Computation of Ratios of Secure Summations in Multi-party Privacy-Preserving Latent Dirichlet Allocation

Bin Yang[1] and Hiroshi Nakagawa[2]

[1] Graduate School of Information Science and Technology, The University of Tokyo, Japan
yangbin@r.dl.itc.u-tokyo.ac.jp
[2] Information Technology Center, The University of Tokyo, Japan
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract. In this paper, we focus our attention on the problem of Gibbs sampling for privacy-preserving Latent Dirichlet Allocation, which is equals to a problem of computing the ratio of two numbers, both of which are the summations of the private numbers distributed in different parties. Such a problem has been studied in the case that each party is semi-honest. Here we propose a new solution based on a weaken assumption that some of the parties may collaborate together to extract information of other parties.

Keywords: Privacy, Data Mining, Latent Dirichlet Allocation, Collusion.

1 Introduction

In recent years, with the increased concerns on privacy protection of personal information, a number of techniques such as randomization and homomorphic encryption have been suggested in order to perform data mining with the private information preserved. In this paper, we propose a new privacy preserving data mining method with *m* parties from a general problem in data mining: Latent Dirichlet Allocation model (LDA, Blei et al. [1]). LDA is a generative probabilistic model for collections of discrete data such as text corpora, which can be seen as a special mixture model, in which each document is generated by choosing a distribution over topic and then choosing each word in the document from a distribution according to the topic selected. We can employ a Markov chain Monte Carlo algorithm (Griffiths et al. [2]) for inference in this model.

This paper is organized as follows. We introduce the terminology of privacy-preserving data mining and Latent Dirichlet Allocation in section 2. In section 3, we formulate the main problem. And in section 4, we describe our main protocol: RSS protocol, and evaluate the security and efficiency of this protocol. Section 5 illustrates how to assemble our protocol to solve the problem of privacy-preserving LDA described in section 2. In section 6, we present experimental evaluations of these techniques. Finally, we conclude with a summary in section 7.

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 189–197, 2010.

2 Preliminary

Large numbers of conclusions in privacy-preserving data mining have been obtained by researchers, and almost all of them are under the assumption of semi-honest. In particular, one is a semi-honest party means that the party follows the protocol properly with the exception that it keeps a record of all its intermediate computation results and tries to deduce additional information from them other than the protocol result. In this paper, we also consider a problem called collusion that a subset of the participants, a coalition, might get together after the execution of the protocol in an attempt to deduce additional information of non-coalition parties, although every party also follows the protocol properly, as same as in semi-honest assumption.

A protocol is called t-private if no coalition containing at most t parties can get any additional information from its execution.

We will utilize some standard cryptographic tools and secure protocols to preserve privacy in our protocols. For convenience, we name the two parties "Alice" and "Bob" in the case of 2-party system.

Homomorphic encryption. Public key encryption is a fundamental and widely used technology. The asymmetric key algorithm generates a pair of encryption keys – a public key and a private key. The public key used to encrypt a message is different from the private key used to decrypt it. The private key is kept secret, while the public key may be widely distributed. Given a key pair (sk, pk) and a message msg, $c=E_{pk}(msg)$ denotes an encryption of msg, and $d=D_{sk}(c)$ denotes the decryption of c. In particular, we call such an encryption system a homomorphic encryption, if there is an operation *, satisfying the following condition for any msg_1 and msg_2:

$$E_{pk}(msg_1+msg_2) = E_{pk}(msg_1)*E_{pk}(msg_2).$$

Secure linear function evaluation (SLFE). Alice has b, while Bob has c and d. After running the secure linear function evaluation protocol, called $SLFE(b,c,d)$, Alice learns the value of d-bc, and Bob learns *nothing*.

Protocol SLFE:
1.Alice generates pk and sk, computes $E=E_{pk}(-b)$, then sends E and pk to Bob;
2.Bob computes $E'=E_{pk}(d)*E^c$, then sends E' back to Alice;
3.Alice decrypts E' by sk, $D=D_{pk}(E')$.

Here E_{pk} and D_{sk} denote the encrypt function and the decrypt function of a homomorphic encryption system respectively. From the homomorphism, we can get $E^c = E_{pk}(-b)^c = E_{pk}(-b)*E_{pk}(-b)*\ldots*E_{pk}(-b) = E_{pk}(-bc)$, hence $D=d-bc$.

2.1 Latent Dirichlet Allocation

Typically, LDA model is a generative probabilistic model proposed to research text corpora. A document is seen as a sequence of words. A corpus is a collection of M documents, then we can also see a corpus as a sequence of N words, denoted by $\mathbf{w}=(w_1,w_2,\ldots,w_N)$. A document can deal with multiple topics, and the words that appear in that document reflect the particular set of topics it addresses. Each topic is treated as a probability distribution over words in the vocabulary. Thus we can view a

document as a probabilistic mixture of these topics. If there are T topics in total, we can write the probability of the i^{th} word in a document as

$$P(w_i) = \sum_{j=1}^{T} P(w_i \mid z_i = j)P(z_i = j) \tag{1}$$

where z_i is a latent variable indicating the topic from which the i^{th} word w_i was drawn. $P(w_i|z_i=j)$ is the probability of the word w_i under the j^{th} topic, whereas $P(z_i=j)$ is the probability of that the topic of i^{th} word, z_i, is equal to j. In other words, each word is generated by drawing a topic z_i from $P(z)$ first, and then drawing a word from $P(w_i|z_i)$.

2.1.1 Gibbs Sampling for LDA

To infer z_i for each word w_i using Gibbs sampling, we need only compute the posterior probabilities that w_i is assigned to topic j, for j from *1* to T, which were derived by Griffiths et al. [2]:

$$p(i, j) = P(z_i = j \mid \mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta} \cdot \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + T\alpha} \tag{2}$$

$\mathbf{z}_{-i}=(z_1,z_2,\ldots,z_{i-1},z_{i+1},\ldots,z_N)$ is a sequence of N-1 topics corresponding to the sequence of words $\mathbf{w}$ except w_i; $n_{-i,j}^{(w_i)}$ is the number of times that word w_i has been assigned to topic j except the current assignment of z_i; $n_{-i,j}^{(d_i)}$ is the number of times that a word from document d_i has been assigned to topic j except the current assignment of z_i; $n_{-i,j}^{(\cdot)} = \sum_{w=1}^{W} n_{-i,j}^{(w)}$ is the number of times that a word from the corpus has been assigned to topic j except the current assignment of z_i; and $n_{-i,\cdot}^{(d_i)} = \sum_{j=1}^{T} n_{-i,j}^{(d_i)}$ is the number of words in the current document d_i except the current assignment of z_i.

To draw a sampling for a latent variable of a word w_i, we only need to compute T probabilities: $P(z_i=1|\mathbf{z}_{-i},\mathbf{w})$, …, $P(z_i=T|\mathbf{z}_{-i},\mathbf{w})$, each of which means the probability that z_i is equal to j under the condition of $\mathbf{z}_{-i}$ and $\mathbf{w}$.

2.1.2 Distributed LDA

Now we consider a scenario in which more than one parties hold their own data, respectively. They attempt to run Gibbs sampling to infer all z_i with respect to the whole corpus spreading around them without revealing anything. In this case, because any single document is located in just one party, this party can compute $n_{-i,j}^{(d_i)}$ and $n_{-i,\cdot}^{(d_i)}$, thus compute the second item of (2) locally. Then the only remaining problem is computing the first item of (2) in m parties setting. Because the first item of (2) is only related to i and j, we can express it as a function of i and j:

$$q(i, j) = \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta}. \tag{3}$$

Let $n_j^{(w_i)}$ denote the number of times that word w_i has been assigned to topic j including the current assignment of z_i, and $n_j^{(\cdot)} = \sum_{w=1}^{W} n_j^{(w)}$. Then we have:

$$\begin{cases} n_j^{(w_i)} = n_{-i,j}^{(w_i)} + 1 & \text{if } j = z_i \\ n_j^{(w_i)} = n_{-i,j}^{(w_i)} & \text{if } j \neq z_i \end{cases} \tag{4}$$

$$\begin{cases} n_j^{(\cdot)} = n_{-i,j}^{(\cdot)} + 1 & \text{if } j = z_i \\ n_j^{(\cdot)} = n_{-i,j}^{(\cdot)} & \text{if } j \neq z_i \end{cases} \tag{5}$$

When the data size is quite large, we propose the following approximation:

$$q(i,j) \approx q'(w_i, j) = \frac{n_j^{(w_i)} + \beta}{n_j^{(\cdot)} + W\beta}. \tag{6}$$

Here we notice that $q'(w_i,j)$ is dependent only on w_i and j but not on i. And because the number of words in vocabulary is less than the number of words in corpus, we attempt to approximate $q(i,j)$ by $q'(w_i,j)$, then draw a sampling with privacy being protected. From (4) and (5),

$$q'(w_i, j) = \begin{cases} \dfrac{n_{-i,j}^{(w_i)} + 1 + \beta}{n_{-i,j}^{(\cdot)} + 1 + W\beta} & \text{if } j = z_i \\ q(i,j) & \text{if } j \neq z_i \end{cases} \tag{7}$$

which means that this kind of approximation always make $q'(w_i,j)$ keep the value of $q(i,j)$ except that $q'(w_i,j)$ is a little bigger than $q(i,j)$ when j is equal to current topic. This means that the value of z_i is a little harder to be changed when sampling. Hence, it will somewhat cause a slow convergence. However, this kind of approximation always does keep the convergence in Gibbs sampling because of the following reasons: 1) It is so near to the real value if data size if big enough; 2) The second term of right hand side of formula (2) is still intact. 3) When converged, this approximation holds because all the values of probabilities are proportional to the real values except what corresponds to current topic. This will be verified by experiments in section 6.

In the rest of the paper, the index attached to the left-top of a variable denotes the party which the variable belongs to. For instance, ${}^k n_j^{(w)}$ denotes the number of times that word w has been assigned to topic j in all documents in party k. So we can get:

$$n_j^{(w_i)} = \sum_{k=1}^{m} {}^k n_j^{(w_i)} \text{ and } n_j^{(\cdot)} = \sum_{k=1}^{m} {}^k n_j^{(\cdot)} \tag{8}$$

The variables with nothing being attached to the left-top denote the variables corresponding to the whole m parties, as shown in (8).

To draw a sampling for a word w, we need to compute $\mathbf{q'}_w=(q'(w,1),\ldots, q'(w,T))^{\mathrm{T}}$. Hence, it is sufficient for sampling all the words in the corpus that we only compute $\mathbf{q'}_w$ for w from 1 to W. In other words, we need not compute anything for all words in the entire corpus. Therefore, our whole strategy of sampling is a 2-step iteration: one step is securely computing $\mathbf{q'}_w$ for w from 1 to W, and the other step is locally sampling each word in each party using $\mathbf{q'}_w$. Since sampling step is similar to original LDA, we concentrate our attention on the computation of $\mathbf{q'}_1,\ldots,\mathbf{q'}_W$, expressed as:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q'}_1 & \mathbf{q'}_2 & \cdots & \mathbf{q'}_W \end{bmatrix} = \begin{bmatrix} q'(1,1) & q'(2,1) & \cdots & q'(W,1) \\ q'(1,2) & q'(2,2) & \cdots & q'(W,2) \\ \vdots & \vdots & \ddots & \vdots \\ q'(1,T) & q'(2,T) & \cdots & q'(W,T) \end{bmatrix} \tag{9}$$

3 Problem Formulation

The problem of privacy-preserving LDA encountered the problem that how to securely compute the ratio of two summations of the data spreading around m parties with nothing of each party being revealed. We formulate our problem as below.

Problem of RSS (Ratio of Secure Summations):

Input:

Party 1 has a pair of data $({}^1x, {}^1y)$;
Party 2 has a pair of data $({}^2x, {}^2y)$;
……
Party m has a pair of data $({}^mx, {}^my)$.

Output:

All the m parties collaborate to compute:

$$r = \frac{\sum_{i=1}^{m} {}^i x}{\sum_{i=1}^{m} {}^i y} \tag{10}$$

Condition:

Neither ${}^ix$ nor ${}^iy$ of party i is revealed to any party other than i.
Moreover, $\sum_{i=1}^{m} {}^i x$ and $\sum_{i=1}^{m} {}^i y$ are revealed to nobody.

Here it is worthy emphasizing that although the ratio of summations r is known to each party after this computation, ${}^ix$ and ${}^iy$ and any other function of them could not be evaluated by any party other than i.

4 Protocol

We propose a symmetric protocol solving the RSS problem.

Stage 1. Each party generates $2m+1$ random numbers in this stage.

Protocol 2-1:

1. Each party defines $2m+1$ variables of :
Party 1: ${}^1b_1, {}^1b_2, \ldots, {}^1b_m, {}^1c_1, {}^1c_2, \ldots, {}^1c_m, {}^1d$;
Party 2: ${}^2b_1, {}^2b_2, \ldots, {}^2b_m, {}^2c_1, {}^2c_2, \ldots, {}^2c_m, {}^2d$;
……
Party m: ${}^mb_1, {}^mb_2, \ldots, {}^mb_m, {}^mc_1, {}^mc_2, \ldots, {}^mc_m, {}^md$.
And then generates a random value uniformly to each variable, except ${}^1b_1, {}^2b_2, \ldots, {}^mb_m$ and ${}^1c_1, {}^2c_2, \ldots, {}^mc_m$ are reserved to be decided;
2. For each party i, where i from 1 to m:
2-1. Party i runs $SLFE({}^ib_j, {}^jc_i, {}^jd)$ with the collaboration of party j where $j=1,\ldots,i-1,i+1,\ldots,m$, hence computes the values: ${}^ie_j={}^jd-{}^ib_j\,{}^jc_i$, where $j=1,\ldots,i-1,i+1,\ldots,m$;
2-2. Party i views ${}^ib_i{}^ic_i$ as a single variable, and decides the value of ${}^ib_i{}^ic_i$ from the following equation, where only ${}^ib_i{}^ic_i$ is unknown:

$${}^ie_1+{}^ie_2+\ldots+{}^ie_{i-1}+({}^id-{}^ib_i{}^ic_i)+{}^ie_{i+1}+\ldots+{}^ie_m=0.$$

Therefore, each party generated $2m+1$ numbers satisfying the following condition:

$$\sum_{i=1}^{m} {}^{j}b_i\,{}^{i}c_j = d \quad (j=1,2,\ldots,m), \tag{11}$$

where

$$d={}^{1}d+{}^{2}d+\ldots+{}^{m}d. \tag{12}$$

In next stage, We only use the value of ${}^{i}b_i{}^{i}c_i$ but not ${}^{i}b_i$ and ${}^{i}c_i$, so we do not compute the values of ${}^{i}b_i$ and ${}^{i}c_i$, even though they are two variables actually.

Stage 2. In this stage, each party collaborate to compute the value of $\sum_{j=1}^{m} {}^{j}x \cdot d$.

Protocol 2-2:

1. For i from 1 to m, party i computes ${}^{i}b_1{}^{i}x$, ${}^{i}b_2{}^{i}x$,…, ${}^{i}b_m{}^{i}x$, and sends ${}^{i}b_1{}^{i}x$ to party 1, ${}^{i}b_2{}^{i}x$ to party 2, …, ${}^{i}b_m{}^{i}x$ to party m, respectively;
2. For i from 1 to m, party i receives ${}^{1}b_i{}^{1}x$ from party 1, ${}^{2}b_i{}^{2}x$ from party 2, … , ${}^{m}b_i{}^{m}x$ from party m, respectively, and then computes ${}^{1}b_i{}^{i}c_1{}^{1}x$, ${}^{2}b_i{}^{i}c_2{}^{2}x$, …, ${}^{m}b_i{}^{i}c_m{}^{m}x$ locally, by multiplying ${}^{i}c_1$, ${}^{i}c_2$,…, ${}^{i}c_m$, to each number respectively. Then summates these m numbers to ${}^{i}f$ and publish it.
$$ {}^{i}f={}^{1}b_i{}^{i}c_1{}^{1}x+{}^{2}b_i{}^{i}c_2{}^{2}x+\ldots+{}^{m}b_i{}^{i}c_m{}^{m}x $$
3. All parties compute the summation f,
i.e. $f={}^{1}f+{}^{2}f+\ldots+{}^{m}f$.

In the case that $i=j$, since ${}^{i}b_i{}^{i}c_i$ is known to party i, ${}^{i}b_i{}^{i}c_i{}^{i}x$ can be computed locally. Actually, f is the summation of ${}^{j}b_i{}^{i}c_j{}^{j}x$, for i and j from 1 to m. In addition, from (11), we can get that f is just what we want:

$$\begin{aligned} f &= \sum_{i=1}^{m}\sum_{j=1}^{m} {}^{j}b_i\,{}^{i}c_j\,{}^{j}x = \sum_{j=1}^{m}\sum_{i=1}^{m} {}^{j}b_i\,{}^{i}c_j\,{}^{j}x \\ &= \sum_{j=1}^{m} {}^{j}x \sum_{i=1}^{m} {}^{j}b_i\,{}^{i}c_j = \sum_{j=1}^{m} {}^{j}x \cdot d \end{aligned} \tag{13}$$

Stage 3. In terms of ${}^{i}y$, we achieve the operators in stage 1 to regenerate another series of bs, and cs, with the values of ${}^{1}d, {}^{2}d,\ldots, {}^{m}d$ not being changed, such that

$$\sum_{i=1}^{m} {}^{j}b'_i\,{}^{i}c'_j = d \quad (j=1,2,\ldots,m),$$

where $d={}^{1}d+{}^{2}d+\ldots+{}^{m}d$. Hence we can compute the following value by protocol 2-2:

$$f' = \sum_{j=1}^{m} {}^{j}y \cdot d \tag{14}$$

Therefore, the value of f/f' is just the ratio r in (10), which we want to evaluate:

$$\frac{f}{f'} = \frac{\sum_{j=1}^{m} {}^{j}x \cdot d}{\sum_{j=1}^{m} {}^{j}y \cdot d} = \frac{\sum_{j=1}^{m} {}^{j}x}{\sum_{j=1}^{m} {}^{j}y} \tag{15}$$

The value $d={}^{1}d+{}^{2}d+\ldots+{}^{m}d$ is known to nobody unless all the m parties publish their own ${}^{i}d$. Thus all parties can deduce nothing. In addition, we achieve stage 1 for only two times, and achieve stage 2 repeatedly.

Security. Following theorem ensures the security of the RSS protocol.

Theorem. The RSS protocol is (m-1)-private.

Performance. For convenience, we denote t_a, t_c, t_r, t_e and t_d the time of arithmetic, communication, random number generation, encryption and decryption, respectively.

Because of the symmetry of RSS protocol, all parties can execute the protocol in parallel. Thus the running time of one party is equal to the global running time. We summarize the running time of RSS protocol in Table 1.

Table 1. The running time of the RSS protocol

	Stage1	Stage2	Stage3	Total
t_a	$2m$	$5m$	1	$14m+1$
t_r	$2m$	m	-	$6m$
t_e	$2m$	-	-	$4m$
t_d	m	-	-	$2m$
t_c	$2m$	$3m$	-	$10m$

In summary, The running time with respect to each operation is $O(m)$.

To compute R ratios, RSS protocol needs just two times stage 1, $2R$ times stage 2 and R time stage 3. Thus we summarize the result in Table 2.

Table 2. Running time of the RSS protocol corresponding to the R times of computations

t_a	t_r	t_e	t_d	t_c
$O(Rm)$	$O(Rm)$	$O(m)$	$O(m)$	$O(Rm)$

Compared with t_c, t_e and t_d, the values of t_a and t_r are so small that we can ignore it in our protocols. So we can say that t_e and t_d are $O(m)$ in RSS protocol.

5 Implementations and Performance

Our main task of distributed LDA is to securely compute a matrix **Q** defined by (9) which contains WT ratios of secure summations defined by (6). We can utilize our protocol to compute one ratio in terms of w by setting the variables as follows:

$$\begin{aligned} {}^1x &= {}^1n_j^{(w)} + \frac{\beta}{m}, {}^2x = {}^2n_j^{(w)} + \frac{\beta}{m}, \ldots, {}^mx = {}^mn_j^{(w)} + \frac{\beta}{m}, \\ {}^1y &= {}^1n_j^{(\cdot)} + \frac{W\beta}{m}, {}^2y = {}^2n_j^{(\cdot)} + \frac{W\beta}{m}, \ldots, {}^my = {}^mn_j^{(\cdot)} + \frac{W\beta}{m}. \end{aligned} \tag{16}$$

Hence we can securely compute one ratio as

$$\frac{{}^1x + {}^2x + \ldots + {}^mx}{{}^1y + {}^2y + \ldots + {}^my} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + W\beta} \tag{17}$$

Sampling can be drawn, after all the WT values in **Q** are computed.

6 Experiments

In our 2-step iteration of distributed LDA, sampling step is as same as original LDA. As we have mentioned, the approximation in our method makes convergence slower than original LDA. To evaluate it, we compare the results of these two methods in the same machine. The experiment data is a corpus with 300 documents. All documents contain about 200,000 words. The number of words in vocabulary is W=20, where the words occurring in almost every document are omitted. The number of topics is T=5. Here we just evaluate and compare generated assignments of both methods. As shown in Figure 1, more than 167,000 data are changed in first round of both methods. Original LDA converges after about 10^{th} round, while approximate LDA converges after about 18^{th} round. The final results of sampling of both methods are also the same as each other in terms of generated assignments corresponding to each of them.

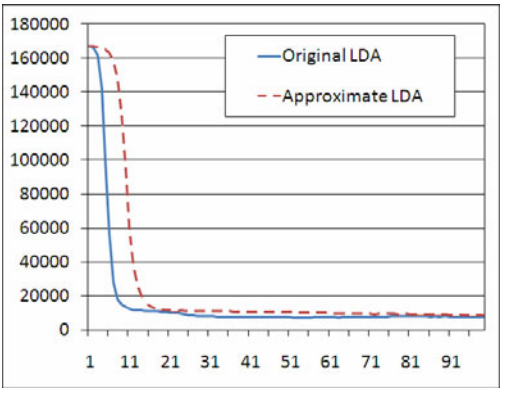

Fig. 1. Speeds of convergence of original LDA and approximate LDA. Horizontal axis denotes the times of sampling, while vertical axis denotes the number of data being changed each time.

In multi-computer environment, sampling step is achieved by m parties in parallel, and without updating parameters, so the 1-round cost of sampling step in distributed LDA is less than $1/m$ of original LDA.

Now we discuss the computing step. Since original LDA does not contain this step, the efficiency of this step is with strong relation to the whole efficiency of distributed LDA. By the limitation of number of computers, we just simulate such a system in a single computer by running all the necessary steps in sequence, recording the running time of each step, and then overlapping the time of parallel steps to compute the whole running time. Splitting all experiment data into m parties, we implement just one round of secure computation in computing step. To compare the cost in different cases, we simulate the cases that m is 5, 10 and 30, and T is 5, 10 and 50, where the data are different from experiment of text analysis above. The result is shown in Table 3. Here, we just fixed W=10 in each case. We notice from Table 3 that the cost of RSS protocol is proportional to m, because the total cost is $O(m)$ when T is fixed. In addition, the increment of the cost of RSS protocol is not so fast when T increases, since t_e and t_d is independent on T.

Table 3. Running time (sec.) of one round of RSS protocol

	m=5	m=10	m=30
T=5	*89*	*178*	*548*
T=10	*165*	*329*	*989*
T=50	*774*	*1547*	*4644*

7 Conclusions

In this paper, we have introduced a protocol of computation of ratio of secure summations. Our RSS protocol has higher security in the case of coalition. Furthermore, we also discussed the performance of the RSS protocols in this paper.

We have also introduced the Privacy-Preserving distributed LDA model which can be applied by the RSS protocol. Moreover, the RSS protocol can be utilized in large numbers of secure computation problems in privacy-preserving data mining, which require computing ratios of secure summations.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Griffiths, T.L., Steyvers, M.: Finding scientific topics. In: PNAS, pp. 5228–5235 (2004)
3. Jha, S., Kruger, L., McDamiel, P.: Privacy Preserving Clustering. In: 10th European Symposium On Research In Computer Security, Milan, Italy (2005)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
5. Goldreich, O.: Foundations of Cryptography, Basic Tools, vol. 1. Cambridge University Press, Cambridge (2001)
6. Goldreich, O.: Foundations of Cryptography, Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
7. Chor, B., Gereb-Graus, M., Kushilevitz, E.: On the Structure of the Privacy Hierarchy. Journal of Cryptology 7, 53–60 (1994)
8. Chor, B., Ishai, Y.: On Privacy and Partition Arguments. Information and Computation 167, 2–9 (2001)
9. Aggarwal, C.C., Yu, P.S.: Privacy-Preserving Data Mining: Models and Algorithms. Springer, Heidelberg (2008)
10. Paillier, P.: Public-Key Cryptosystems based on Composite Degree Residue Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
11. Damgard, I., Jurik, M.: A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
12. Benaloh, J.C.: Secret sharing homomorphisms: keeping shares of a secret secret. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 251–260. Springer, Heidelberg (1987)
13. Vaidya, J., Kantarcıoglu, M., Clifton, C.: Privacy-preserving Naïve Bayes classification. The VLDB Journal 17, 879–898 (2008)