathcal{S}_{11}] \times \dots \times Pr[\mathcal{M}'_{nm} \rightarrow \mathcal{S}_{nm}]} \\ &= \frac{Pr[\mathcal{M}_{ij} \rightarrow \mathcal{S}_{ij}]}{Pr[\mathcal{M}'_{ij} \rightarrow \mathcal{S}_{ij}]} \end{aligned} \quad (4)$$

Then, to prove that f is ϵ -DP using the inequality $\frac{Pr[f(\mathcal{M})=\mathcal{S}]}{Pr[f(\mathcal{M}')=\mathcal{S}]} \leq e^\epsilon$, we have to prove that $\frac{Pr[\mathcal{M}_{ij} \rightarrow \mathcal{S}_{ij}]}{Pr[\mathcal{M}'_{ij} \rightarrow \mathcal{S}_{ij}]} \leq e^\epsilon$. Each of \mathcal{M}_{ij} , \mathcal{M}'_{ij} and \mathcal{S}_{ij} can have two values 0 and 1. In case, $\mathcal{M}_{ij} = \mathcal{M}'_{ij}$ then $\frac{Pr[\mathcal{M}_{ij} \rightarrow \mathcal{S}_{ij}]}{Pr[\mathcal{M}'_{ij} \rightarrow \mathcal{S}_{ij}]} = 1 \leq e^\epsilon$. The four other cases to compute are as follows:

- $\frac{Pr[1 \rightarrow 1]}{Pr[0 \rightarrow 1]} = \frac{p}{q} = \frac{\frac{e^\epsilon}{1+e^\epsilon}}{\frac{1}{1+e^\epsilon}} = e^\epsilon$
- $\frac{Pr[0 \rightarrow 1]}{Pr[1 \rightarrow 1]} = \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[1 \rightarrow 0]}{Pr[0 \rightarrow 0]} = \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[0 \rightarrow 0]}{Pr[1 \rightarrow 0]} = \frac{p}{q} = e^\epsilon$.

In the four cases, the operation respects the inequality, which proves that f is ϵ -DP. \square

5.1 Achieving the Mechanism f in \mathcal{G}

This section discusses how to populate the binary flips of the values in the logical matrix in their corresponding set of graphs \mathcal{G} . We also evaluate how these flips may affect the privacy of our proposed mechanism. In the implementation, these flips are achieved by suppressing or adding edges in \mathcal{G} . We notice that this might affect the ability of the mechanism to achieve DP. The flips made by the operation f have two types:

- $0 \rightarrow 1$: Here, the element in \mathcal{M} is 0 (which means that the subgraph does not exist in the corresponding original graph) but has been flipped to 1 in \mathcal{M}^* (the subgraph exists in the correspondent anonymized graph). All the missed edges of the subgraph should be added to perform this modification in the anonymized graph.
- $1 \rightarrow 0$: Here, the element in \mathcal{M} is 1 (which means that the subgraph exists in the corresponding original graph) but has been flipped to 0 in \mathcal{M}^* (the subgraph does not exist in the correspondent anonymized graph). One or more edges should be deleted from the subgraph to perform this modification in the anonymized graph.

Let g be the operation that implements the flips made by $f(\mathcal{M})$ in \mathcal{G} . g cannot guarantee a full projection of the flips in \mathcal{G} . A possible scenario to explain this drawback in privacy is when the same edge should remain in a subgraph while it should be deleted in another subgraph in the same graph. Then, it is obvious that it is not possible to implement in the graphs all the modifications done on the matrix. In Figure 7, for example, to perform the first flip, the operation g deletes the edge (1, 4). For the second flip, the operation adds the two edges (1, 4) and (1, 5); thus, the second





Sampled Subgraphs	G_1	G_2	G_3	G_1^*	G_2^*	G_3^*
	1	1	0	1	1	1
	1	1	1	1	1	1
	0	1	1	0	0	1
	1	0	1	1	1	0

Fig. 6. An anonymization operation applied on a logical matrix results a noisy logical matrix.

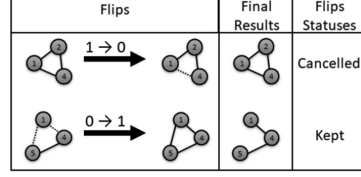


Fig. 7. Two flips performed by g where the first flip was cancelled by the second one.

flip cancels the effect of the first flip, and then the final status of the subgraph in the second flip is similar to its status before the flip. In this way, the privacy guarantee provided by f could be reduced when applying g on the graphs. In other words, after releasing \mathcal{G} , an adversary may create a matrix \mathcal{M}_{adv} that might represent the real status of the subgraphs more accurately than \mathcal{M}^* . That is the effect of g on the privacy guarantee. Thus, the implementation cannot provide the desired privacy guarantee provided by the operation of f . Still, it can offer a relaxed guarantee called (ϵ, δ) -BP, where ϵ is a privacy parameter provided by the data owner and δ is the relaxation parameter of the privacy definition.

In the next subsection, we will discuss in detail the privacy guarantee. However, this discussion cannot be reliable if we do not define the implementation steps first. As we have explained, the purpose of the implementation is to populate the flips in the graphs. We can list four types of these flips:

- $0 \rightarrow 0$: An element 0 in the matrix \mathcal{M} remains 0 in the noisy matrix \mathcal{M}^* .
- $0 \rightarrow 1$: An element 0 in the matrix \mathcal{M} is flipped to 1 in the noisy matrix \mathcal{M}^* .
- $1 \rightarrow 0$: An element 1 in the matrix \mathcal{M} is flipped to 0 in the noisy matrix \mathcal{M}^* .
- $1 \rightarrow 1$: An element 1 in the matrix \mathcal{M} remains 1 in the noisy matrix \mathcal{M}^* .

Because the implementation of a flip can cancel the effects of some previous implementations, we can assume that the ones performed initially have a higher probability of being canceled than those performed at the end. A random implementation of the flips will make impossible to compute the level of privacy provided by the implementation and to define the relaxation parameter δ . Therefore, two possible orders can be applied. If we are interested in preserving the subgraphs that should exist in \mathcal{G} , then the steps are as follows:

- (1) Remove at least one edge from each subgraph represented by 1 in \mathcal{M} and 0 in \mathcal{M}^* .
- (2) Ensure that all the subgraphs represented by 1 in \mathcal{M} and \mathcal{M}^* remain in the graphs after Step (1). In other words, if any edge removed in Step (1) leads to the removal of a subgraph represented by 1 in the matrix, this edge should be re-added.
- (3) Add all the missing edges for the subgraphs represented by 0 in \mathcal{M} and 1 in \mathcal{M}^* .

In this way, we guarantee that all the subgraphs represented by 1 in the matrix \mathcal{M}^* will exist in the released version of \mathcal{G} . Also, we may find in the set some subgraphs that should not exist. Whilst, if we are interested in guaranteeing that all the subgraphs represented by 0 in \mathcal{M}^* do not exist in the released version of \mathcal{G} , then the order need to be reversed as follows:

- (1) Add all the missing edges for the subgraphs represented by 0 in \mathcal{M} and 1 in \mathcal{M}^* .

- (2) Ensure that all the subgraphs represented by 0 in \mathcal{M} and \mathcal{M}^* do not exist in the graphs after Step (1).
- (3) Remove one or more edges from each subgraph represented by 1 in \mathcal{M} and 0 in \mathcal{M}^* .

By following this order, the set is clean from all undesirable subgraphs, but it may also miss some subgraphs that should appear in \mathcal{G} .

5.2 Toward a BP Mechanism

After explaining the operation f and its reliable privacy guarantee as well as the drawbacks of implementing g in the privacy domain, we will determine the implementation's privacy guarantee in this subsection.

LEMMA 2. Let δ' be the rate of subgraphs in \mathcal{G} that does not reflect their values in \mathcal{M}^* . Let \mathcal{G} , \mathcal{G}' and \mathcal{S} be three sets of graphs where \mathcal{G} and \mathcal{G}' are two neighboring sets that differ in just one sampled subgraph and \mathcal{S} is a possible output of g applied on \mathcal{G} and \mathcal{G}' . g respects the inequality

$$Pr[g(\mathcal{G}) = \mathcal{S}] \leq e^\epsilon \cdot Pr[g(\mathcal{G}') = \mathcal{S}] + \delta, \quad (5)$$

if

$$\delta' \leq \frac{\delta}{e^\epsilon - 1}. \quad (6)$$

PROOF. To prove the Lemma 2, we have to prove that:

$$\frac{Pr[g(\mathcal{G}) = \mathcal{S}] - \delta}{Pr[g(\mathcal{G}') = \mathcal{S}]} \leq e^\epsilon. \quad (7)$$

The left-hand side part of the inequality can be written in the following way:

$$\frac{Pr[g(\mathcal{M}_{11}) = \mathcal{S}_{11}] \times \cdots \times Pr[g(\mathcal{M}_{nm}) = \mathcal{S}_{nm}] - \delta}{Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times \cdots \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}]}.$$

Let SG_{dif} be the only subgraph that differs between \mathcal{G} and \mathcal{G}' . The status of SG_{dif} in \mathcal{G} , \mathcal{G}' , and \mathcal{S} can be indicated by the binary values \mathcal{M}_{ij} , \mathcal{M}'_{ij} , and \mathcal{S}_{ij} . Then, the above fraction can be reduced to

$$\frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}]}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} - \frac{\delta}{Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times \cdots \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}]}.$$

All the probabilities are less than or equal to 1. Thus,

$$Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times \cdots \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}] \leq Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}].$$

Consequently, we can assume that

$$\begin{aligned} & \frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}]}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} - \frac{\delta}{Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times \cdots \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}]} \leq \\ & \frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}]}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} - \frac{\delta}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]}, \end{aligned}$$

which leads to

$$\frac{Pr[g(\mathcal{G}) = \mathcal{S}] - \delta}{Pr[g(\mathcal{G}') = \mathcal{S}]} \leq \frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}] - \delta}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} \leq e^\epsilon.$$

Therefore, to prove the Inequality (7), it is enough to prove that

$$\frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}] - \delta}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} \leq e^\epsilon. \quad (8)$$

To continue our proof, we have to find all possible values of these probabilities. In this step of the proof, we can see the importance of the orders in the second stage. We will continue this proof based on the first order listed, but the second order will also lead to the same final result. We can list four possible values to the probabilities in Equation (8):

- Case 1: $Pr[1 \rightarrow 0]$ is the probability that f flips the binary value in the matrix. Nevertheless, because these flips are performed at first by the operation g , based on the first order, then there is a probability δ' that the following flip will cancel the current one, thus: $Pr[1 \rightarrow 0] = q - \delta'$.
- Case 2: $Pr[0 \rightarrow 0]$ is the probability that f does not flip the value. But some following flips performed by g may cause the appearance of the subgraph in \mathcal{G} , thus: $Pr[0 \rightarrow 0] = p - \delta'$.
- Case 3: $Pr[1 \rightarrow 1]$ is the probability that f does not flip the value. Based on the order of flips in g , all the upcoming flips will be performed by adding edges so that no flip can cause the drop of this subgraph from \mathcal{G} , thus: $Pr[1 \rightarrow 1] = p$.
- Case 4: $Pr[0 \rightarrow 1]$ is the probability that f flips the binary value in the matrix. In this case, too, no further flip could drop this subgraph, thus: $Pr[0 \rightarrow 1] = q$.

In this way, we have computed the four possible values for the probability. The next step is to show that by using any two of these values in Equation (8), the result remains lesser than e^ϵ :

- $\frac{Pr[1 \rightarrow 1] - \delta}{Pr[0 \rightarrow 1]} = \frac{p - \delta}{q} \leq \frac{p}{q} = e^\epsilon$
- $\frac{Pr[0 \rightarrow 1] - \delta}{Pr[1 \rightarrow 1]} = \frac{q - \delta}{p} \leq \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[1 \rightarrow 0] - \delta}{Pr[0 \rightarrow 0]} = \frac{q - \delta' - \delta}{p - \delta'} \leq \frac{q}{p}$ if:

$$\begin{aligned} p(q - \delta' - \delta) &\leq q(p - \delta') \\ p\delta' + p\delta &\geq q\delta' \\ \delta &\geq \delta' \times \frac{q - p}{p} \end{aligned} \tag{9}$$

$q - p \leq 0$ and $\delta \geq 0$, then Inequality (9) is always true, thus:

- $\frac{Pr[1 \rightarrow 0] - \delta}{Pr[0 \rightarrow 0]} \leq \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[0 \rightarrow 0] - \delta}{Pr[1 \rightarrow 0]} = \frac{p - \delta' - \delta}{q - \delta'} \leq \frac{p}{q}$ if

$$\begin{aligned} q(p - \delta' - \delta) &\leq p(q - \delta') \\ q\delta' + q\delta &\geq p\delta' \\ \delta &\geq \delta' \times \frac{p - q}{q}. \end{aligned}$$

Then, the only inequality that should be verified to ensure that \mathcal{G} can be released under (ϵ, δ) -BP is $\delta' \leq \delta \times \frac{q}{p - q}$. By substituting p and q by their values, we get the following:

$$\delta' \leq \frac{\delta}{e^\epsilon - 1}. \quad \square$$

THEOREM 1. *Let g be the operation that implements the flips made by $f(\mathcal{M})$ in \mathcal{G} , we say that g is (ϵ, δ) -BP if there exists a policy $P = (\mathcal{T}, G_S, I_Q)$ that applies on \mathcal{G} and composed of the Universe \mathcal{T} , the discriminative secret graph G_S and I_Q that denotes all the possible \mathcal{G} under the constraints Q .*

PROOF. \mathcal{T} contains all the possible graphs from the anonymized version of the set \mathcal{G} . $G_S = (V_S, E_S)$ is a discriminative secret graph where $V_S = \mathcal{T}$ and $e_S \in E_S$ connects two vertices that should be indistinguishable for the public. In this work, e_S will connect any two vertices of G_S

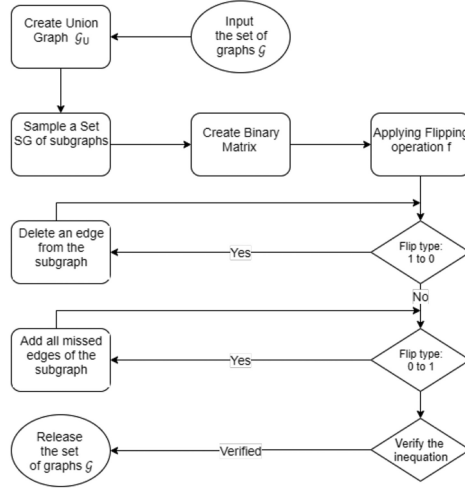


Fig. 8. Process diagram.

that represent two graphs that differ by just one subgraph sampled in our mechanism. Two graphs that differ by one or more edges but resemble in all the sampled subgraphs do not form a discriminative pair. They might be distinguishable, and the two vertices representing them in G_S are not connected by an edge e_S . Therefore, the discriminative secret graph G_S is not complete. In other words, E_S is the set of edges that connects two vertices representing a difference in the status of one or more sampled subgraphs. By proving that our mechanism's discriminative secret graph is not complete, we have demonstrated that our mechanism does not respect the DP definition requirements.

We can assume that our mechanism is under the Attribute version of BP. The Attribute Definition in BP describes a mechanism that aims to hide any changes made to any attribute in a set of tuples, which is literally what we aim to do by protecting any modifications done to the element of the matrix \mathcal{M} by manipulating the existence and non-existence of its corresponding subgraph in the released set of graphs.

Finally, the constraint Q is related to the Inequality 6: $\delta' \leq \frac{\delta}{e^{\epsilon}-1}$ beside any constraint provided by the data owner. Therefore, if \mathcal{G} respects this inequality, then $\mathcal{G} \subset \mathcal{I}_Q$. \square

By proving that g respects:

- the BP Inequality in Equation (5),
- a well-defined Policy P ,

we have proved that the mechanism is (ϵ, δ) -BP.

6 ANONYMIZATION ALGORITHM

This section details our mechanism by applying the Blowfish Privacy for sequentially releasing graphs discussed in the previous section.

The process is depicted in Figure 8. To note that we should possess all the graphs before starting the anonymization process. To proceed, we sample a number of subgraphs from a Union Graph \mathcal{G}_U , based on some criteria. For example, in our implementations, we will focus on the high correlated subgraphs. These are the subgraphs that exist in the most graphs in the set of graphs \mathcal{G} : a subgraph that exists in 90% of graphs has a much higher possibility of being sampled than a

subgraph that exists in 50% of the graphs. Based on these sampled subgraphs, we create a matrix \mathcal{M} , as in Figure 4(b), with binary attributes showing the status of each subgraph in each graph. Then, we apply the flipping probability to the values of the attributes. In this section, we will adapt the first order of implementation explained in Subsection 5.1; for this reason, all the $1 \rightarrow 0$ flips should be performed first by suppressing the edge that occurs the least in the sampled subgraphs to minimize the effect of this flip on other subgraphs. When all the $1 \rightarrow 0$ flips are performed, the $0 \rightarrow 1$ flips are applied by adding all the subgraph's missing edges. If all the required flips are completed, then we can start to release the noisy graphs.

In summary, our proposed mechanism is composed of these steps:

- (1) Sampling a number of subgraphs of K vertices.
- (2) Creating a matrix that represents the existence of the sampled subgraphs in each graph of the set.
- (3) Applying the operation f by anonymizing the matrix under the requirements of Differential Privacy.
- (4) Applying the operation g by adding/suppressing edges in the set of graphs.

In the following subsection, we will propose an algorithm to apply the operation and the implementation.

6.1 Applying the Algorithm

The list of sampled subgraphs is converted into a matrix \mathcal{M} , as explained in the Definition 2. \mathcal{M} will be subject to the flipping operation f , as in Figure 6 where the binary output of the algorithm relies on the probabilities p and q shown in Lemma 1.

ALGORITHM 1: Main Algorithm

Require: $\mathcal{G}, \{SG_{BK}\}, p$

Ensure: \mathcal{G}^*

$\{SG_1, \dots, SG_n\} \leftarrow \text{SampleSG}(\mathcal{G}_\cup, \{SG_{BK}\})$

$\mathcal{M}^{n \times m} \leftarrow \text{BuildLogicalMatrix}(\mathcal{G}, \{SG\})$

for $i \leftarrow 1$ to n **do**

for $j \leftarrow 1$ to m **do**

$\mathcal{M}_{ij}^* \leftarrow \text{Flip}(\mathcal{M}_{ij}, p)$

end for

end for

$\mathcal{G}^* = \mathcal{G}$

for $i \leftarrow 1$ to n **do**

for $j \leftarrow 1$ to m **do**

if $\mathcal{M}_{ij} == 1$ & $\mathcal{M}_{ij}^* == 0$ **then**

$G_i^* \leftarrow \text{DropSG}(SG_i, G_j^*)$

end if

end for

end for

for $i \leftarrow 1$ to n **do**

for $j \leftarrow 1$ to m **do**

if $\mathcal{M}_{ij}^* == 1$ **then**

$G_i^* \leftarrow \text{AddSG}(SG_i, G_j^*)$

end if

end for

end for

The main algorithm takes as input the original set of graphs \mathcal{G} , a number of subgraphs that we estimate representing the background knowledge or public knowledge of this set, and the probability p . The output is the noisy version of the set \mathcal{G}^* . The algorithm adapts the first order explained in Subgraph 5.1, which focuses on well representing the 1s of the matrix \mathcal{M}^* in the released graphs. In step 1, we sample a number of subgraphs based on the data provided by the Union Graph \mathcal{G}_U and the set of background knowledge subgraphs. Each edge in \mathcal{G}_U contains data concerning the graphs where this edge exists. In the second step, the logical matrix $\mathcal{M}^{n \times m}$ is created where each row represents a subgraph, and each column represents a graph of \mathcal{G} and each binary value \mathcal{M}_{ij} represents the existence of subgraph SG_i in graph G_j .

```

1: function FLIP( $val, p$ )
2:    $rnd := \text{random}(0, 1)$ 
3:   if  $rnd \leq p$  then
4:      $val^* := val$ 
5:   else
6:     if  $val == 0$  then
7:        $val^* := 1$ 
8:     else
9:        $val^* := 0$ 
10:    end if
11:  end if return  $val^*$ 
12: end function

```

In steps 3–7, the algorithm creates the noisy matrix \mathcal{M}^* by applying the function Flip. This function takes a binary value and a probability of p as input; it generates a random number of rnd between 0 and 1; if rnd is higher than p , then the binary value is flipped.

In steps 9–15, the algorithm drops all the subgraphs that should be deleted based on the flipping process. While in steps 16–22, the algorithm adds the corresponding subgraphs, even if these subgraphs are already in the original graph, because steps 9–15 might have caused the drop of some of these subgraphs unintentionally.

AddSG searches for all the edges that should be in the subgraph SG_i and adds all of these edges that do not exist in the graph G_j^* . While the *DropSG* function sorts all the edges of SG_i , then deletes the one having the least weight in the Union Graph \mathcal{G}_U . The edges' weight is calculated while creating the Union Graph based on the required type of the subgraphs' sampling, for example, sampling the high correlated subgraphs or the least correlated subgraphs.

It is important to note that before releasing, we have to compute $\delta' = \frac{FaultSg}{n \times m}$, where *FaultSg* is the number of subgraphs in \mathcal{G} that does not reflex their binary values in \mathcal{M}^* and $n \times m$ is the number of elements in \mathcal{M}^* .

Finally, we have to verify the Inequality (6). In case the inequality is verified, then it is safe to release the graphs under the guarantee of (ϵ, δ) -BP. Otherwise, the mechanism must be re-executed, or the data owner may decide to increase the relaxation parameter δ .

6.2 Discussion

In this subsection, we will discuss three issues. The first one is as follows: Is it possible to have a different result by re-executing the mechanism without changing any of the inputs ϵ , δ , and K ? Actually, yes, it is possible. Any BP or DP algorithms are called a randomized algorithm, which means they employ a degree of randomness as part of their logic. In our case, the two probabilities p and q are concerned with this randomness. The value of these probabilities will remain the same as the ϵ is not changed. However, even if the probabilities and the original matrix did not change,

each value in this matrix is subjected to a flip with a probability of q . The group of subgraphs that should be flipped and the rate of subgraphs in this group that will not be flipped practically because of other flips, as explained in Subsection 5.1 and especially in Figure 7, cannot be controlled by the mechanism. Therefore, if δ' is not respecting the Inequality (6), then maybe a re-execution will fix the problem. However, if many re-executions did not help, this is an indicator that the relaxation parameter δ provided by the data owner is very small for this dataset and should be increased.

The second issue is about how n the number of sampled subgraphs may impact the result. Any increase in the number of sampled subgraphs means more manipulation in the edges. That is because the number of subgraphs that should be added or removed will increase, leading to a big increase in the added edges and a smaller increase in the removed edges, especially if we are removing just one edge for each subgraph. Overall, the rise in the value of n will lead to more privacy and less utility.

The third issue is why we have chosen to remove just one edge from each subgraph that should be dropped. In Figure 7, we have explained how deleting one edge may effect a subgraph that should be added but does not exist in the final result. When deleting two or more edges from each subgraph, the possibility of this impact increases, so the rate δ' . Hence, the possibility that the Inequality 6 is not respected becomes higher, forcing the data owner to re-execute the mechanism or increase the relaxation parameter δ .

```

function AddSG( $SG_i, G_j^*$ )
  for each edge  $e$  in  $SG_i$  do
    if  $e$  not in  $G_j^*$  then
       $add(e)$  to  $G_j^*$ 
    end if
  end for
  return  $G_j^*$ 
end function

```

```

function DropSG( $SG_i, G_j^*$ )
  Sort all edges of  $SG_i$  based on
  their weights in  $\mathcal{G}_U$ 
   $\# \mathcal{G}_U = G_1 \cup G_2 \cup \dots \cup G_n$ 
   $e_{min} :=$  edge of  $SG_i$  with
  minimum weight
   $drop(e_{min})$  from  $G_j^*$  return  $G_j^*$ 
end function

```

7 EXPERIMENTS

This section will present our experiments' results to evaluate our approach both in terms of utility and privacy. We implement our algorithm in Python on the "Autonomous Systems AS-733" dataset [29, 30] containing 733 releases, 6,474 vertices, and 13,895 edges in the largest release, and we conducted experiments on an Intel Core i7 2.4-GHz PC with 8 GB RAM.

As well as our algorithm, we will apply an edge-DP algorithm called TmF [42] that adds noise to each graph without considering the correlations. We will compare the results of both algorithms based on many privacy and utility measures.

7.1 Tradeoff between Privacy and Utility

When applying DP or BP mechanisms, one of the data owners' issues is choosing the privacy parameters to set an acceptable tradeoff between privacy and utility. This subsection proposes a method for the data owners applying our mechanism and the TmF mechanism to choose ϵ .

The values we have used for the privacy parameter ϵ of our algorithm are: 0.1, 0.2, 0.5 and 1. The TmF mechanism has two privacy parameters ϵ_1 and ϵ_2 . In Reference [42], they fix the value of $\epsilon_2 = 0.1$, while ϵ_1 is related to the number of nodes $|V|$: $\epsilon_1 = coef \times \ln(|V|)$, where $coef = 1, 2$ and 3. In our work, we add also the value 0.5 to the values of $coef$.

In this subsection, we use the confusion matrix as a method for the data owners to choose ϵ for our BP mechanism and ϵ_1 for the TmF mechanism. We extract a number of subgraphs from each

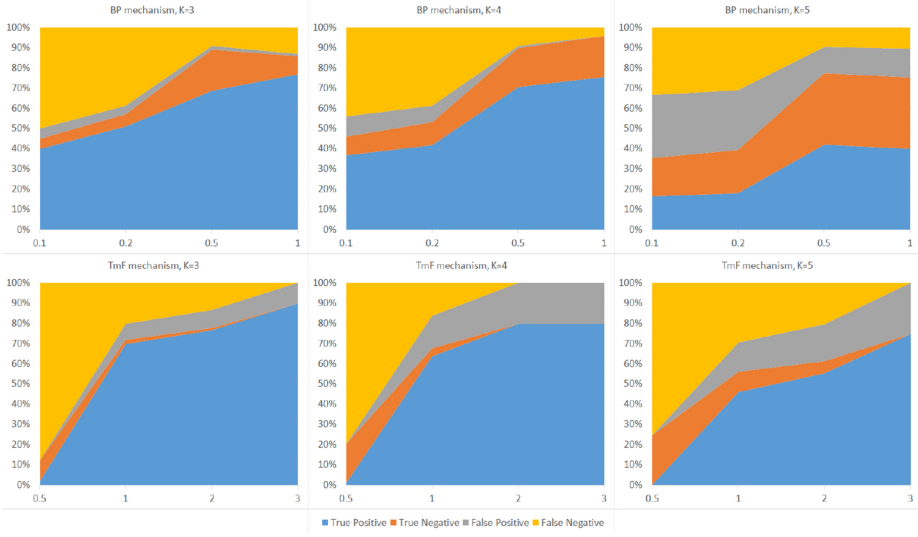


Fig. 9. Confusion matrix for sampled subgraphs in every graph and its noisy versions based on K (number of vertices for each subgraph) and privacy parameter.

original graph and project them on original and private versions of the graph to find isomorphic subgraphs. We compare the number of subgraphs found in all the versions and compute the ratio of:

- **True Positive:** Subgraphs that appear in both original and released versions.
- **False Positive:** Subgraphs appear in the released version but do not exist in the original one.
- **True Negative:** Subgraphs that exist neither in the original graph nor in the released one.
- **False Negative:** Subgraphs that exist in the original graph but do not appear in the released one.

In Figure 9, we consider the percentage of True Positive and True Negative as the ratio of utility, while the rate of False Positive and False Negative is the ratio of privacy. It is up to the data owners to choose the privacy parameter based on how much privacy they want in their released graphs. For the BP mechanism, increasing ϵ means more utility and less privacy. Increasing $coef$ for ϵ_1 in the TmF mechanism also leads to less privacy, but this drop in privacy is quicker than the decline in the BP mechanism. We can then say that our mechanism maintains a more balanced tradeoff between privacy and utility than TmF while changing each mechanism's privacy parameters.

7.2 Measuring Privacy

In these experiments, we are focusing on protecting the high correlated subgraphs. By high correlated, we mean the subgraphs that frequently appear in the original graphs. Therefore, we have sampled 1,000 high correlated subgraphs. We consider the scenario that an adversary has generated the Intersection graph from our released graphs. Then we compare the percentage of these 1,000 subgraphs that appear in the Intersection graphs generated from the released graphs under our mechanism and TmF mechanism. Higher numbers mean more high correlated subgraphs are not protected.

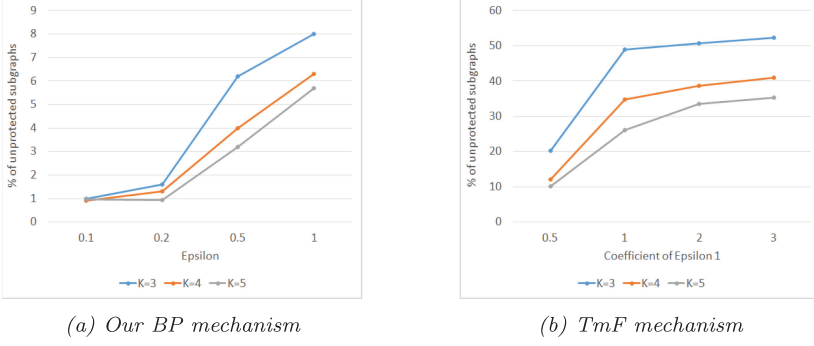


Fig. 10. Percentage of sampled high correlated subgraphs unprotected in Intersection graphs.

Figure 10 shows that the percentages of high correlated subgraphs that our mechanism failed to protect in the Intersection of the released graphs are much smaller than these of the TmF mechanism. It also shows that increasing the number of nodes in the sampled subgraphs from 3 to 5 leads to a smaller portion. The reason is that a higher number of nodes in a subgraph means a higher number of edges, which increases the possibility that one of these edges is removed in the Intersection, which is enough to consider this subgraph as protected.

Also, in Figure 10, we can notice that a higher ϵ increases the percentage of failure for both algorithms and the three K values. That is reasonable, because having a higher ϵ means that the value of probability p has increased. In contrast, the value of q has decreased, which leads to a decline in the number of flips. Therefore an rise in the number of unprotected subgraphs can be predictable.

7.3 Kullback–Leibler Divergence for Union and Intersection Graphs

In this subsection, we use Kullback–Leibler (KL) Divergence to compare the distributions of degrees of nodes between the Union of the original graphs and the Union of the released graphs under our mechanism and TmF mechanism. We generate the Union of the original graphs then the Union of the released graphs. Each edge in these Unions is weighted by the number of times it appears in the graphs. For example, an edge that appears in 35 graphs, its weight in the Union is 35. We create two lists of edges W_U containing all the edges and their weights in the original Union and W'_U for the edges of the Union of released graphs,

$$D_{KL}(W'_U \parallel W_U) = \left| \sum_{e \in E_{W'_U}} W'_U[e] \times \log \frac{W'_U[e]}{W_U[e]} \right|, \quad (10)$$

where $E_{W'_U}$ are all the edges listed in list W'_U and $W'_U[e]$ is the weight of edge e .

For the second KL Divergence, we generate the weighted Intersections of the original and the released graphs. Then we create two lists W_\cap and W'_\cap containing the edges that appear in at least one of the two Intersections. The weight of each edge in W_\cap is the number of times it appears in the original graphs, while its weight in W'_\cap is the number of times it appears in the released graphs.

$$D_{KL}(W'_\cap \parallel W_\cap) = \left| \sum_{e \in E_{W'_\cap}} W'_\cap[e] \times \log \frac{W'_\cap[e]}{W_\cap[e]} \right|. \quad (11)$$

In Figure 11(a), KL Divergence of Union graphs under the TmF mechanism is much higher than those under the BP mechanism for $K = 3, 4$, and 5. That is because TmF adds and removes a large

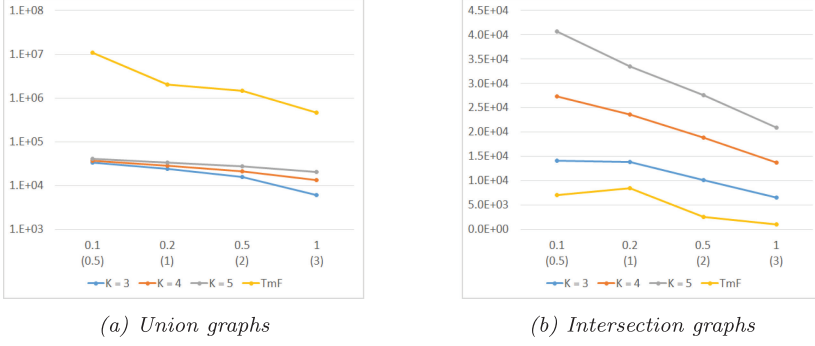


Fig. 11. KL Divergence for Union and Intersection graphs. (The above values in horizontal axis are the values of ϵ for our algorithm. The bottom values represent $coef$ for TmF mechanism.)

number of edges all over each graph, and all these manipulations affect the Union graphs, leading to this high KL Divergence for TmF. However, our mechanism focuses on manipulating the edges that create high correlated subgraphs. All other edges are not affected by our algorithm; hence, the KL Divergence under our mechanism is much smaller than the KL divergence of TmF.

In Figure 11(b), we see that KL Divergence for TmF is smaller than KL Divergence for our algorithm. This chart proves the motivation for this work. As we have explained, using a DP mechanism on sequentially released graphs will cause the injection of much noise, as shown in Figure 11(a). Still, most of the noise disappears when generating the released graphs' Intersection, which may lead to a serious privacy breach.

We also notice from the two charts in Figure 11 that a higher K leads to a higher KL Divergence, which means more privacy and less utility. It can be explained as a result of adding a high number of edges for each sampled subgraph that should be added to $K = 5$ nodes, for example, compared to $K = 3$.

7.4 Centrality Measures

The majority of the studies on social networks focus on extracting the most important individuals (vertices) in the network. We will prove that the effect of our mechanism on the list of most important individuals is negligible.

To find the most important individuals, we study the centrality of the vertices, and according to the purpose of each study, we choose the type of centrality measures. Linton C. Freeman said in 1978 [19]: “There is certainly no unanimity on exactly what centrality is or on its conceptual foundations, and there is little agreement on the proper procedure for its measurement,” which is still true today. However, we can define it as a measurement of the extent to which an individual interacts with other individuals in the network. We compare the 100 most important nodes of the original graphs, the BP version, and the TmF version based on four centralities:

- Degree: It is the number of vertices at a distance one for each vertex. In our case, where the graph is a simple undirected graph, we can define the degree simply as the number of edges connected to each vertex. The degree centrality [9] of a vertex v is defined as $C_D = deg(v)$. In many social studies, people with the most connections are the most important individuals in the network.
- Farness: The farness of a node v is the sum of all shortest paths of v to all other nodes. The closeness [44] is the reciprocal of the farness : $C(v) = \frac{1}{\sum_y d(y, v)}$, where $d(y, v)$ is the

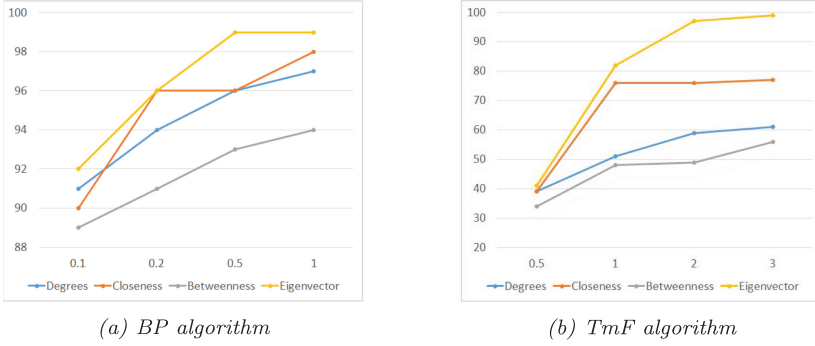


Fig. 12. Number of common nodes for the 100 most important nodes based on four centralities.

distance between vertices v and y . In this case, the most important vertices are the closest ones to all other nodes. Thus, we can rely on these individuals to spread information to all other nodes sequentially.

- Betweenness [1]: This centrality quantifies the number of times a vertex occurs on a geodesic; in other words, it is the number of times a vertex appears in the shortest path between two other nodes.

This centrality was proposed by Freeman [18], and his idea was that actors who exist between other individuals might control the interactions between these individuals. Then, this centrality quantifies the control of a vertex on the communication of other vertices. Therefore, in this case, the most important individuals have a high probability of occurring in the shortest path of two vertices randomly chosen.

The betweenness centrality of vertex v is defined as $C_B(v) = \sum_{i < j} \frac{sp_{ij}(v)}{sp_{ij}}$, where sp_{ij} is the number of shortest paths between vertices i and j , and $sp_{ij}(v)$ is the number of shortest paths between vertices i and j that pass through v .

- Eigenvector [8]: This centrality measures the influence of the vertex in the graph. It depends on the number and the quality of the connections. Therefore, a vertex v_1 , with number of connections less than a vertex v_2 , may outrank v_2 if the quality of its connections is higher. The centrality score of vertex v can be defined as: $x_v = \sum_{i \in N(v)} x_i = \sum_i A_{vi} x_i$, where $N(v)$ is a set of the neighboring nodes of v , A is the non-negative adjacency matrix of the graph.

The main idea of releasing multiple graphs sequentially instead of releasing just one graph is to improve the utility of the released data. For this reason, we compare the 100 most important nodes between each graph and its released versions to measure each graph's utility instead of just the Union and the Intersection graphs. Then we count the common nodes in these 100 nodes to determine the number of most important nodes unaffected by the anonymization mechanism.

In Figure 12, we compute the mean of the common nodes counts for all the graphs in each release. We consider this mean as a measure of utility for the set of released graphs. The number of common important nodes of our algorithm for $K = 3, 4$, and 5 are incredibly close; then, we choose to use the number of common important nodes for just $K = 4$. Figure 12(a) shows that our algorithm highly preserves the most important nodes. Approximately more than 90% of the most important nodes are preserved in our algorithm's released graphs. While in Figure 12(b), we see that the algorithm does not preserve most of the important nodes, especially for $coef = 0.5$. That changes when $coef$ increases, but in most cases, our utility is still much higher than the utility provided by TmF. We can explain that difference in utility due to focusing just on manipulating

the edges related to particular subgraphs (in our experiments, the high correlated subgraphs) we aim to protect, while the TmF mechanism is manipulating edges all over the graph.

8 CONCLUSION

This article has proposed, implemented, and evaluated a BP mechanism to provide privacy for sequentially released graphs. We have proved that DP cannot ensure strong privacy for subgraphs in an intersection of several releases. Then we have proposed our solution based on the BP definition. In the experiments, we have proposed a method for the data owners to help them calibrating the tradeoff of privacy and utility of their graphs. We have proved that our mechanism preserves the graphs' utility better than a DP mechanism and provides strong privacy for subgraphs.

In our future work, we will focus on the online approach instead of the offline, where we have to deal with each instance based on past instances without any info about future instances. We will also study the online model's robustness against other types of attacks than the background knowledge represented in this article. The other types of attacks could be degree-trail or the scenario that the attackers are presented in the graph and try to identify themselves and then identify other related to their node in the graph.

REFERENCES

- [1] Taras Agryzkov, Leandro Tortosa, and Jose F. Vicent. 2019. A variant of the current flow betweenness centrality and its application in urban networks. *Appl. Math. Comput.* 347 (2019), 600–615. DOI: [10.1016/j.amc.2018.11.032](https://doi.org/10.1016/j.amc.2018.11.032)
- [2] Faraz Ahmed, Rong Jin, and Alex X. Liu. 2013. A random matrix approach to differential privacy and structure preserved social network graph publishing. *arXiv:1307.0475*. Retrieved from <https://arxiv.org/abs/1307.0475>.
- [3] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geoindistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. 901–914.
- [4] Michael Barbaro, Tom Zeller, and Saul Hansell. A face is exposed for AOL searcher no. 4417749. *New York Times*, August 9, 2006.
- [5] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. 2010. Privacy in dynamic social networks. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 1059–1060.
- [6] Smriti Bhagat, Graham Cormode, Divesh Srivastava, and B. Krishnamurthy. 2010. Prediction promotes privacy in dynamic social networks. In *Proceedings of the 3rd Conference on Online Social Networks*.
- [7] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2012. The johnson-lindenstrauss transform itself preserves differential privacy. In *Proceedings of the IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS'12)*. IEEE, 410–419.
- [8] Phillip Bonacich. 2007. Some unique properties of eigenvector centrality. *Soc. Netw.* 29, 4 (2007), 555–564.
- [9] Piotr Bródka, Krzysztof Skibicki, Przemysław Kazienko, and Katarzyna Musiał. 2011. A degree centrality in multi-layered social network. In *Proceedings of the 2011 International Conference on Computational Aspects of Social Networks (CAsoN'11)*. IEEE, 237–242.
- [10] Rui Chen, Noman Mohammed, Benjamin C. M. Fung, Bipin C. Desai, and Li Xiong. 2011. Publishing set-valued data via differential privacy. *Proc. VLDB Endow.* 4, 11 (2011), 1087–1098.
- [11] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 653–664.
- [12] Xihui Chen, Sjouke Mauw, and Yuniór Ramírez-Cruz. 2020. Publishing community-preserving attributed social graphs with a differential privacy guarantee. *Proc. Priv. Enhanc. Technol.* 2020, 4 (2020), 131–152. DOI: [10.2478/popets-2020-0066](https://doi.org/10.2478/popets-2020-0066)
- [13] Elie Chicha, Bechara Al Bouna, Mohamed Nassar, and Richard Chbeir. 2018. Cloud-based differentially private image classification. *Wireless Netw.* 24 (2018), 1–8. DOI: [10.1007/s11276-018-1885-y](https://doi.org/10.1007/s11276-018-1885-y)
- [14] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 123–138.
- [15] Yitao Duan, Jingtao Wang, Matthew Kam, and John Canny. 2005. Privacy preserving link analysis on dynamic weighted graph. *Comput. Math. Org. Theory* 11, 2 (2005), 141–159.
- [16] Cynthia Dwork. 2006. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Part II (ICALP'06)*, Vol. 4052. Springer Verlag, Berlin, 1–12.

- [17] Úlfar Erlingsson, Vasily Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1054–1067.
- [18] Linton C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40, 1 (1977), 35–41. DOI: [10.2307/3033543](https://doi.org/10.2307/3033543)
- [19] Linton C. Freeman. 1978. Centrality in social networks conceptual clarification. *Soc. Netw.* 1, 3 (1978), 215–239.
- [20] Anupam Gupta, Aaron Roth, and Jonathan Ullman. 2012. Iterative constructions and private data release. In *Proceedings of the Theory of Cryptography Conference*. Springer, 339–356.
- [21] Michael Hay, Chao Li, Jerome Miklau, and David Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09)*. IEEE, 169–178.
- [22] Michael Hay, Jerome Miklau, David Jensen, Don Towsley, and Philipp Weis. 2008. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.* 1, 1 (2008), 102–114. DOI: [10.14778/1453856.1453873](https://doi.org/10.14778/1453856.1453873)
- [23] Xi He, Ashwin Machanavajjhala, and Bolin Ding. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. ACM, 1447–1458.
- [24] Shen-Shyang Ho and Shuhua Ruan. 2011. Differential privacy for location pattern mining. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. ACM, 17–24.
- [25] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proc. VLDB Endow.* 4, 11 (2011), 1146–1157.
- [26] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Proceedings of the Theory of Cryptography Conference (TCC'13)*. Springer, 457–476.
- [27] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, 193–204.
- [28] Jaewoo Lee and Chris Clifton. 2011. How much is enough? Choosing ϵ for differential privacy. In *Proceedings of the International Conference on Information Security*. Springer, 325–340.
- [29] Jure Leskovec. 2000. *Autonomous Systems AS-733 @ONLINE*. Retrieved from <https://snap.stanford.edu/data/as-733.html>.
- [30] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 177–187.
- [31] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 106–115.
- [32] Xiang-Yang Li, Chunhong Zhang, Taeho Jung, Jianwei Qian, and Linlin Chen. 2016. Graph-based privacy-preserving data publication. In *Proceedings of the IEEE 35th Annual IEEE International Conference on Computer Communications (INFOCOM'16)*. 1–9.
- [33] Zhuolin Li, Xiaolin Zhang, Haochen Yuan, Yongping Wang, and Jian Li. 2019. Distributed privacy preserving technology in dynamic networks. *Int. J. High Perf. Comput. Netw.* 15, 3–4 (2019), 223–232.
- [34] Kun Liu, Kamalika Das, Tyrone Grandison, and Hillol Kargupta. 2008. Privacy-preserving data analysis on graphs and social networks. In *Next Generation of Data Mining. Chapman & Hall/Crc Data Mining and Knowledge Discovery Series*. 419–438.
- [35] Kun Liu and Evimaria Terzi. 2008. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 93–106.
- [36] Zhenpeng Liu, Yawei Dong, Xuan Zhao, and Bin Zhang. 2017. A dynamic social network data publishing algorithm based on differential privacy. *J. Inf. Secur.* 8, 04 (2017), 328.
- [37] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. 2006. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*. IEEE, 24–24.
- [38] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. l-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data* 1, 1 (2007), 3–es. DOI: [10.1145/1217299.1217302](https://doi.org/10.1145/1217299.1217302)
- [39] Kamalkumar Macwan and Sankita Patel. 2019. Privacy preserving approach in dynamic social network data publishing. In *Proceedings of the International Conference on Information Security Practice and Experience*. Springer, 381–398.
- [40] Frank D. McSherry. 2009. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, 19–30.
- [41] Mohamed Nassar, Elie Chicha, Bechara Al Bouna, and Richard Chbeir. 2020. VIP Blowfish Privacy in communication graphs. In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, Volume 3 (SECRYPT'20)*. INSTICC, SciTePress, 459–467. DOI: <https://doi.org/10.5220/0009875704590467>

- [42] Hiep Nguyen, Abdessamad Imine, and Michaël Rusinowitch. 2016. Network structure release under differential privacy. *Trans. Data Priv.* 9, 3 (2016), 26.
- [43] Kobbi Nissim, Thomas Steinke, Alexandra Wood, Micah Altman, Aaron Bembeneke, Mark Bun, Marco Gaboardi, David R. O'Brien, and Salil Vadhan. 2017. Differential privacy: A primer for a non-technical audience. In *Proceedings of the Privacy Law Scholars Conference*.
- [44] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. 2008. Ranking of closeness centrality for large-scale social networks. In *Proceedings of the International Workshop on Frontiers in Algorithmics*. Springer, 186–195.
- [45] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2012. A workflow for differentially-private graph synthesis. In *Proceedings of the 2012 ACM Workshop on Workshop on Online Social Networks*. ACM, 13–18.
- [46] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow.* 7, 8 (2014), 637–648.
- [47] Gu Qiuyang, Ni Qilian, Meng Xiangzhao, and Yang Zhijiao. 2019. Dynamic social privacy protection based on graph mode partition in complex social network. *Pers. Ubiqu. Comput.* 23, 3–4 (2019), 511–519.
- [48] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. 2011. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. ACM, 81–98.
- [49] Pierangela Samarati and Latanya Sweeney. 1998. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory*. SRI International. DOI: <https://doi.org/10.1184/R1/6625469.v1>
- [50] Kumar Sharad and George Danezis. 2013. De-anonymizing d4d datasets. In *Proceedings of the Workshop on Hot Topics in Privacy Enhancing Technologies*.
- [51] Kumar Sharad and George Danezis. 2014. An automated social graph de-anonymization technique. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 47–58.
- [52] Latanya Sweeney. 2001. *Computational Disclosure Control: A Primer on Data Privacy Protection*. Ph.D. Dissertation. Massachusetts Institute of Technology.
- [53] Chih-Jui Lin Wang, En Tzu Wang, and Arbee LP Chen. 2013. Anonymization for multiple released social network graphs. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 99–110.
- [54] Yue Wang and Xintao Wu. 2013. Preserving differential privacy in degree-correlation based graph generation. *Trans. Data Priv.* 6, 2 (2013), 127.
- [55] Yue Wang, Xintao Wu, and Leting Wu. 2013. Differential privacy preserving spectral graph analysis. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 329–340.
- [56] Yonghui Xiao and Li Xiong. 2015. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1298–1309.
- [57] Xiaowei Ying and Xintao Wu. 2008. Randomizing social networks: A spectrum preserving approach. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 739–750.
- [58] Liangwen Yu, Yonggang Wang, Zhengang Wu, Jiawei Zhu, Jianbin Hu, and Zhong Chen. 2014. Edges protection in multiple releases of social network data. In *Proceedings of the International Conference on Web-Age Information Management*. Springer, 669–680.
- [59] Rong Yue, YiDong Li, Tao Wang, and Yi Jin. 2018. An efficient adaptive graph anonymization framework for incremental data publication. In *Proceedings of the 5th International Conference on Behavioral, Economic, and Socio-Cultural Computing*. IEEE, 103–108.
- [60] Elena Zheleva and Lise Getoor. 2007. Preserving the privacy of sensitive relationships in graph data. In *Proceedings of the International Workshop on Privacy, Security, and Trust in KDD*. Springer, 153–171.
- [61] Bin Zhou and Jian Pei. 2008. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. IEEE, 506–515.