3I2-OS-20-2

Secure Clustering in Private Networks

Bin Yang*1 Issei Sato*2 Hiroshi Nakagawa*2

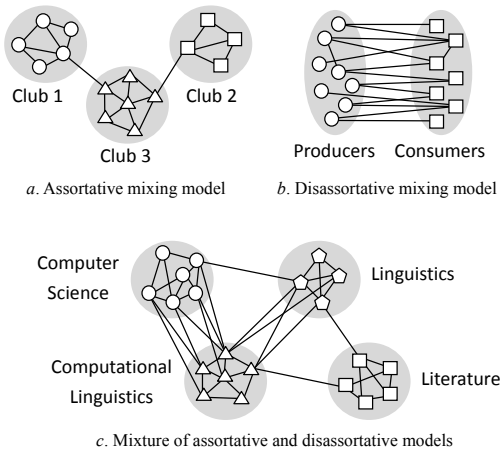
*1 Graduate School of Information Science and Technology, The University of Tokyo
*2 Information Technology Center, The University of Tokyo

Many clustering methods have been proposed for analyzing the relations inside networks with a mixture of assortative and disassortative structures. All these methods are based on the fact that the entire network is observable. However, the entities in some real networks may be private, and thus, cannot be observed. We focus on private peer-to-peer networks in which all vertices are independent and private, and each vertex only knows about itself and its neighbors. We propose a privacy-preserving Gibbs sampling for clustering these types of private networks and detecting their mixed structures without revealing any private information about any individual entity. Moreover, the running cost of our method is related only to the number of clusters and the maximum degree, but is nearly independent of the number of vertices in the entire network.

1. Introduction

Numerous methods have been recently proposed to solve network-clustering problems. Some deal with the assortative mixing model, in which the vertices are divided into groups such that the members of each group are mostly connected to the other members of the same group (Figure 1-*a*). Inversely, in the so-called disassortative mixing model, the vertices have most of their connections outside their group (Figure 1-*b*). A combination of these two models in many applications is more meaningful than using just one, i.e., the vertices generally have more connections inside the group, while some vertices may also frequently linked to the vertices in some correlated groups (Figure 1-*c*).

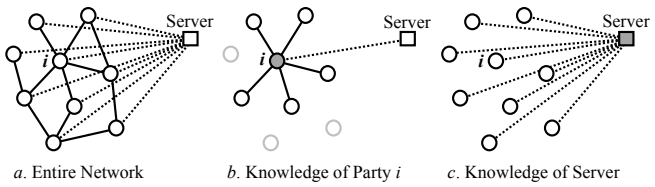
We consider the personal information protection issue in social networks, e.g., each person contacts the others via various means of communication, such as MSN, Yahoo Messenger, mobile phones of different providers, and others. Each of these organizations, such as Microsoft, Yahoo, and mobile service providers, stores part of user's data from the entire network. Hence, some valuable information about the entire network, such as the cluster structure, could not be inferred. This motivated us to develop a new privacy-preserving clustering to be conducted by the individual entities in a network without the support of the organizations. We formulate this type of network as follows.



a. Assortative mixing model *b*. Disassortative mixing model

c. Mixture of assortative and disassortative models

Figure 1: Assortative mixing vs. Disassortative mixing.



a. Entire Network *b*. Knowledge of Party *i* *c*. Knowledge of Server

Figure 2: A Private Peer-to-peer Network.

1.1 Fully Distributed (Peer-to-peer) Private Network Model (Figure 2)

1.1.1 Distributed Assumptions

One vertex one party: We take into consideration a peer-to-peer network in which each party only contains a single vertex. So, the network becomes an n-party system, where n is the number of all vertices.

Local communication: Each vertex only contains knowledge about itself and its neighbors. It does not even know of the existence of parties other than its neighbors.

Fairness: Each vertex has the same status and performs the same operations.

Auxiliary server: An auxiliary server stores the intermediate and final results, and has a connection to each party.

1.1.2 Private Assumptions

Local knowledge: Each vertex only knows about itself and the connections to its neighbors. In particular, it does not even know: (i) the internal information of any other vertex; (ii) the existence of any vertex that is not adjacent; and (iii) the adjacency information about its neighbors.

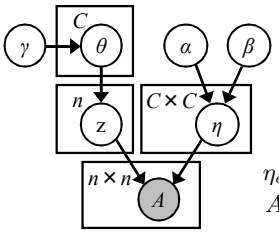
Semi-honest: All the parties properly follow the protocol except that they keeps all the intermediate record.

Non-collusion: The parties do not collude with each other or reveal any information to each other.

Contact: Bin Yang, The University of Tokyo, 7-3-1, Hongou, Bunkyo, Tokyo, yangbin@r.dl.itc.u-tokyo.ac.jp

Table 1: Comparison of related work.

	Secure Method	Partition Manner	Network Data	Problem Type
Kamp[4]	-	-	○	Clustering
Jha[3]	○	Horizon	-	K-means
Vaidya[9]	○	Vertical	-	K-means
Lin[6]	○	Horizon	-	EM-alg.
Hay[2]	○	P2P	○	Degree
Sakuma[8]	○	P2P	○	Ranking
This paper	○	P2P	○	Clustering



$$\begin{aligned} \boldsymbol{\theta}|\gamma &\sim Dirichlet(\gamma) \\ z_i|\boldsymbol{\theta} &\sim Multinomial(\boldsymbol{\theta}) \\ \eta_{ab}|\alpha,\beta &\sim Beta(\alpha,\beta) \\ A_{ij}|\mathbf{z},\eta &\sim Bernoulli(\eta_{z_i z_j}) \end{aligned}$$

Figure 3: Generative model of IRM.

2. Related Work

Kemp et al. [4] proposed a mixed network model with assortative and disassortative structures, *Infinite Relational Model* (*IRM*). A Gibbs sampling method was also proposed to perform the clustering for this model. Many distributed privacy-preserving data mining methods, such as private K-means clustering [3, 9] and private EM algorithm [6], have been proposed. Privacy-preserving data mining used in network analysis has attracted a lot of attention recently [2, 8]. Nevertheless, secure clustering in peer-to-peer networks has not yet received an adequate of attention (Table 1).

3. Preliminaries

An unweighted directed graph $G(V, E)$, where V contains n vertices, and E contains all edges. G can also be represented by an adjacency matrix $\boldsymbol{A}$. See [11] for all the notations. Suppose these n vertices fall into C clusters.

3.1 Infinite Relational Model (IRM)

The graphic model (Figure 3) of *IRM* illustrates the generation of a network, where γ, α, and β are the parameters. The clustering can be done using Gibbs sampling.

$$\Pr(z_i^{new} = z|A, \mathbf{z}_{-i}) \propto \Pr(A|\mathbf{z}^{new}) \Pr(z_i^{new} = z|\mathbf{z}_{-i}), \quad (1)$$

where

$$\Pr(z_i^{new} = z|\mathbf{z}_{-i}) = \begin{cases} \dfrac{n_z - 1 + \gamma}{n - 1 + C \cdot \gamma} & (z = z_i) \\ \dfrac{n_z + \gamma}{n - 1 + C \cdot \gamma} & (z \neq z_i) \end{cases}, \quad (2)$$

$$\Pr(A|\mathbf{z}^{new}) = \prod_{a,b \in [C]} \frac{Beta(m_{ab}^{new} + \alpha, M_{ab}^{new} + \beta)}{Beta(\alpha, \beta)}. \quad (3)$$

We need to compute (1) to sample z_is repeatedly.

Table 2: Distributed Sampling Schema.

01 **Initialization**: The server chooses and publishes α, β, γ;
02 **Accumulation stage** (Section 4.1):
03 The server computes $\boldsymbol{m}$ and $\boldsymbol{n}$ with all parties;
04 **Sampling stage** (Section 4.2):
05 All parties sample and update their own z_i in parallel;
06 If not convergent, goto 02.

In the end, given the final results $\boldsymbol{m}$, $\boldsymbol{M}$, and $\boldsymbol{n}$, we can recover $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, which express the network structure:

$$\theta_a = \frac{n_a}{n}; \quad \eta_{ab} = \frac{m_{ab} + \alpha}{m_{ab} + \alpha + M_{ab} + \beta}. \quad (4)$$

3.2 Tools for Secure Computation

Paillier encryption [7] satisfies an additive homomorphism. Let pk and sk be the public key and the private key. Let m be the plaintext, c be a random encryption of m, and d be the decryption of c. We have

$$E_{pk}(m_1 + m_2, r) \equiv E_{pk}(m_1, r_1) \cdot E_{pk}(m_2, r_2), \quad (5)$$

where r, r_1, and r_2 are random numbers.

A random share generation can be used to generate a private number for each of two parties, such that their summation is equal to some secure value. Both numbers are random so that in isolation they provide zero information about that secure value. Goethals et al. [1] proposed a two-party protocol for the random share generation of a secure product, called SSP. By performing SSP, two random numbers s_a and s_b are securely generated by *Alice* and *Bob*, respectively, such that $s_a + s_b = \boldsymbol{x} \cdot \boldsymbol{y}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the private inputs of *Alice* and *Bob*, respectively.

4. Distributed Clustering

4.1 Distributed Accumulation Stage

We define the contribution of a single party i as a C-size vector $\boldsymbol{\nu}^{(i)}$, whose a^{th} element is $\nu_a^{(i)} = \begin{cases} 1 & (a = z_i) \\ 0 & otherwise \end{cases}$. So we have,

$$\boldsymbol{n} = \sum_i \boldsymbol{\nu}^{(i)}. \quad (6)$$

We also define the contribution of a single edge (i, j) as a $C \times C$ matrix $\boldsymbol{\mu}^{(ij)}$, whose (a, b) element is $\mu_{ab}^{(ij)} = \begin{cases} 1 & (a = z_i, b = z_j) \\ 0 & otherwise \end{cases}$. So we have, $\boldsymbol{\mu}^{(ij)} = \boldsymbol{\nu}^{(i)} \boldsymbol{\nu}^{(j)\mathrm{T}}$, i.e., $\mu_{ab}^{(ij)} = \nu_a^{(i)} \nu_b^{(j)}$, and then,

$$\boldsymbol{m} = \sum_{A_{ij}=1} \boldsymbol{\mu}^{(ij)} = \sum_{A_{ij}=1} \boldsymbol{\nu}^{(i)} \boldsymbol{\nu}^{(j)\mathrm{T}}. \quad (7)$$

If each party i computes a matrix $\boldsymbol{\mu}^{(ij)}$ with each of its children, j, and sends all these $\boldsymbol{\nu}^{(i)}$ and $\boldsymbol{\mu}^{(ij)}$s to the server, the server can compute $\boldsymbol{n}$ (6), $\boldsymbol{m}$ (7), and $\boldsymbol{M}$ (Notice that $M_{ab} = n_a n_b - m_{ab}$).

Table 3: Privacy Schema.

	Private		Publish
	Party i	Server	All
Input	$pa(i)$, $ch(i)$	n	C, γ, α, β
Intermediate	z_i	$\boldsymbol{m}$, $\boldsymbol{M}$, $\boldsymbol{n}$	-
Output	z_i	$\boldsymbol{m}$, $\boldsymbol{M}$, $\boldsymbol{n}$	$\boldsymbol{\theta}$, $\boldsymbol{\eta}$

4.2 Distributed Sampling Stage

The goal of this stage is to compute the probabilities (1): $\Pr(z_i^{new} = z|A, \mathbf{z}_{-i})$, for all $z \in [C]$, and then sample a z_i^{new} for party i from this distribution.

4.2.1 Computation of (2)

Given $\boldsymbol{n}$, $\Pr(z_i^{new} = z|\mathbf{z}_{-i})$ in (2) is a function of z and z_i, i.e,

$$g(z, z_i) \in \Omega_1 := \{g(a, b)|a, b \in [C]\}. \tag{8}$$

4.2.2 Computation of (3)

We set:

$$h_{ab} := \frac{Beta(m_{ab}^{new} + \alpha, M_{ab}^{new} + \beta)}{Beta(\alpha, \beta)}. \tag{9}$$

$$\Pr(A|\mathbf{z}^{new}) = \prod_{a,b\in[C]} h_{ab}. \tag{10}$$

In each step, the cluster label z_i is replaced with z_i^{new} (3.1). For all $a, b \in [C]$, the range of h_{ab} is $\{h_{ab}(\kappa)|a, b \in [C], \kappa \in \{0, 1, \cdots, K\}\}$, which contains at most $(K+1)C^2$ elements.

4.2.3 Computation of (1)

Using (10), we obtain that (1) is equivalent to:

$$p(z) := g(z, z_i) \prod_{a,b\in[C]} h_{ab}, \ (\forall z \in [C]). \tag{11}$$

In our schema, the server computes all possible values for $g(z, z_i)$ and h_{ab}, and sends them to all parties. Each party only needs to choose the corresponding values from these values to compute $p(z)$, without computing them again.

5. Private Clustering

We enhance the security of the schema in Table 2.

5.1 Private Schema

The adjacent matrix $\boldsymbol{A}$ is treated as private input shared with the relevant parties. Each party i only knows its neighbors, $pa(i)$ and $ch(i)$, i.e., party i only knows the i^{th} row and i^{th} column in $\boldsymbol{A}$. Throughout the sampling, $\boldsymbol{m}, \boldsymbol{M}, \boldsymbol{n}$ are stored in the server, while each cluster label z_i is updated by party i. The final results of these values are also regarded as their private outputs. The final outputs $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ reflect the relative size of each cluster and the relations between each cluster pair, and published to all (Table 3).

5.2 Private Accumulation Stage

The $\boldsymbol{\mu}^{(ij)}$ in stage (4.1) cannot be explicitly computed. Nevertheless, it can be protected by using *SSP*, s.t.,

$$\boldsymbol{\mu}^{(ij)} = \boldsymbol{\rho}^{(ij)} + \boldsymbol{\sigma}^{(ij)}, \tag{12}$$

where $\boldsymbol{\rho}^{(ij)}$ and $\boldsymbol{\sigma}^{(ij)}$ are the random matrices known to parties i and j, respectively. Since $\boldsymbol{\mu}^{(ij)} = \boldsymbol{\nu}^{(i)}\boldsymbol{\nu}^{(j)^T}$, i.e., $\mu_{ab}^{(ij)} = \nu_a^{(i)}\nu_b^{(j)}$, so $\rho_{ab}^{(ij)} + \sigma_{ab}^{(ji)} = \nu_a^{(i)}\nu_b^{(j)}$. That is to say, $\boldsymbol{\rho}^{(ij)}$ and $\boldsymbol{\sigma}^{(ij)}$ can be generated by executing the *SSP* for each element of $\boldsymbol{\mu}^{(ij)}$. After the generations for all the edges, each party i sums up all its $\boldsymbol{\rho}^{(ij)}$ and $\boldsymbol{\sigma}^{(ki)}$:

$$\boldsymbol{\rho}^{(i)} = \sum_{(i,j)\in E} \boldsymbol{\rho}^{(ij)} + \sum_{(k,i)\in E} \boldsymbol{\sigma}^{(ki)}. \tag{13}$$

Obviously,

$$\boldsymbol{m} = \sum_i \boldsymbol{\rho}^{(i)}. \tag{14}$$

If each party i sends $\boldsymbol{\rho}^{(i)}$ to the server, $\boldsymbol{m}$ can be accumulated using (14). However, we must keep the server from knowing the value of $\boldsymbol{\rho}^{(i)}$s, since these values may be used to infer the adjacency information of some special vertices, e.g., when a vertex has only one neighbor. To do this, each party i sends $\boldsymbol{X}^{(i)}(= E_{pk}(\boldsymbol{\rho}^{(i)}))$, instead of $\boldsymbol{\rho}^{(i)}$, to the server. From homomorphism (5), the ciphertext of $\boldsymbol{m}$ can be computed by the server without decrypting $\boldsymbol{X}^{(i)}$:

$$E_{pk}(\boldsymbol{m}) = \prod_i \boldsymbol{X}^{(i)}. \tag{15}$$

Since the cryptosystem is $(2, n+1)$-threshold, the server can discover $\boldsymbol{m}$ with any vertex in the network. Similarly, the $\boldsymbol{n}$ can be securely obtained (6). The server computes and encrypts all possible values, $E_{pk}(g(a, b))$ and $E_{pk}(h_{ab}^{(t)}(\kappa))$ $(a, b \in [C])$, and then, publishes all these $(12K + 10)C^2$ ciphertexts to all parties.

5.3 Private Sampling Stage

In the private sampling stage, h_{ab} cannot be obtained by any one, since h_{ab} has at most $K + 1$ possible values, hence its parameter, such as npa_{ia} in $h_{ab}^{(1)}(npa_{ia})$, can be easily recovered from the value of h_{ab}. To solve this problem, we propose a fully secure sampling algorithm, by which a party securely performs a sampling with its neighbors without explicitly computing h_{ab}. See [11] for more detail.

6. Analysis

6.1 Security

Since the distribution of the ciphertext is always a uniform distribution, it can be simulated by anyone. Goethals et al. have guaranteed the security of the *SSP* [1].

Theorem 1 (Security). *The intermediate information in the secure clustering protocol does not reveal any information about each individual vertex in the network.*

6.2 Efficiency

All parties in the network run the protocol in parallel, but each party can simultaneously run its operations with only one neighbor. From Vizing's theorem [10], we can obtain that the total running cost is $O((K + n)C^2)$.

All the above running times are calculated corresponding to one round of sampling. Suppose the sampling needs to be performed t times, then the total running time becomes t times longer than all the above running times.

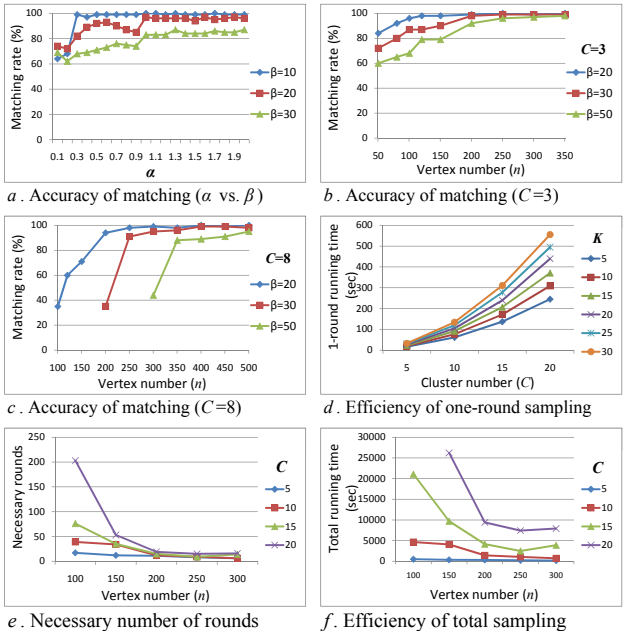


Figure 4: Experiments.

7. Experiments

7.1 Artificial Data

We first evaluate the accuracy of our clustering method by using artificial data generated from *IRM* (Figure 3). The relation between the matching rate and the parameters, α and β, are shown in Figure 4-*a*. The large difference between α and β may lead to a low level of accuracy. Fortunately, using more sample data can increase the accuracy. The relations between the matching rate and the vertex number are shown in Figures 4-*b* and 4-*c*.

We evaluated the running time of our secure protocol with encryption. Figure 4-*d* shows a one-round sampling time with respect to different cluster numbers and maximum degrees. We found that the running time is nearly independent of the vertex number. We also measured the necessary round for convergence and the total running time in Figures 4-*e* and 4-*f*, respectively. Since the necessary round number drastically decreases with an increase in the vertices number, the total running time decreases.

7.2 Real Data

We evaluated a network of books about US politics compiled by Krebs [5]. The nodes represent books about US politics sold by Amazon.com. The edges represent the frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these other books". The nodes were given three labels to indicate whether they are liberal, neutral, or conservative (Figure 5-a). The sampling results are shown in Figure 5-b. This execution of the secure clustering only contained a 12-round sampling and spent about 136 seconds.

a. Original Labeled Data *b. Experimental Results*

Figure 5: Experimental Results of Real Data.

8. Conclusion

We proposed a method to securely cluster vertices in a private network. This method can detect both assortative and disassortative mixing models. Under the assumption that each vertex is independent, private, and semi-honest, our method is secure enough to preserve the privacy of each party. The running cost of our method depends only on the number of clusters and the maximum degree, and therefore, it can be used for analyzing networks in practice.

References

[1] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, *On private scalar product computation for privacy-preserving data mining*, ICISC, 2004.

[2] M. Hay, C. Li, G. Miklau, and D. Jensen, *Accurate estimation of the degree distribution of private networks*, ICDM, 2009.

[3] S. Jha, L. Kruger, and P. McDamiel, *Privacy preserving clustering*, ESORICS, 2005.

[4] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, *Learning systems of concepts with an infinite relational model*, AAAI, 2006.

[5] V. Krebs, *http://www.orgnet.com/*.

[6] X. Lin, C. Clifton, and M. Zhu, *Privacy-preserving clustering with distributed EM mixture*, Knowledge and information systems, 2004.

[7] P. Paillier, *Public-Key Cryptosystems based on Composite Degree Residue Classes*, EuroCrypt, 1999.

[8] J. Sakuma and S. Kobayashi, *Link analysis for private weighted graphs*, SIGIR, 2009.

[9] J. Vaidya and C. Clifton, *Privacy-Preserving k-means clustering over vertically partitioned data*, ACM SIGKDD, 2003.

[10] V. G. Vizing, *On an estimate of the chromatic class of a p-graph*, Diskret. Analiz. 3: 25–30, 1964.

[11] B. Yang, I. Sato, and H. Nakagawa, *Secure Clustering in Private Networks*, ICDM, 2011.