Anonymization of Centralized and Distributed Social Networks by Sequential Clustering

Tamir Tassa and Dror J. Cohen

Abstract—We study the problem of privacy-preservation in social networks. We consider the distributed setting in which the network data is split between several data holders. The goal is to arrive at an anonymized view of the unified network without revealing to any of the data holders information about links between nodes that are controlled by other data holders. To that end, we start with the centralized setting and offer two variants of an anonymization algorithm which is based on sequential clustering. Our algorithms significantly outperform the SaNGreeA algorithm due to Campan and Truta which is the leading algorithm for achieving anonymity in networks by means of clustering. We then devise secure distributed versions of our algorithms. To the best of our knowledge, this is the first study of privacy preservation in distributed social networks. We conclude by outlining future research proposals in that direction.

Index Terms—social networks; clustering; privacy preserving data mining; distributed computation.

1 Introduction

Networks are structures that describe a set of entities and the relations between them. A social network, for example, provides information on individuals in some population and the links between them, which may describe relations of friendship, collaboration, correspondence and so forth. An information network, as another example, may describe scientific publications and their citation links. In their most basic form, networks are modeled by a graph, where the nodes of the graph correspond to the entities, while edges denote relations between them. Real social networks may be more complex or contain additional information. For example, in networks where the described interaction is asymmetric (e.g., a financial transaction network), the graph would be directed; if the interaction involves more than two parties (e.g., a social network that describes co-membership in social clubs) then the network would be modeled as a hyper-graph; in case where there are several types of interaction, the edges would be labeled; or the nodes in the graph could be accompanied by attributes that provide demographic information such as `age`, `gender`, `location` or `occupation` which could enrich and shed light on the structure of the network.

Such social networks are of interest to researchers from many disciplines, be it sociology, psychology, market research, or epidemiology. However, the data in such social networks cannot be released as is, since it might contain sensitive information. Therefore, it is needed to anonymize the data prior to its publication in order to address the need to respect the privacy of the individuals whose sensitive information is included in the data. Data anonymization typically trades off with utility. Hence, it is required to find a golden path in which the released anonymized data still holds enough utility, on one hand, and preserves privacy to some accepted degree on the other hand.

A naïve anonymization of the network, in the sense of removing identifying attributes like names or social security numbers from the data, is insufficient. As shown in [2], the mere structure of the released graph may reveal the identity of the individuals behind some of the nodes. The idea behind the attack described in [2] is to inject a group of nodes with a distinctive pattern of edges among them into the network. The adversary then may link this distinctive structure to some set of targeted individuals. When the naïvely anonymized network is published, the adversary traces his injected subgraph in the graph; if successful (namely, there is only one such subgraph in the graph, an event of probability that can be made sufficiently high), the targets who are connected to this subgraph are re-identified and the edges between them are disclosed. Even less sophisticated adversaries may use prior knowledge of some property of their target nodes (say, the number of their neighbors and their interrelations) in order to identify them in the published graph and then extract additional information on them.

Hence, one needs to apply a more substantial procedure of anonymization on the network before its release. The methods of privacy preservation in networks fall into three main categories. The methods of the first category [15], [23], [31] provide k-anonymity via a deterministic procedure of edge additions or deletions. In those methods it is assumed that the adversary has a background knowledge regarding some property of its target node, and then those methods modify the graph so that it becomes k-anonymous with respect to that assumed property. The methods of the second category [5], [8], [10], [26], [27], [28] add noise to the data, in the form of random additions, deletions or switching of edges, in order to prevent adversaries from identifying their target in the network, or inferring the existence of links between nodes. The methods of the third category [6], [9], [29] do not alter the graph data like the methods of the two previous categories;

• *T. Tassa and D. Cohen are with the Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel.*

instead, they cluster together nodes into super-nodes of size at least k, where k is the required anonymity parameter, and then publish the graph data in that coarse resolution.

The study of anonymizing social networks has concentrated so far on centralized networks, i.e., networks that are held by one data holder. However, in some settings, the network data is split between several data holders, or players. For example, the data in a network of email accounts where two nodes are connected if the number of email messages that they exchanged was greater than some given threshold, might be split between several email service providers. As another example, consider a transaction network where an edge denotes a financial transaction between two individuals; such a network would be split between several banks. In such settings, each player controls some of the nodes (his clients) and he knows only the edges that are adjacent to the nodes under his control. It is needed to devise secure distributed protocols that would allow the players to arrive at an anonymized version of the unified network. Namely, protocols that would not disclose to any of the interacting players more information than that which is implied by its own input (being the structure of edges adjacent to the nodes under the control of that player) and the final output (the anonymized view of the entire unified network). The recent survey by X. Wu et al. about privacy-preservation in graphs and social networks [24] concludes by recommendations for future research in this emerging area. One of the proposed directions is distributed privacy-preserving social network analysis, which "has not been well reported in literature".

In this study we deal with social networks where the nodes could be accompanied by descriptive data, and propose two novel anonymization methods of the third category (namely, by clustering the nodes). Our algorithms issue anonymized views of the graph with significantly smaller information losses than anonymizations issued by the algorithms of [6] and [29]. We also devise distributed versions of our algorithms and analyze their privacy and communication complexity.

Organization of the paper. We begin by formal definitions in Section 2 and a survey of related work in Section 3. In Section 4 we stay in the realm of centralized networks and propose two variants of an anonymization algorithm which is based on sequential clustering [7]. In Section 5 we describe a distributed version of our algorithms that computes a k-anonymization of the unified network by invoking secure multiparty protocols. The results of our experiments are given in Section 6. We conclude in Section 7 by outlining future research directions in the study of privacy preservation in distributed networks.

2 THE MODEL

2.1 The data

We view the social network as a simple undirected graph, $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of nodes and $E \subseteq \binom{V}{2}$ is the set of edges[1]. Each node corresponds to an individual in the underlying group, while an edge that connects two nodes describes a relationship between the two corresponding individuals.

In addition to the structural data that is given by E, each node is described by a set of non-identifying attributes, such as `age` or `zipcode`, that are called quasi-identifiers. Combinations of such attributes could be used for unique identification by means of linking attacks [20], whence, they should be generalized in order to thwart such attacks. We let $A_1, \ldots, A_I$ denote the quasi-identifiers, as well as the set of values that they may attain (e.g., if $A_1 =$`gender` then $A_1 = \{M, F\}$). Then each node v_n, $1 \leq n \leq N$, is described by a quasi-identifier record, $R_n = (R_n(1), \ldots, R_n(I)) \in A_1 \times \cdots \times A_I$. (We adopt herein the notation convention that an integral index is denoted by a lower-case letter, while the upper limit of that index is denoted by the corresponding upper-case letter.)

To summarize, a (naïvely anonymized) social network is defined as follows:

Definition 2.1. *Let $A_1, \ldots, A_I$ be a collection of quasi-identifier attributes. A social network over $V = \{v_1, \ldots, v_N\}$ is $\mathcal{SN} = \langle V, E, \mathcal{R} \rangle$ where $E \subseteq \binom{V}{2}$ is the structural data (edges), describing relationships between individuals in V, and $\mathcal{R} = \{R_1, \ldots, R_N\}$, where $R_n \in A_1 \times \cdots \times A_I$, $1 \leq n \leq N$, are the descriptive data of the individuals in V.*

2.2 Anonymization by clustering

As in [6], [9], [29], we consider anonymizations of a given social network by means of clustering. Let $\mathcal{C} = \{C_1, \ldots, C_T\}$ be a partition of V into disjoint subsets, or clusters; i.e., $V = \bigcup_{t=1}^{T} C_t$ and $C_t \cap C_s = \emptyset$ for all $1 \leq t \neq s \leq T$. The corresponding clustered graph $G_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ is the graph in which the set of nodes is $V_{\mathcal{C}} = \mathcal{C}$, and an edge connects C_t and C_s in $E_{\mathcal{C}}$ iff E contains an edge from a node in C_t to a node in C_s. Each node $C_t \in V_{\mathcal{C}}$ is accompanied by two pieces of information — $|C_t|$ (the number of original V-nodes that C_t contains), and e_t, which is the number of edges in E that connect nodes within C_t. In addition, each edge $\{C_t, C_s\} \in E_{\mathcal{C}}$ is labeled by a weight $e_{t,s}$ that stands for the number of edges in E that connect a node in C_t to a node in C_s.

Let $G_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ be a clustered graph that was derived from a graph $G = (V, E)$ of some social network $\mathcal{SN} = \langle V, E, \mathcal{R} \rangle$. Then, in addition to the structural data, which is given by $E_{\mathcal{C}}$ and the integral labels of the nodes, $(|C_t|, e_t)$, and of the edges, $e_{t,s}$, one accompanies such a graph with descriptive data that is derived from the original descriptive data $\mathcal{R}$. We apply the common method in anonymizing tabular data, and that is the generalization of the quasi-identifiers. Each of the quasi-identifiers, A_i, $1 \leq i \leq I$, is accompanied by a collection of subsets, $\overline{A}_i$, which are the subsets of A_i that could be used for generalization. For example, $\overline{A}_i$ could be a taxonomy for A_i; namely, a hierarchical generalization tree where each node represents some subset of A_i, the leaves are all the singleton subsets, the root is the entire set, and the direct descendants of any node form a partition of the subset corresponding to that node.

Given a cluster of nodes in V, say $C_t = \{v_{n_1}, \ldots, v_{n_m}\}$, one associates with it a generalized quasi-identifier record,

1. $\binom{V}{2}$ denotes the set of all unordered pairs of elements from V.

$\overline{R}_t = (\overline{R}_t(1), \ldots, \overline{R}_t(I)) \in \overline{A}_1 \times \cdots \times \overline{A}_I$, called *the closure* of C_t, which is the minimal generalized record that generalizes all of the original quasi-identifier records $R_{n_1}, \ldots, R_{n_m}$. Namely, $\overline{R}_t(i)$, $1 \leq i \leq I$, is the minimal (with respect to inclusion) subset in $\overline{A}_i$ that contains all of the values $R_{n_1}(i), \ldots, R_{n_m}(i) \in A_i$.

Definition 2.2. *Let $\mathcal{SN} = \langle V, E, \mathcal{R} \rangle$ be a social network and let $\overline{A}_1, \ldots, \overline{A}_I$ be generalization taxonomies for the quasi-identifier attributes $A_1, \ldots, A_I$. Then given a clustering $\mathcal{C} = \{C_1, \ldots, C_T\}$ of V, the corresponding clustered social network is $\mathcal{SN}_\mathcal{C} = \langle V_\mathcal{C} = \mathcal{C}, E_\mathcal{C}, \overline{\mathcal{R}} \rangle$ where:*

- *$E_\mathcal{C} \subseteq \binom{V_\mathcal{C}}{2}$ is a set of edges on $V_\mathcal{C}$, where $\{C_t, C_s\} \in E_\mathcal{C}$ iff there exist $v_n \in C_t$ and $v_{n'} \in C_s$ such that $\{v_n, v_{n'}\} \in E$;*
- *The clusters in $V_\mathcal{C}$ are labeled by their size and the number of intra-cluster edges, while the edges in $E_\mathcal{C}$ are labeled by the corresponding number of inter-cluster edges in E;*
- *$\overline{\mathcal{R}} = \{\overline{R}_1, \ldots, \overline{R}_T\}$, where $\overline{R}_t$ is the minimal record in $\overline{A}_1 \times \cdots \times \overline{A}_I$ that generalizes all quasi-identifier records of individuals in C_t, $1 \leq t \leq T$.*

An example of a network of seven nodes, with two-dimensional quasi-identifier records (`age` and `gender`), and a corresponding clustered network with three super-nodes, is given in Figure 1. (The two numbers in each super-node are its size and the number of intra-cluster edges.)

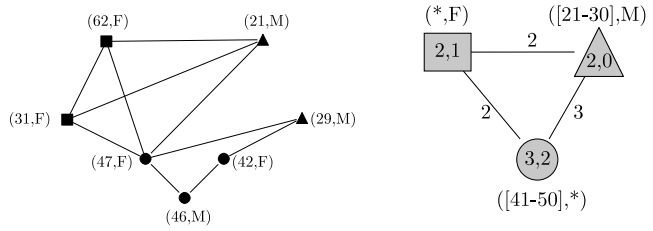

Fig. 1. A network and a corresponding clustering

Given a social network $\mathcal{SN} = \langle V, E, \mathcal{R} \rangle$, a corresponding clustered social network is called k-anonymous (or a k-anonymization of $\mathcal{SN}$) if the size of all of its clusters is at least k. Our goal is to find a k-anonymization in which the loss of information is minimal (or, in other words, the utility is maximal). To that end, we need to define measures of information loss.

2.3 Measuring the loss of information

We shall use here the same measure of information loss that was used in [6]. Given a social network $\mathcal{SN}$ and a clustering $\mathcal{C}$ of its nodes, the information loss associated with replacing $\mathcal{SN}$ by the corresponding clustered network, $\mathcal{SN}_\mathcal{C}$, is defined as a weighted sum of two metrics,

$$I(\mathcal{C}) = w \cdot I_D(\mathcal{C}) + (1 - w) \cdot I_S(\mathcal{C}) \,; \tag{1}$$

here, $w \in [0, 1]$ is some weighting parameter, $I_D(\mathcal{C})$ is the *descriptive* information loss that is caused by generalizing the exact quasi-identifier records $\mathcal{R}$ to $\overline{\mathcal{R}}$, while $I_S(\mathcal{C})$ is the *structural* information loss that is caused by collapsing all nodes of V in a given cluster of $V_\mathcal{C}$ to one super-node.

We define those two metrics similarly to [6]. For the descriptive metric we utilize the Loss Metric (LM) measure [11], [17]. Assume that the original node $v_n \in V$, $1 \leq n \leq N$, belongs to a cluster $C_t \in \mathcal{C}$, $1 \leq t \leq T$; then its quasi-identifier record, $R_n = (R_n(1), \ldots, R_n(I))$, is generalized to $\overline{R}_t = (\overline{R}_t(1), \ldots, \overline{R}_t(I))$. The LM metric associates the following loss of information with each of the nodes in that cluster,

$$I_D(C_t) = \frac{1}{I} \sum_{i=1}^{I} \frac{|\overline{R}_t(i)| - 1}{|A_i| - 1} \,; \tag{2}$$

here, $|\overline{R}_t(i)|$ is the size of the subset $\overline{R}_t(i)$ which generalizes the original value $R_n(i)$, and $|A_i|$ is the number of values in the domain of attribute A_i. Note that $I_D(C_t)$ ranges between zero and one, where $I_D(C_t) = 0$ iff all records in C_t are equal, whence no generalization was applied, while $I_D(C_t) = 1$ iff all records in C_t are so far off, that all attributes in the generalized record had to be totally suppressed.

The overall LM information loss is the result of averaging those losses of information over all nodes in V, i.e.,

$$I_D(\mathcal{C}) = \frac{1}{N} \cdot \sum_{t=1}^{T} |C_t| \cdot I_D(C_t) \,. \tag{3}$$

Due to averaging, also $I_D(\mathcal{C})$ ranges between zero and one.

As for the structural information loss, we distinguish between two types of such information losses:

- Intra-cluster information loss: Given a cluster C_t, the structure of C_t in the original graph is lost, and is replaced by the number of nodes in C_t, and the number e_t of edges in E that connected nodes in C_t. The corresponding information loss is quantified as the probability of wrongly identifying a pair of nodes in C_t as an edge or as a non-connected pair, and it equals

$$I_{S,1}(C_t) := 2e_t \cdot \left(1 - \frac{2e_t}{|C_t| \cdot (|C_t| - 1)}\right) . \tag{4}$$

- Inter-cluster information loss: Given two clusters, C_t and C_s, the structure of edges that connect nodes from C_t to nodes in C_s is lost, and is replaced by the number $e_{t,s}$ of edges between nodes in those two clusters. Here too, the information loss is quantified as the probability of wrongly identifying a pair of nodes in C_t and C_s as an edge or as a non-connected pair, and it equals

$$I_{S,2}(C_t, C_s) := 2e_{t,s} \cdot \left(1 - \frac{e_{t,s}}{|C_t||C_s|}\right) . \tag{5}$$

The overall structural information loss for the clustering $\mathcal{C} = \{C_1, \ldots, C_T\}$ is then

$$I_S(\mathcal{C}) = \frac{4}{N(N-1)} \left[\sum_{t=1}^{T} I_{S,1}(C_t) + \sum_{1 \leq t \neq s \leq T} I_{S,2}(C_t, C_s) \right] , \tag{6}$$

where, as shown in [6], the normalizing factor $\frac{4}{N(N-1)}$ guarantees that $I_S(\mathcal{C})$ ranges between zero and one. The maximal value of one occurs when all edge counters (e_t and $e_{t,s}$) fall in the middle of the intervals where they range (i.e., $e_t = \binom{|C_t|}{2}/2$ and $e_{t,s} = |C_t||C_s|/2$ for all $1 \leq t \neq s \leq T$).

2.4 The k-anonymization clustering problem

We are now ready to define the problem of k-anonymizing a social network by means of clustering. Let $\mathcal{SN} = \langle V, E, \mathcal{R} \rangle$ be a social network as defined in Definition 2.1. Let $\overline{A}_1, \ldots, \overline{A}_I$ be generalization taxonomies for the quasi-identifier attributes $A_1, \ldots, A_I$. Given an integer $1 < k < n$, the goal is to find a clustering $\mathcal{C} = \{C_1, \ldots, C_T\}$ of V, where $|C_t| \geq k$ for all $1 \leq t \leq T$, that minimizes the information loss $I(\mathcal{C})$ that is defined in Eqs. (1)–(6).

If we take $E = \emptyset$, namely – we totally suppress the structural information, the social network reduces to a collection of tabular records and then we are looking at the well-known problem of k-anonymization in the context of tables. Since the latter problem is NP-hard [1], so is the problem for social networks. Hence, one should look at either approximation algorithms or heuristical algorithms. In Section 3 we describe existing heuristical algorithms for anonymization by clustering, and then, in Section 4, we introduce our algorithm.

An important question in this context is how to use the clustered network for the sake of performing data analysis. We defer that discussion to Section 6.3.

3 Previous algorithms of k-anonymization by clustering

The first study that considered the problem of k-anonymization of social networks by clustering was by Zheleva and Getoor [29]. The idea there was to apply any standard k-anonymization algorithm on the quasi-identifier records describing the nodes, in order to arrive at a clustering of the nodes, and then to hide the structural information in one of five suggested ways. One of the suggested ways was, as described above, to disclose the edge structure in the resolution of the super-nodes.

Campan and Truta [6] were the first to apply an anonymization algorithm that takes into account both the descriptive and structural data. Their algorithm, dubbed SaNGreeA (Social Network Greedy Anonymization), builds the clustering greedily, one cluster at a time, by selecting a seed node and then keep adding to it the next node that would minimize some measure of information loss, until it matures into a cluster of size k. Since that algorithm builds the clustering gradually, it cannot use the actual information loss measure $I(\cdot)$, Eq. (1), since it includes the structural information loss, $I_S(\cdot)$, Eq. (6), which may be evaluated only when all of the clustering is defined. Hence, they replaced $I_S(\cdot)$ with a distance metric between nodes, which was shown experimentally to be an effective substitute. The sequential clustering algorithm that we present herein does not suffer from that problem, since in each stage of its execution it has a full clustering and hence it may always make decisions according to the real measure of information loss.

In the same year as [6], Hay et al. [9] outlined an algorithm for k-anonymization by clustering. They restricted their attention to the case of structural information only. Their algorithm searched for the clustering which minimizes the number of "possible worlds" (namely, the number of graphs that are consistent with the information in the released clustered graph), using simulated annealing [14].

4 Anonymization by sequential clustering

The sequential clustering algorithm for k-anonymizing tables was presented in [7]. It was shown there to be a very efficient algorithm in terms of runtime as well as in terms of the utility of the output anonymization. We proceed to describe an adaptation of it for anonymizing social networks.

Algorithm 1 starts with a random partitioning of the network nodes into clusters. The initial number of clusters in the random partition is set to $\lfloor N/k_0 \rfloor$ and the initial clusters are chosen so that all of them are of size k_0 or $k_0 + 1$, where $k_0 = \alpha k$ is an integer and α is some parameter that needs to be determined.

The algorithm then starts its main loop (Steps 2-4). In that loop, the algorithm goes over the N nodes in a cyclic manner and for each node it checks whether that node may be moved from its current cluster to another one while decreasing the information loss of the induced anonymization. If such an improvement is possible, the node is transferred to the cluster where it currently fits best.

Algorithm 1.

- Input: A social network $\mathcal{SN}$, an integer k.
- Output: A clustering of $\mathcal{SN}$ into clusters of size $\geq k$.

1) Choose a random partition $\mathcal{C} = \{C_1, \ldots, C_T\}$ of V into $T := \lfloor N/k_0 \rfloor$ clusters of sizes either k_0 or $k_0 + 1$.
2) For $n = 1, \ldots, N$ do:
   a) Let C_t be the cluster to which v_n currently belongs.
   b) For each of the other clusters, C_s, $s \neq t$, compute the difference in the information loss, $\Delta_{n:t \to s}$, if v_n would move from C_t to C_s.
   c) Let C_{s_0} be the cluster for which $\Delta_{n:t \to s}$ is minimal.
   d) If C_t is a singleton, move v_n from C_t to C_{s_0} and remove cluster C_t.
   e) Else, if $\Delta_{n:t \to s_0} < 0$, move v_n from C_t to C_{s_0}.
3) If there exist clusters of size greater than k_1, split each of them randomly into two equally-sized clusters.
4) If at least one node was moved during the last loop, go to Step 2.
5) While there exist clusters of size smaller than k, select one of them and unify it with the cluster which is closest.
6) Output the resulting clustering.

During that main loop, we allow the size of the clusters to vary in the range $[2, k_1]$, where $k_1 = \beta k$ for some predetermined fixed parameter β. When a cluster becomes a singleton, we remove it and transfer the node that was in that cluster to the cluster where it fits best, in terms of information loss (Step 2d). On the other hand, when a cluster becomes too large (i.e., its size becomes larger than the upper bound k_1), we split it into two equally-sized clusters in a random manner.

The main loop of the algorithm is repeated until we reach a stage where an entire loop over all nodes in the network found no node that could be moved to another cluster in order

to decrease the information loss. That stopping criterion may be replaced by another one, in order to avoid iterations with negligible improvements. A natural criterion of that sort is to stop the main loop and continue to the next stage if the improvement in the information loss in the last execution of the main loop was too small. We used in our experiments the latter criterion, with a threshold of 0.5%.

At this point, some of the clusters are large, in the sense that their size is at least k, while others are small. If there exist small clusters, we apply an agglomerative procedure on them in the following manner (Step 5): We arbitrarily select one of them and then find which of the other clusters (of any size) is closest to it, in the sense that unifying them will cause the smallest increase in the information loss; after finding the closest cluster, we unify the two clusters. We repeat this procedure until all clusters are of size at least k.

The parameters α and β control the sizes of the clusters and, consequently, the information loss of the final output. The goal is to find a setting of α and β that would yield lower information losses. For example, higher values of β would result in larger clusters at the output, what implies higher information losses; on the other hand, a too small β would lead to a greater number of small clusters at the end of the first phase; those small clusters would need to be unified in the agglomerative phase (Step 5) and that too might result in higher information losses since the agglomerative phase is more crude than the first stage, as it involves the unification of whole clusters instead of moving just one node. (α has a much smaller effect on the information loss since it is used only once at the beginning.) Our experimentation with various values of α and β revealed that in all types of networks that we tested, and in all sizes, $\alpha = 0.5$ and $\beta = 1.5$ gave good or best results. Those were the values that we used in the experimental evaluation (Section 6).

Sequential clustering, like simulated annealing, is a local search algorithm. As local search procedures may be attracted to local minima, it is necessary to devise a mechanism that would allow the algorithm to explore other domains of the search space. Simulated annealing uses a temperature that determines the probability of accepting locally bad decisions. That temperature begins with a high value and then it is gradually cooled down until it reaches a predetermined level in which the search stops. The algorithm's performance is highly sensitive to the cooling schedule and how the probability of moving to a neighboring state is determined by the temperature. Sequential clustering, on the other hand, may be repeated several times with different random partitions as the starting point, in order to find the best local minimum among those repeated searches. Sequential clustering is known to perform better both in terms of runtime and quality of the output [19].

Sequential clustering achieves significantly better results than SanGreeA, in terms of information loss, as we demonstrate later on in Section 6. One reason is that greedy algorithms, such as SaNGreeA, do not have a mechanism of correcting bad clustering decisions that were made in an earlier stage; sequential clustering, on the other hand, constantly allows the correction of previous clustering decisions. Another advantage of sequential clustering over SaNGreeA is that it may evaluate at each stage during its operation the actual measure of information loss, since at each stage it has a full clustering of all nodes.

The latter advantage in terms of utility translates to a disadvantage in terms of runtime. While SaNGreeA requires $O(N^2)$ evaluations of the cost function, the number of cost function evaluations in the sequential clustering depends on N^3. (The algorithm scans all N nodes and for each one it considers $O(N/k)$ alternative cluster allocations; the computation of the cost function for each such candidate alternative clustering requires to update the inter-cluster costs $I_{S,2}(\cdot,\cdot)$ for all $O(N/k)$ pairs of clusters that involve either the cluster of origin or the cluster of destination in that contemplated move.) Hence, we proceed to describe a relaxed variant of sequential clustering which requires only $O(N^2)$ evaluations of the cost function.

4.1 A modified structural information loss measure

In [6], the proposed SaNGreeA algorithm uses a measure of structural information loss that differs from the measure $I_S(\cdot)$ that is given by Eqs. (4)-(6). We proceed to define it.

Let B be the $N \times N$ adjacency matrix of the graph $G = (V, E)$, i.e., $B(n, n') = 1$ if $\{v_n, v_{n'}\} \in E$ and $B(n, n') = 0$ otherwise. Then, a Hamming-like distance is defined on V as follows,

$$D(n, n') := \frac{|\{\ell \neq n, n' : B(n, \ell) \neq B(n', \ell)\}|}{N - 2}\,. \quad (7)$$

This definition of distance induces the following measure of structural information loss per cluster,

$$I'_S(C_t) = \frac{1}{\binom{|C_t|}{2}} \cdot \sum_{v_n, v_{n'} \in C_t} \mathrm{D}(n, n')\,, \quad (8)$$

and a corresponding overall structural information loss,

$$I'_S(\mathcal{C}) = \frac{1}{N} \sum_{t=1}^{T} |C_t| \cdot I'_S(C_t) = \sum_{t=1}^{T} x(C_t) \quad (9)$$

where

$$x(C_t) = \frac{2}{N(|C_t| - 1)} \sum_{v_n, v_{n'} \in C_t} \mathrm{D}(n, n')\,. \quad (10)$$

In other words, I'_S of a given cluster is the average distance between all pairs of nodes in that cluster, and I'_S of the whole clustering is the corresponding weighted average of structural information losses over all clusters. The corresponding weighted measure of information loss is then,

$$I'(\mathcal{C}) = w \cdot I_D(\mathcal{C}) + (1 - w) \cdot I'_S(\mathcal{C})\,, \quad (11)$$

where, as before, $w \in [0, 1]$ and $I_D(\mathcal{C})$ is given by (2)-(3).

The significant difference between $I(\cdot)$ (Eq. (1)) and $I'(\cdot)$ is that the former cannot be evaluated before the entire clustering is determined, while the latter one can, since it is defined as a sum of independent intra-cluster information loss measures. As the SaNGreeA algorithm needs to make clustering decisions before all clusters are formed, it uses a distance function between a node and a cluster that is geared towards minimizing

the measure $I'(\cdot)$. As opposed to SaNGreeA, the sequential clustering algorithm can use either $I(\cdot)$ or $I'(\cdot)$. In Section 6 we show that sequential clustering which is guided by $I(\cdot)$ offers much better utility than the same algorithm when it is guided by $I'(\cdot)$.

While I' offers results of lesser utility than I, using I' leads to shorter runtimes. Whenever the sequential clustering algorithm implements one of its decisions – be it moving a node from one cluster to another, splitting a large cluster, or unifying two small clusters – all that is needed in order to update I' is to update the intra-cluster information loss measures of the two clusters that are involved in such an action; there is no need to update also the inter-cluster information loss measures that involve all other clusters (as is the case when using I). This is why the number of cost function evaluations that sequential clustering needs to perform reduces from $O(N^3)$ to $O(N^2)$, when switching from I to I', in similarity to the SaNGreeA algorithm.

5 THE DISTRIBUTED SETTING

Here we consider the distributed setting, in which the network data is split among M sites (or players) in the following manner: player m, $1 \leq m \leq M$, holds N_m of the nodes, say $V^m = \{v_1^m, \ldots, v_{N_m}^m\}$. The overall number of nodes is $N = \sum_{m=1}^{M} N_m$ and the unified set of nodes is $V = \bigcup_{m=1}^{M} V^m$. As for the structural data, $E \subseteq \binom{V}{2}$, it is split between the players in the following manner: Edges that connect two nodes in V^m are known only to player m; edges that connect nodes in V^m and $V^{m'}$ are known only to players m and m'.

There are two scenarios to consider in this setting:

1) Scenario A: Each player needs to protect the identities of the nodes under his control from other players, as well as the existence or non-existence of edges adjacent to his nodes.
2) Scenario B: All players know the identities of all nodes in V; the information that each player needs to protect from other players is the existence or non-existence of edges adjacent to his nodes.

To illustrate the difference between the two scenarios, let us return to the toy network in the left of Figure 1. Assume that it is split between three players — the "circular", the "square", and the "triangular" players; namely, the circular player controls the three circular nodes in the graph, while the square and triangular players control the corresponding square and triangular nodes. Assume that those players are banks, that the nodes are accounts in those banks, and that the edges denote financial transactions between the accounts. Here, each node is identified by an account number, but the bank is trusted to protect the identity of the clients that hold those accounts. Hence, the square bank is expected to hide the information that one of his clients is a 62 year old female and the other is a 31 year old female (as indicated by the quasi-identifier records (62,F) and (31,F) next to his nodes in Figure 1) since that might reveal the identity of the account holders. In addition, the square bank is expected to hide from the circular bank the internal transactions among his clients (there is one such edge in the illustrated graph) or between his clients and clients of the triangular bank (there are two such edges in the graph). This is an example of Scenario A. However, assume that the network is a correspondence network between email addresses. Here it is natural to assume that the identity of the nodes is not confidential, since typical email addresses disclose the name of the individual that holds them. In this case, it is needed only to protect the existence of edges between the nodes. Namely, even though the identity of the individuals behind the email addresses `MeTarzan@Site1.com` and `YouJane@Site2.com` is expected to be known to player 3, players 1 and 2 are trusted to withhold from player 3 the fact that those two individuals exchanged emails. This is an example of Scenario B.

The goal is to arrive at a k-anonymization of the combined social network. A naïve solution would be to unify the network data from all players and apply any anonymization algorithm on that unified network. However, such an approach is unacceptable since it requires each player to reveal to the other players sensitive information of his clients. (In Scenario A, the sensitive information that each player must withhold from the other players is the quasi-identifier records of his clients, as well as the structure of edges adjacent to his clients; in Scenario B the sensitive information is just the edges.) If there exists a trusted third party, each player may surrender to him his corresponding partial view of the network, and then the trusted third party will have a complete view of the entire network, on which he may apply any anonymization algorithm. Alas, such an ideal setting cannot always be assumed, whence the players must rely on themselves in order to carry out this computation while still respecting privacy. Therefore, a distributed protocol that protects the sensitive information from being disclosed is in order. Herein, we describe such a distributed protocol which is based on the sequential clustering algorithm that we presented in Section 4. Like all previous studies on anonymization protocols in distributed settings, we too assume that the players are semi-honest, i.e., they respect the protocol, but try to learn as much as they can (even by means of forming coalitions) from their own view of the protocol on the private information held by other players. (See [12], [18], [30] for a discussion and justification of that assumption.)

In this study we focus on Scenario B; Scenario A is significantly harder and is left for future research (see more about it in Section 7). We present here a distributed version of the sequential clustering that uses the modified information loss measure, $I'(\mathcal{C})$, as defined in (11). A distributed sequential clustering which is guided by the original information loss measure, $I(\mathcal{C})$, Eq. (1), goes along the same lines. We describe the distributed protocol in Section 5.1 and analyze it in Sections 5.2–5.5.

5.1 Distributed sequential clustering

5.1.1 Overview

In order to implement Algorithm 1 in the distributed setting, the players need to compute differences of the form

$$I'(\mathcal{C}) - I'(\hat{\mathcal{C}}) = w \cdot (I_D(\mathcal{C}) - I_D(\hat{\mathcal{C}})) + (1-w) \cdot (I'_S(\mathcal{C}) - I'_S(\hat{\mathcal{C}}))$$

where $I'(\cdot)$ is the weighted measure of information loss, (11), $\mathcal{C}$ is the current clustering, and $\hat{\mathcal{C}}$ is a contemplated clustering. During the main loop of the algorithm, $\hat{\mathcal{C}}$ are clusterings that differ from $\mathcal{C}$ in the location of just one node; in the agglomerative stage that follows, the clusters of $\hat{\mathcal{C}}$ coincide with those of $\mathcal{C}$, except for one cluster that equals the union of two clusters in $\mathcal{C}$. Such computations are needed in order to make the right decision – what is the currently best cluster for a given node, or which two clusters are closest and should be unified.

In Scenario B, the descriptive information of all nodes can be made known to all players. Hence, the difference in the descriptive information loss, $I_D(\cdot)$, can be computed in an open public manner. It is the difference in the structural information loss, $I'_S(\cdot)$, that must be computed in a secure manner since it depends on the edge structure of the graph which is split between the various players and must not be disclosed. We proceed to describe a secure multi-party protocol (SMP hereinafter) that performs such computations.

Throughout this section, we shall assume that the nodes in $V = \bigcup_{m=1}^{M} V^m = \{v_1, \ldots, v_N\}$ are ordered so that the first N_1 nodes are those from V^1, the next N_2 nodes are those from V^2 and so forth. For each $1 \leq m \leq M$, let $\Omega_m = \{1 + \sum_{i=1}^{m-1} N_i, \ldots, \sum_{i=1}^{m} N_i\}$ be the set of indices in $\{1, \ldots, N\}$ that correspond to the nodes from V^m.

Section 5.1 is organized as follows. In Sections 5.1.2 to 5.1.4 we revisit the three main stages of Algorithm 1 and explain how they can be implemented securely in the distributed setting. In Section 5.1.5 we explain how the players can securely compute the clustered social network that corresponds to the clustering of the nodes of the unified network that they jointly computed. In Section 5.1.6 we comment on how the number of SMP calls may be reduced by means of parallelization. Then, in Section 5.1.7 we describe the SMP that our algorithm invokes for secure computation of sums.

5.1.2 Initial partitioning (Step 1)

In Step 1 of Algorithm 1, each player generates a random and uniform labeling of his own nodes by labels from $\{1, \ldots, T := \lfloor N/k_0 \rfloor\}$. The cluster C_t, $1 \leq t \leq T$, consists of all nodes in V that have the label t. The allocation of nodes to clusters is made known to all players.

5.1.3 Single node transitions (Step 2)

During the main loop in Algorithm 1 (Step 2), it is needed to compute the *difference* in the structural information loss if a given node v_n would move from its current cluster C_t to any of the other clusters, C_s, $s \in [T] \setminus \{t\}$. In view of Eq. (9), that difference equals

$$\delta(n : t \to s) := \tag{12}$$
$$(x(C_t \setminus \{v_n\}) + x(C_s \cup \{v_n\})) - (x(C_t) + x(C_s)) \,.$$

We proceed to define a set of values $\delta_m(n : t \to s)$, $1 \leq m \leq M$, with the following properties: $\delta_m(n : t \to s)$ can be computed independently by the mth player; and

$$\delta(n : t \to s) = \sum_{m=1}^{M} \delta_m(n : t \to s) \,. \tag{13}$$

Hence, each player may compute his value $\delta_m(n : t \to s)$, and then, using the SMP that we describe in Section 5.1.7, the players may arrive at the sought-after change in the structural information loss, $\delta(n : t \to s)$, if v_n is moved from C_t to C_s.

Let B be the $N \times N$ adjacency matrix of the unified graph (V, E), as defined in Section 4.1. By our assumption, player m knows only the N_m rows (and columns) of B that correspond to the nodes in V^m, i.e., only the entries $B(n, n')$ where $n \in \Omega_m$ or $n' \in \Omega_m$. Hence, he may compute the $N \times N$ matrix H_m which is defined as follows,

$$H_m(n, n') = \frac{|\{\ell \in \Omega_m \setminus \{n, n'\} : B(n, \ell) \neq B(n', \ell)\}|}{N - 2} \,. \tag{14}$$

The distance (7) between every two nodes is then

$$D(n, n') = \sum_{m=1}^{M} H_m(n, n') \,. \tag{15}$$

Assume that the players wish to compute the information loss $x(C_t)$, (10), of some cluster which typically consists of nodes from all players. To that end, the mth player computes

$$x_m(C_t) := \frac{2}{N(|C_t| - 1)} \cdot \sum_{v_n, v_{n'} \in C_t} H_m(n, n') \,, \tag{16}$$

and then, by Eqs. (10) and (15),

$$x(C_t) = \sum_{m=1}^{M} x_m(C_t) \,. \tag{17}$$

Hence, we may define

$$\delta_m(n : t \to s) := \tag{18}$$
$$(x_m(C_t \setminus \{v_n\}) + x_m(C_s \cup \{v_n\})) - (x_m(C_t) + x_m(C_s)) \,.$$

Indeed, in view of Eqs. (12) and (17), such values satisfy the sum property (13).

5.1.4 The agglomerative stage (Step 5)

The change in I'_S if clusters C_t and C_s would be unified is, as implied by Eq. (9),

$$\delta(t, s) := x(C_t \cup C_s) - (x(C_t) + x(C_s)) \,. \tag{19}$$

In order to compute it, the mth player, $1 \leq m \leq M$, computes

$$\delta_m(t, s) = x_m(C_t \cup C_s) - x_m(C_t) - x_m(C_s) \,, \tag{20}$$

where $x_m(\cdot)$ is as in Eq. (16). Eq. (17) implies that $\delta(t, s) = \sum_{m=1}^{M} \delta_m(t, s)$. Hence, it may be computed by the sum SMP.

We note that in the main loop, if we move a node from a singleton cluster C_t to another cluster C_s, it is actually a unification of C_t and C_s; in that case player m computes $\delta_m(t, s)$, Eq. (20), rather than $\delta_m(n : t \to s)$, Eq. (18), and then they proceed to compute the corresponding sum.

5.1.5 Computing the clustered social network

After the completion of the distributed implementation of Algorithm 1, the players have the clustering $\mathcal{C}$. At this point, they need to jointly compute the corresponding clustered social network, $\mathcal{SN}_{\mathcal{C}}$, see Definition 2.2. Specifically, for every super-node they need to compute its number of intra-cluster edges, while for every pair of super-nodes it is necessary to compute the number of edges in the original graph that connect nodes in those two clusters. To that end, player m, $1 \leq m \leq M$, computes the following counters for all $1 \leq t \leq s \leq T$,

$$E^m_{t,s} = \sum \{B(n,\ell) : \ n \in \Omega_m, \ell > n, \text{ and } (v_n, v_\ell) \in [C_t \times C_s] \bigcup [C_s \times C_t]\Big\} , \quad (21)$$

where B is the adjacency matrix of the unified graph. By invoking the sum SMP, the players may compute the sum of those counters,

$$E_{t,s} = \sum_{m=1}^{M} E^m_{t,s} = \sum \{B(n,\ell) : \ \ell > n, \text{ and } (v_n, v_\ell) \in [C_t \times C_s] \bigcup [C_s \times C_t]\Big\} , \quad (22)$$

for all $1 \leq t \leq s \leq T$. It is easy to see that $E_{t,t}$ is the number of intra-cluster edges within C_t, while $E_{t,s}$ is the number of inter-cluster edges between C_t and C_s, $1 \leq t \neq s \leq T$. After the completion of those computations, the players have a k-anonymization of the unified network.

5.1.6 Parallelizing summation computations

In each stage in Algorithm 1, there are several candidate clustering decisions. During the main loop, if v_n is a node in cluster C_t, the players need to compute $\sum_{m=1}^{M} \delta_m(n : t \to s)$ for all $s \in [T] \setminus \{t\}$, in order to check all possible alternative clusters for that node. Hence, in order to reduce the time and message communication costs, each player computes the vector $(\delta_m(n : t \to s))_{s \in [T] \setminus \{t\}}$ and then the players engage in a single SMP to securely compute the sum of those vectors. Similarly, in the agglomerative stage, when looking for the best cluster to unify with a given small cluster C_t; each player computes the vector $(\delta_m(t,s))_{s \neq t}$, and then the players engage in a single SMP to securely compute the sum of those vectors, in order to make the optimal decision regarding which cluster to unify with C_t.

5.1.7 A secure multiparty protocol for computing sums

Computing the sum of private integers has well known simple SMPs (e.g. [4], [13]). Here, as mentioned above, it is necessary to add private vectors. The components of the vectors are rational numbers, as can be seen in Eqs. (18) and (20), together with (14) and (16). The denominators of those numbers are common and known to all, but their numerators depend on private integers (those are the private integers that appear in the numerator of Eq. (14)). Hence, that problem reduces to computing sums of private vectors over the integers. Moreover, it is possible to compute upfront an upper bound p on the size of those integers and of their sum. Hence, the problem may be further reduced to computing sums of private vectors over $\mathbb{Z}_p$. Algorithm 2 (due to Benaloh [4]) does that.

Algorithm 2. Secure computation of sums

- Input: Each player m, $1 \leq m \leq M$, has a private input vector $\mathbf{a}_m \in \mathbb{Z}_p^d$.
- Output: $\mathbf{a} = \sum_{m=1}^{M} \mathbf{a}_m$.

1) Player m selects M random share vectors $\mathbf{a}_{m,\ell} \in \mathbb{Z}_p^d$, $1 \leq \ell \leq M$, such that $\sum_{\ell=1}^{M} \mathbf{a}_{m,\ell} = \mathbf{a}_m \bmod p$.
2) Player m sends $\mathbf{a}_{m,\ell}$ to the ℓth player, for all $1 \leq \ell \neq m \leq M$.
3) Player ℓ, $1 \leq \ell \leq M$, computes $\mathbf{s}_\ell = \sum_{m=1}^{M} \mathbf{a}_{m,\ell} \bmod p$.
4) Players ℓ, $2 \leq \ell \leq M$, send $\mathbf{s}_\ell$ to the player 1.
5) Player 1 computes $\mathbf{a} = \sum_{\ell=1}^{M} \mathbf{s}_\ell \bmod p$ and broadcasts it.

5.2 Distributed and centralized implementations of Algorithm 1

By applying an M-distributed version of Algorithm 1 on a social network $\mathcal{SN}$ and anonymity parameter k, we obtain a sequence of clusterings $\sigma = (\mathcal{C}_1, \ldots, \mathcal{C}_z)$, where $\mathcal{C}_1$ is the initial random clustering and $\mathcal{C}_z$ is the final one. That sequence depends on the random selections made by the players in Steps 1 and 3. Let $\Sigma_M(\mathcal{SN}, k)$ denote the set of all possible sequences σ that may be realized during an M-distributed implementation of Algorithm 1 on inputs $\mathcal{SN}$ and k. Then:

Theorem 5.1. *The set $\Sigma_M(\mathcal{SN}, k)$ is independent of M.*

Namely, each sequence of clusterings that can be realized during an M-distributed implementation of Algorithm 1 on given inputs, is a possible sequence also in a centralized implementation ($M = 1$), and vice-versa.

Proof: Let $\sigma = (\mathcal{C}_1, \ldots, \mathcal{C}_z)$ be a sequence of clustering in an M-distributed implementation of Algorithm 1. We prove by induction that every prefix of length i in σ can be realized in any other distribution setting, with any number of players. When $i = 1$, any initial random clustering $\mathcal{C}_1$ can clearly be realized in any setting. If $\mathcal{C}_{i+1}$ is obtained from $\mathcal{C}_i$ without randomization, then $\mathcal{C}_{i+1}$ will follow $\mathcal{C}_i$ in any distribution setting since it is implied by Algorithm 1. If $\mathcal{C}_{i+1}$ is obtained from $\mathcal{C}_i$ by randomization (namely, $\mathcal{C}_{i+1}$ results from applying Step 3 on $\mathcal{C}_i$), then it can be obtained from $\mathcal{C}_i$ in any distribution setting, since any random partition is possible in any distributed setting. Since the termination condition is independent of M, the sequence σ can be realized in any distribution setting. □

5.3 Privacy

A perfectly secure multiparty protocol does not reveal to any of the participating parties more information than what is implied by their own input and the final output. While such perfect security may be theoretically achieved, as was shown by Yao in [25], some relaxations are usually inevitable when looking for practical solutions, provided that the excess information is deemed benign (see examples of such protocols in e.g. [13], [21], [30]). Our protocol is not perfectly secure. In Theorem

5.3 we bound the excess information that it may leak to the interacting players. We then proceed to argue why such leakage of information is benign.

Our protocol invokes the basic SMP for computing sums of private vectors that are held by the players. That SMP is perfectly secure, as stated in the following theorem:

Theorem 5.2. *Algorithm 2 computes the required sum* $\mathbf{a} = \sum_{m=1}^{M} \mathbf{a}_m$. *It offers perfect privacy with respect to semi-honest players in the following sense: Any coalition of players cannot learn from their view of the protocol about the input vectors of other players more information than what is implied by their own input vectors and the final output.*

The proof of Theorem 5.2 is standard and omitted due to page limitations.

Hence, after each invocation of the SMP, the players of any possible coalition learn the sum of the input vectors, but they receive no information on other input vectors beyond what is implied by their own input vectors and the final sum. (In case $M-1$ players collude, they will be able to infer the input vector of the last player; but that is not a limitation of the protocol, as it is a natural consequence of the computation even if it would be executed using a trusted third party.)

The perfect security of the sum SMP implies that whenever it is invoked by the sequential clustering algorithm, the only information that is disclosed to the players is the difference $I'_S(\mathcal{C}) - I'_S(\hat{\mathcal{C}})$ for each of the potential clusterings that were contemplated at that stage. The definition of $I'_S(\cdot)$, Eqs. (8)–(9), implies that the difference $I'_S(\mathcal{C}) - I'_S(\hat{\mathcal{C}})$ is a linear combination of the entries of the distance matrix D.

Prior to the execution of the protocol, each of the players knows only the entries of D that correspond to his nodes; the remaining nodes are unknown to him at that stage. More generally, if some of the players collude (even $M-1$ out of the M players), they know only the entries of D that correspond to nodes under their control, but not the entries corresponding to nodes under the control of players outside the coalition. As described above, during the execution of the distributed protocol, the players learn differences of the form $I'_S(\mathcal{C}) - I'_S(\hat{\mathcal{C}})$. Let $\{I'_S(\mathcal{C}_i) - I'_S(\hat{\mathcal{C}}_i)\}_{i=1}^{q}$ be all differences that the players learnt throughout the execution of the sequential algorithm. Then the information that is being disclosed to the players is q linear equations in the entries $D(n,n')$ of the distance matrix. Those linear equations may enable the players to learn the value of the entries of D which were not known to them a-priori.

If q is smaller than the number of unknowns in the matrix D, then the given under-determined system of linear equations enable to confine the matrix D to some subspace of $\mathbb{R}^{N(N-1)/2}$, whence, any entry $D(n,n')$ that is not determined completely by the available equations can be of any value. However, if q is sufficiently large so that the players, or coalition of players, may assemble a full system of independent linear equations in the unknown entries of D, it is possible to fully recover D. Therefore, our protocol is not perfectly secure, as it may leak to the interacting players excess information that is not implied by their own input and the final output. However, the players cannot learn more than D. Theorem 5.3 summarizes our discussion above by bounding the worst case information leakage:

Theorem 5.3. *At the completion of the distributed sequential clustering protocol, the excess information that the interacting players may learn, even when there are coalitions of size* $M-1$*, is at the worst case the distance matrix* D.

Our simulations on graphs of 1000-4000 nodes, distributed evenly among $M \leq 10$ non-colluding players, showed that q is sufficiently large for $k \leq 30$, but becomes too small for larger values of k. Namely, for small values of k the players will be able to recover all of D, but for larger values of k they might not assemble a sufficient number of linear equations.

Coalitions help to reduce the number of unknown entries in D. Hence, larger coalitions would need to assemble a smaller number of linear equations before being able to recover D. But Theorem 5.3 applies even to coalitions of size $M-1$, in the sense that even such coalitions cannot learn anything beyond the matrix D.

Despite the above described leakage of information, we deem our distributed protocol safe, for two reasons which we proceed to discuss.

The first reason is that the distance matrix (typically) does not surrender the sensitive information which is the link information. Assume that the first player, to whom we refer hereinafter as Bob, wishes to learn whether v_r and v_s are connected or not. (As Bob controls the first N_1 nodes, we assume that $r, s > N_1$.) Then if Bob was able to recover the distance matrix D, he would be able to use it in order to derive a probability for the event that v_r and v_s are connected. (Only rarely that probability would be definitive, namely zero or one; for example, if the distance of v_r to one of the nodes which Bob controls is exactly 0 or 1.) However, a similar type of information is leaked also by the final clustered network. Indeed, if v_r and v_s belong to the same cluster C_t in the final output, then the probability that they are connected is $\frac{2e_t}{|C_t|\cdot(|C_t|-1)}$, where e_t is the number of intra-cluster edges; in case $e_t = 0$ (or $e_t = |C_t| \cdot (|C_t|-1)/2$), the output reveals that v_r and v_s are (not) connected; similarly when v_r and v_s belong to two different clusters. Moreover, as opposed to the probabilities that may be inferred easily from the final clustering, the computational process by which edge probabilities may be derived from the distance matrix D is computationally harder by far. We postpone the discussion of that computational problem to Section 5.3.1 below.

Another reason is that even if it is possible to assemble a full system of linear equations, the number of unknowns may be very large; for example, even for a modest value of $N = 10^4$, the number of unknowns is in the tens of millions. Such systems can be solved only approximately by iterative methods (e.g., Generalized Minimal Residual or Nonlinear Conjugate Gradient methods). The approximation errors of such methods together with the rounding errors might be larger than $1/(N-2)$, which is the resolution of the distance values in D. Larger, and still very reasonable values of N, would lead to systems of linear equations of dimensions that currently cannot be solved, not even approximately.

5.3.1 Deriving edge probabilities from the distances

Assume that the first player, Bob, was able to recover the matrix D. Bob's goal is now to learn whether v_r and v_s (where $r, s > N_1$) are connected ($B(r,s) = 1$) or not ($B(r,s) = 0$). Let us denote the n-th row in the adjacency matrix B by $\mathbf{b}_n$ (namely, $\mathbf{b}_n$ is a binary vector of length N). Bob knows the vectors $\mathbf{b}_n$ for all $1 \leq n \leq N_1$, since those vectors describe the adjacency relations of the nodes under his control. Bob does not know the vectors $\mathbf{b}_r$ or $\mathbf{b}_s$ and he wishes to learn information about the bit $\mathbf{b}_r(s) = \mathbf{b}_s(r)$. The information that he possesses and could be used towards that goal is as follows:

1) $D(r,n) = \text{dist}(\mathbf{b}_r, \mathbf{b}_n)$ and $D(s,n) = \text{dist}(\mathbf{b}_s, \mathbf{b}_n)$ for all $1 \leq n \leq N_1$.
2) The first N_1 components in $\mathbf{b}_r$ and $\mathbf{b}_s$. (As B is symmetric, its first N_1 rows reveal its first N_1 columns.)

Let W_r (respectively, W_s) denote the set of all N-length binary vectors that satisfy the two conditions above about $\mathbf{b}_r$ ($\mathbf{b}_s$). Define $W_r(0)$ ($W_r(1)$) to be the subset of vectors in W_r in which the s-th bit is zero (one). Similarly, $W_s(0)$ ($W_s(1)$) is the subset of vectors in W_s in which the r-th bit is zero (one). Then, since by symmetry of the adjacency matrix $\mathbf{b}_r(s) = \mathbf{b}_s(r)$, the set of all possible values for the ordered pair $(\mathbf{b}_r, \mathbf{b}_s)$ is

$$(W_r(0) \times W_s(0)) \cup (W_r(1) \times W_s(1)) \ .$$

Hence, the probability of the event $\mathbf{b}_r(s) = \mathbf{b}_s(r) = b$, where b is either zero or one, is

$$p_b = \frac{|W_r(b) \times W_s(b)|}{|W_r(0) \times W_s(0)| + |W_r(1) \times W_s(1)|} \ .$$

The number of vectors in the above sets is, in the general case, exponential. Hence, it is practically impossible to derive the probabilities p_0 and p_1 by enumerating all vectors in the sets $W_r(b)$ and $W_s(b)$, $b = 0, 1$. It is not clear to us whether it is possible to compute the size of the sets $|W_r(b) \times W_s(b)|$, $b = 0, 1$, in polynomial time.

5.4 The limitations of k-anonymity

Several studies have pointed out weaknesses of the k-anonymity model in the context of tabular data. The main weakness of k-anonymity is that it does not guarantee sufficient diversity in the private attribute in each equivalence class of indistinguishable records. Machanavajjhala et al. [16] proposed to make sure that the anonymized table respects ℓ-diversity, in the sense that the private attribute in each equivalence class will have at least ℓ "well represented" values.

Similar problems may occur also with the structural information in anonymizing social networks. For example, the nodes in a given cluster C_t may form a clique; in such cases, the corresponding intra-cluster count e_t would equal $\binom{|C_t|}{2}$, whence it would leak the fact that all nodes in the cluster are connected. To avoid such cases, we define the notion of structural ℓ-diversity.

Definition 5.4. *Let $\mathcal{SN} = \langle V, E, \mathcal{R} \rangle$ be a social network and let $\mathcal{SN}_\mathcal{C} = \langle V_\mathcal{C} = \mathcal{C}, E_\mathcal{C}, \overline{\mathcal{R}} \rangle$ be a corresponding clustered social network. $\mathcal{SN}_\mathcal{C}$ respects structural ℓ-diversity, for some $\ell > 0$, if for all clusters $C_t, C_s \in \mathcal{C}$, it holds that $I_{S,1}(C_t) \geq \ell$ and $I_{S,2}(C_t, C_s) \geq \ell$.*

Recall that $I_{S,1}(C_t)$, Eq. (4), is the probability of wrongly identifying a pair of nodes in C_t as an edge or as a non-connected pair. Similarly, $I_{S,2}(C_t, C_s)$, Eq. (5), is the probability of wrongly identifying a pair of nodes in C_t and C_s as an edge or as a non-connected pair. Hence, if $\mathcal{SN}_\mathcal{C}$ satisfies structural ℓ-diversity for some $\ell > 0$, it leaves all edge information in the original graph with uncertainty of at least ℓ.

Algorithm 1 may be modified so that it accepts as an additional input parameter a minimal structural diversity threshold ℓ and then it outputs k-anonymizations of the input network that respect structural ℓ-diversity. To do that, it starts by selecting an arbitrarily initial clustering that respects structural ℓ-diversity and then it rules out contemplated changes in the clustering that would result in violation of that property.

Structural ℓ-diversity may be used by the different players in the distributed implementation of Algorithm 1 in order to protect their private structural information from full disclosure to other players. To that end, each player makes sure, as described above, that the clustered social network, when restricted to his nodes only, respects structural ℓ-diversity. By doing so, he guarantees that even if all other players collude against him, they will not be able to infer links between his nodes with success probability greater than $1 - \ell$.

It is possible that the network of some player cannot be ℓ-"diversified". For example, if all nodes under the control of player 1 are connected to each other, that player could never find a clustering of his own nodes that respects structural ℓ-diversity for any $\ell > 0$. In such cases, that player may choose not to participate. Another possibility is that each player will determine his own level of diversity, ℓ_m, depending on the structure of his private network and the level of protection that the nodes under his control demand.

5.5 Communication complexity

Let L denote the number of iterations in the sequential algorithm. During the main loop, we need to compute for each node the differences in the structural information loss if that node moves to any of the other clusters. As explained in Section 5.1, this may be done by one invocation of an SMP to compute a sum of private vectors (Algorithm 2). Hence, the number of SMP calls in the main loop is NL. In the agglomerative stage that follows, there is a need in one invocation of the SMP for each small cluster. Since Step 5 may be repeated at most N times (and typically much less) the overall number of SMP calls in the entire protocol is bounded by $N(L+1)$. Finally, as Algorithm 2 entails 3 communication rounds, the overall round complexity of the protocol is bounded by $3N(L+1)$.

While the round complexity is independent of M (as implied by Theorem 5.1 and the constant round complexity of Algorithm 2), the bit complexity of the protocol depends on M, since the bit complexity of Algorithm 2 does. In Step 2 of Algorithm 2, each player sends a message to each of the other $M-1$ players, and then, in Step 4, $M-1$

players send a message to player 1. That gives a total of $M(M-1)+(M-1) = M^2-1$ messages of length $d \log_2 p$ each (where d is the number of candidate clustering decisions that the players need to choose from and p is as described in Section 5.1.7).

We have no theoretical bounds on L. However, from experimentation (Section 6), L appears to be weakly correlated with N, and it decreases with k. (In view of the discussion in Section 5.2, L is independent of M.)

6 Experimental Results

6.1 Experimental setup

We tested our algorithms on three types of graphs: A random graph generated by the Watts-Strogatz (WS) model [22]; a random graph generated by the Barabási-Albert (BA) model [3]; and a subset of the DBLP co-authorship graph. Table 1 describes all of the graphs that we tested — their type, number of nodes, and number of edges.

Graph	Type	$\|V\|$	$\|E\|$
WS1	WS	1000	20000
WS2	WS	2000	40000
WS3	WS	4000	80000
BA1	BA	1000	1954
BA2	BA	2000	3920
BA3	BA	4000	8072
DBLP1	DBLP	1000	2534
DBLP2	DBLP	2000	6110
DBLP3	DBLP	4000	12304
DBLP4	DBLP	8000	26302
DBLP5	DBLP	16000	78328

TABLE 1
Graph types and sizes

The descriptive data was extracted from the CENSUS[2] dataset; that data consists of 7 attributes (`age`, `gender`, `education level`, `marital status`, `race`, `work class`, `country`).

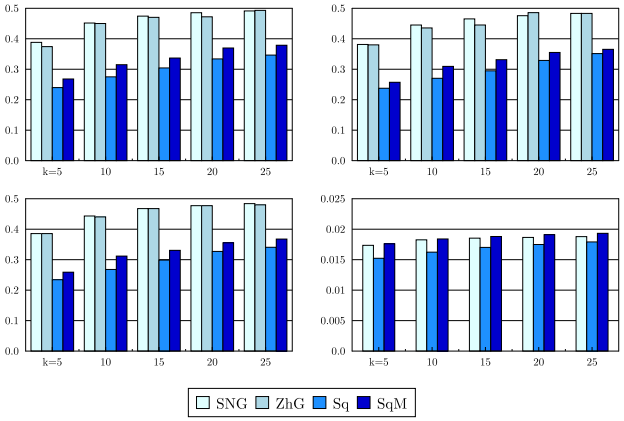

Fig. 2. Average information losses in the WS, BA and DBLP graphs

We tested the two versions of sequential clustering; the original version is denoted in the plots by Sq, while the modified one (that uses I'_S instead of I_S) is denoted by SqM. We compared them against the SaNGreeA algorithm due to Campan and Truta [6] (denoted SNG) and the cluster-edge algorithm of Zheleva and Getoor [29] (denoted ZhG).[3] Our code was implemented in C-sharp and ran on an Intel Quad Core 2.8GHz machine with 2GB of RAM.

6.2 Measuring the information loss

Figure 2 displays the average information losses $I(\cdot)$ (Eq. (1)) that were achieved by those algorithms on the WS (upper left), BA (upper right), and DBLP (lower left) graphs with $w = 0.5$ and the WS graph with $w = 0$ (lower right), where the average is over the results that were obtained on the graphs with 1000, 2000, and 4000 nodes. As can be seen, both variants of sequential clustering always achieve significantly better results than SaNGreeA and the Zheleva-Getoor algorithm when $w = 0.5$. When $w = 0$, the sequential clustering algorithm still issues the best results, but the modified version issues comparable results to SaNGreeA. (The Zheleva-Getoor algorithm is irrelevant for the case $w = 0$ since it totally ignores the structural data.)

6.3 Graph statistics

Next, we measured certain graph statistics in the original graph and compared them to measurements of the same statistics extracted from the anonymized graphs. The statistics that we measured were the *clustering coefficient*, (i.e., the fraction of closed triplets of nodes among all connected triplets), the *average distance* among pairs of nodes, the *diameter* (i.e., the maximum distance among pairs of nodes), the *effective diameter* (the 90th percentile distance, i.e., the minimal value for which 90% of the pairwise distances in the graph are no larger than), and the *epidemic threshold* (the inverse of the largest eigenvalue of the adjacency matrix of the graph).

To extract the statistics from an anonymized clustered graph, we first sampled from it a full graph on N nodes by randomly selecting from it one of the possible original graphs, and then we measured the statistics on that graph. In other words, given an anonymized clustered graph (Definition 2.2) with clusters $\mathcal{C} = \{C_1, \ldots, C_T\}$, cluster sizes $|C_t|$, intra-cluster edge counts e_t, and inter-cluster edge counts $e_{t,s}$, where $1 \leq t \neq s \leq T$, we generated a graph on $N = \sum_{t=1}^{T} |C_t|$ nodes in the following manner: First, we separated the N nodes into clusters of the given sizes; then, within each cluster C_t we randomly connected e_t pairs of nodes by edges; finally, between every pair of clusters, C_t and C_s, we randomly connected $e_{t,s}$ of the pairs of nodes. We then measured the required statistics on that graph. In our experiments we repeated the above process of random sampling and statistics measuring ten times, and we report the average value and the corresponding standard deviation.

Figures 3, 4 and 5 include the five statistics measurements on the WS, BA and DBLP graphs with 4000 nodes. They show the value of each of those statistics in the original graphs and the average value and standard deviation of the

2. http://www.ipums.org/

3. We did not compare against the clustering algorithm which was outlined in [9] since its description omitted critical ingredients such as how the candidate neighboring clustering were selected, the cooling schedule, and the dependence of the probability of making bad choices on the temperature.

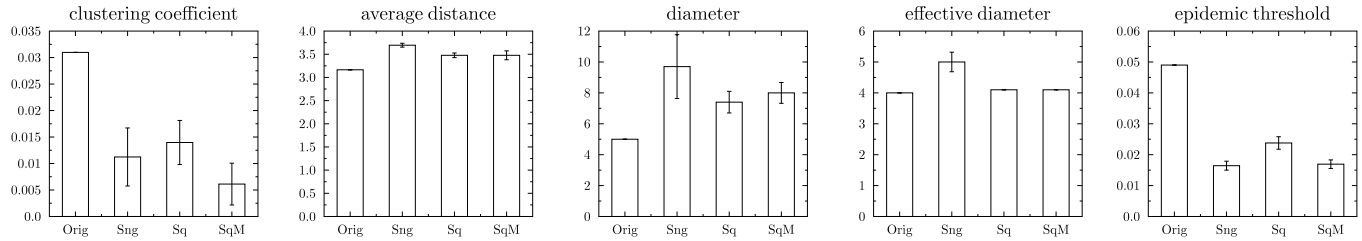

Fig. 3. Statistics comparison for the WS graph

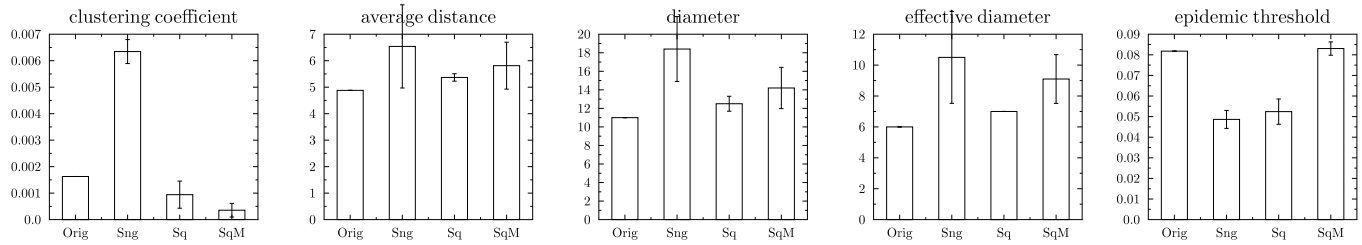

Fig. 4. Statistics comparison for the BA graph

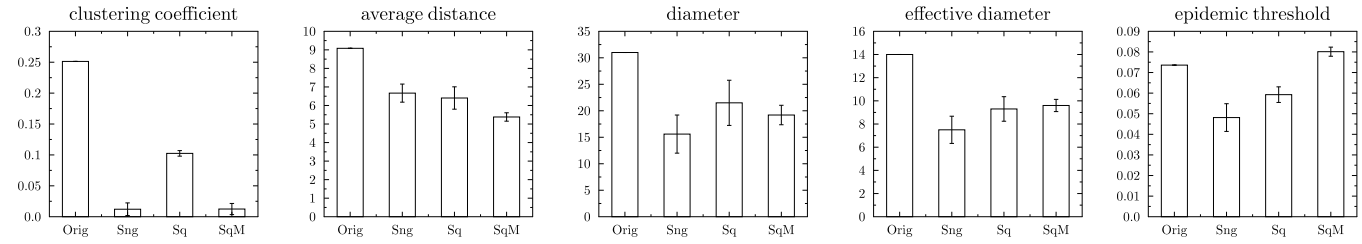

Fig. 5. Statistics comparison for the DBLP graph

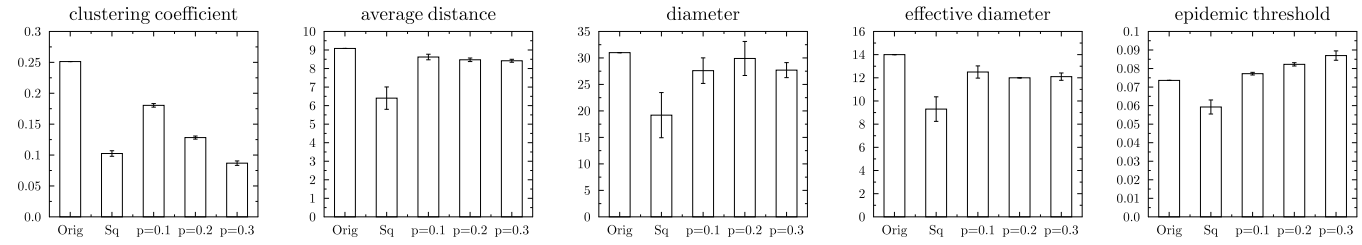

Fig. 6. Statistics comparison for the DBLP graph against perturbed graphs

measurement of the same statistics that were obtained from anonymizations by clustering of that graph, with $k = 10$. We see that the non-modified sequential clustering (Sq) almost always gave the best results; the only exceptions are the epidemic threshold in the BA and DBLP graphs in which the modified sequential clustering issued the value that was closest to the actual clustering coefficient, and the average distance in the DBLP graph, where the SaNGreeA algorithm issued a slightly better value. The SaNGreeA algorithm issued results that were almost always furthest from the correct values. The only two exceptions are the case that was mentioned above and the clustering coefficient in the WS graph in which it issued a better result than SqM (but still less good than that of Sq). In addition, the SaNGreeA algorithm appears to be more sensitive to random samplings: In most experiments, the standard deviation in the SaNGreeA experiments was highest.

6.4 Comparison to anonymization by perturbation

Here, we compare the utility of graphs that were anonymized by clustering to that of graphs that were anonymized by means of random perturbations. There are two main graph perturbation strategies: random addition and deletion of edges,

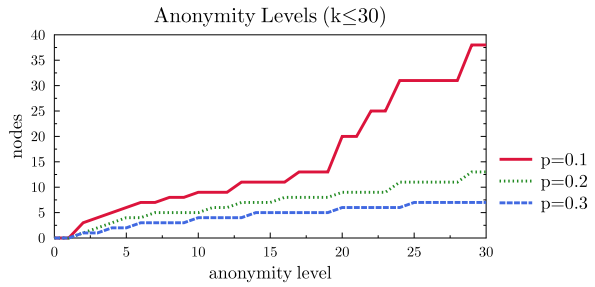

Fig. 7. Anonymity levels

and random switching of edges. In the first strategy one deletes every existing edge with probability p and adds every non-existing edge with probability $q = q(p)$ that is chosen so that the expected number of edges remains like in the original graph [5], [10], [26]. In the second strategy, one selects h quadruples of nodes $\{u, v, x, y\}$ where (u, v) and (x, y) are edges and (u, y) and (v, x) are not, and switches between them, so that the two former edges become non-edges and the two latter non-edges become edges [26], [27], [28]; such techniques preserve the degree of all nodes in the graph. We concentrate here on the first strategy. Figure 6 shows the five statistics that were measured in the previous section in anonymizations of the DBLP graph with 4000 nodes (DBLP3), as achieved by the sequential clustering algorithm, and by random perturbations with different values of p.

Such a comparison must be made between anonymizations that provide a similar level of anonymity. Alas, it is not straightforward to compare the anonymity levels in those two different models of anonymization. While anonymity by clustering, much like k-anonymity, provides a clear and *uniform* notion of security (each node is indistinguishable, in the information theoretic sense, from at least $k - 1$ others), anonymity by random perturbations is more elusive and highly non-uniform. One must assume some property that the adversary knows about his target node and then the level of anonymity of a given node (with respect to that property) is defined as the inverse of the maximal probability of linking that node to a given node in the anonymized graph [5], [10]. Figure 7 shows the anonymity levels with respect to the degree property; for each level $k \leq 30$, it shows how many nodes in the anonymized graph failed to achieve it. As expected, higher values of the perturbation parameter p obfuscate better the original degrees of the nodes and hence provide higher anonymity levels. However, there are few nodes in the graph (usually hubs with high degree) that defy anonymity since their degree is far off from the degrees of most other nodes, and, consequently, they remain distinguishable even in the perturbed graph. In particular, the perturbation method fails to provide a minimal anonymity level of $k = 10$ (the value that was used in the sequential clustering) even with the high perturbation parameter of $p = 0.3$.

6.5 Runtime

Here we report runtime-related results. Figure 8 shows the runtimes of the SNG, Sq and SqM algorithms on the DBLP graph with $N = 4000$ nodes (DBLP3), as a function of k. Figure 9 shows the runtimes of the SNG, Sq and SqM algorithms on the DBLP graphs, for $k = 25$, as a function of N, ranging from $N = 1000$ to $N = 16000$.

Figure 10 shows the average values of L (the number of iterations in the main loop of the sequential clustering) on the three graphs BA, WS and DBLP, as a function of k and N. Figure 11 shows the corresponding average numbers of SMP calls in a distributed implementation of that algorithm. (Recall that the number of SMP calls is independent of the number of interacting players.)

As expected, Sq has the largest runtime, and SqM offers a significant improvement. As anonymization is applied on data repositories that were collected during a long span of time, runtime is a secondary consideration. Hence, an anonymization algorithm that runs several hours is preferable over an algorithm that runs several minutes but issues anonymized views with less utility. Therefore, the non-modified sequential clustering algorithm (Sq) appears to be the algorithm of choice when the network size allows reasonable runtimes. In larger networks, the modified sequential clustering algorithm (SqM) could be used instead, as it still offers a great advantage in terms of utility when compared to SaNGreeA.

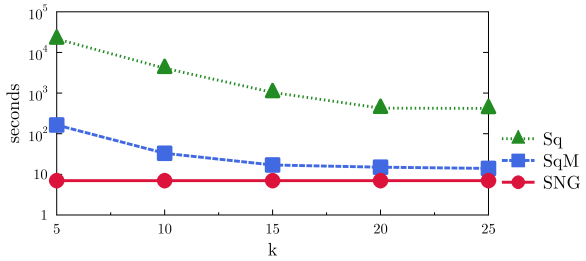

Fig. 8. Runtime vs. k on the DBLP graph ($N = 4000$)

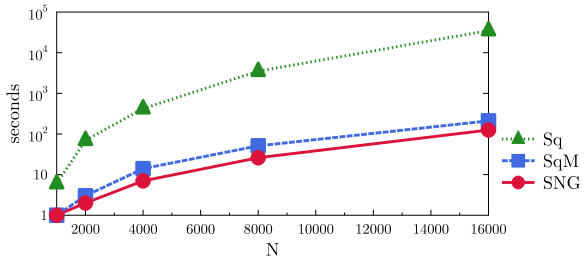

Fig. 9. Runtime vs. N on the DBLP graph ($k = 25$)

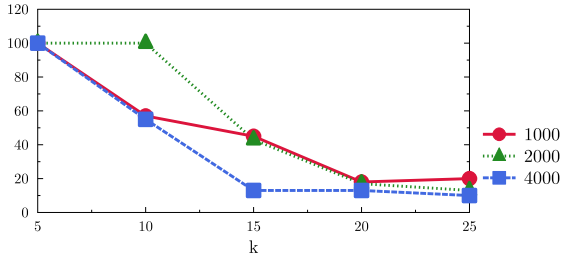

Fig. 10. Number of iterations (L)

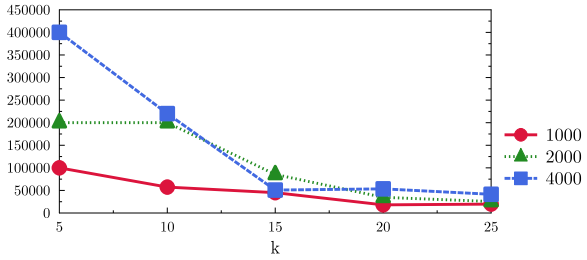

Fig. 11. Number of SMP calls

7 Conclusions

We presented sequential clustering algorithms for anonymizing social networks. Those algorithms produce anonymizations by means of clustering with better utility than those achieved by existing algorithms. We devised a secure distributed version of our algorithms for the case in which the network data is split between several players. We focused on the scenario in which the interacting players know the identity of all nodes in the network, but need to protect the structural information (edges) of the network (Scenario B, as defined in Section 5).

One research direction that this study suggests is to devise distributed algorithms also to Scenario A. In that scenario, each of the players needs to protect the identity of the nodes under his control from the other players. Hence, it is more difficult than Scenario B in two manners: It requires a secure computation of the descriptive information loss (while in Scenario B such a computation can be made in a public manner); and the players must hide from other players the allocation of their nodes to clusters.

Another research direction that this study suggests is to devise distributed versions of the k-anonymity algorithms in [15], [23], [31]; those algorithms might require different techniques than those used here.

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, volume 3363 of LNCS, pages 246–258, 2005.

[2] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190, 2007.

[3] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[4] J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Crypto*, pages 251–260, 1986.

[5] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. In *ICDE*, pages 924–935, 2011.

[6] A. Campan and T. M. Truta. Data and structural k-anonymity in social networks. In *PinKDD*, pages 33–54, 2008.

[7] J. Goldberger and T. Tassa. Efficient anonymizations with enhanced utility. *TDP*, 3:149–175, 2010.

[8] S. Hanhijärvi, G. Garriga, and K. Puolamaki. Randomization techniques for graphs. In *SDM*, pages 780–791, 2009.

[9] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. In *PVLDB*, pages 102–114, 2008.

[10] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *Uni. of Massachusetts Technical Report*, 07(19), 2007.

[11] V. Iyengar. Transforming data to satisfy privacy constraints. In *ACM-SIGKDD*, pages 279–288, 2002.

[12] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *The VLDB Journal*, 15:316–333, 2006.

[13] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16:1026–1037, 2004.

[14] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[15] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD Conference*, pages 93–106, 2008.

[16] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[17] M. E. Nergiz and C. Clifton. Thoughts on k-anonymization. In *ICDE Workshops*, page 96, 2006.

[18] A. Schuster, R. Wolff, and B. Gilburd. Privacy-preserving association rule mining in large-scale distributed systems. In *CCGRID*, pages 411–418, 2004.

[19] N. Slonim, N. Friedman, and N. Tishby. Unsupervised document classification using sequential information maximization. In *SIGIR*, pages 129–136, 2002.

[20] L. Sweeney. Uniqueness of simple demographics in the U.S. population. In *Laboratory for International Data Privacy (LIDAP-WP4)*, 2000.

[21] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644, 2002.

[22] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:409–410, 1998.

[23] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang. k-Symmetry model for identity anonymization in social networks. In *EDBT*, pages 111–122, 2010.

[24] X. Wu, X. Ying, K. Liu, and L. Chen. A survey of privacy-preservation of graphs and social networks. In C. Aggarwal and H. Wang, editors, *Managing and mining graph data*, chapter 14. Springer-Verlag, first edition, 2010.

[25] A. Yao. Protocols for secure computation. In *Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.

[26] X. Ying and X. Wu. Randomizing social networks: A spectrum preserving approach. In *SDM*, pages 739–750, 2008.

[27] X. Ying and X. Wu. Graph generation with prescribed feature constraints. In *SDM*, pages 966–977, 2009.

[28] X. Ying and X. Wu. On link privacy in randomizing social networks. In *PAKDD*, pages 28–39, 2009.

[29] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationship in graph data. In *PinKDD*, pages 153–171, 2007.

[30] S. Zhong, Z. Yang, and R. Wright. Privacy-enhancing k-anonymization of customer data. In *PODS*, pages 139–147, 2005.

[31] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515, 2008.

Tamir Tassa is with the Department of Computer Science at The Open University of Israel. Previously, he served as a lecturer and researcher in the School of Mathematical Sciences at Tel Aviv University, and in the Department of Computer Science at Ben Gurion University. During the years 1993-1996 he served as an assistant professor of Computational and Applied Mathematics at UCLA. He earned his Ph.D. in applied mathematics from the Tel Aviv University in 1993. His research interests include cryptography, privacy preserving data publishing and data mining.

Dror J. Cohen is a senior developer in Microsoft Research and Development, Israel. He earned his B.Sc. in computer science in 2005 and his M.Sc. in computer science in 2011, both from the Open University of Israel. He is currently completing his studies towards a bachelor degree in philosophy.