Privacy-Preserving Computation of Bayesian Networks on Vertically Partitioned Data

Zhiqiang Yang and Rebecca N. Wright, *Member*, *IEEE*

Abstract—Traditionally, many data mining techniques have been designed in the centralized model in which all data is collected and available in one central site. However, as more and more activities are carried out using computers and computer networks, the amount of potentially sensitive data stored by business, governments, and other parties increases. Different parties often wish to benefit from cooperative use of their data, but privacy regulations and other privacy concerns may prevent the parties from sharing their data. Privacy-preserving data mining provides a solution by creating distributed data mining algorithms in which the underlying data need not be revealed. In this paper, we present privacy-preserving protocols for a particular data mining task: learning a Bayesian network from a database vertically partitioned among two parties. In this setting, two parties owning confidential databases wish to learn the Bayesian network on the combination of their databases without revealing anything else about their data to each other. We present an efficient and privacy-preserving protocol to construct a Bayesian network on the parties' joint data.

Index Terms—Data privacy, Bayesian networks, privacy-preserving data mining.

1 Introduction

The rapid growth of the Internet makes it easy to collect data on a large scale. Data is generally stored by a number of entities, ranging from individuals to small businesses to government agencies. This data includes sensitive data that, if used improperly, can harm data subjects, data owners, data users, or other relevant parties. Concern about the ownership, control, privacy, and accuracy of such data has become a top priority in technical, academic, business, and political circles. In some cases, regulations and consumer backlash also prohibit different organizations from sharing their data with each other. Such regulations include HIPAA [19] and the European privacy directives [35], [36].

As an example, consider a scenario in which a research center maintains a DNA database about a large set of people, while a hospital stores and maintains the history records of those people's medical diagnoses. The research center wants to explore correlations between DNA sequences and specific diseases. Due to privacy concerns and privacy regulations, the hospital cannot provide any information about individual medical records to the research center.

Data mining traditionally requires all data to be gathered into a central site where specific mining algorithms can be applied on the joint data. This model works in many data mining settings. However, clearly this is undesirable from a privacy perspective. Distributed data mining [28] removes the requirement of bringing all raw data to a central site, but this has usually been motivated by reasons of efficiency and solutions do not necessarily provide privacy. In contrast, privacy-preserving data mining solutions, including ours, provide data mining algorithms that compute or approximate the output of a particular algorithm applied to the joint data, while protecting other information about the data. Some privacy-preserving data mining solutions can also be used to create modified, publishable versions of the input data sets.

Bayesian networks are a powerful data mining tool. A Bayesian network consists of two parts: the network structure and the network parameters. Bayesian networks can be used for many tasks, such as hypothesis testing and automated scientific discovery. In this paper, we present privacy-preserving solutions for learning Bayesian networks on a database vertically partitioned between two parties. Using existing cryptographic primitives, we design several privacy-preserving protocols. We compose them to compute Bayesian networks in a privacy-preserving manner. Our solution computes an approximation of the existing $K2$ algorithm for learning the structure of the Bayesian network and computes the accurate parameters. In our solution, the two parties learn only the final Bayesian network plus the order in which network edges were added. Based on the security of the cryptographic primitives used, it is provable that no other information is revealed to the parties about each other's data. (More precisely, each party learns no information that is not implied by this output and his or her own input.)

We overview related work in Section 2. In Section 3, we give a brief review of Bayesian networks and the $K2$ algorithm. We present our security model and formalize the privacy-preserving Bayesian network learning problem on a vertically partitioned database in Section 4 and we introduce some cryptographic preliminaries in Section 5. In Sections 6 and 7, we describe our privacy-preserving structure-learning and parameter-learning solutions. In Section 8, we discuss how to efficiently combine the two learning steps together to reduce the total overhead.

• *The authors are with the Computer Science Department, Stevens Institute of Technology, Hoboken, NJ 07030.*
E-mail: {zyang, rwright}@cs.stevens.edu.

Manuscript received 21 July 2005; revised 17 Dec. 2005; accepted 13 Apr. 2006; published online 19 July 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0278-0705.

2 Related Work

Certain data mining computations can be enabled while providing privacy protection for the underlying data using privacy-preserving data mining, on which there is a large and growing body of work [33], [13], [29], [3]. Those solutions can largely be categorized into two approaches. One approach adopts cryptographic techniques to provide secure solutions in distributed settings (e.g., [29]). Another approach randomizes the original data in such a way that certain underlying patterns (such as distributions) are preserved in the randomized data (e.g., [3]).

Generally, the cryptographic approach can provide solutions with perfect accuracy and guarantee the computation itself leaks no information beyond the final results. The randomization approach is typically much more efficient than the cryptographic approach, but it suffers a trade-off between privacy and accuracy [1], [27]. Note that, in some cases, an accurate solution may be considered too privacy-invasive. Both the randomization approach and the cryptographic approach can purposely introduce additional error or randomization in this case.

Privacy-preserving algorithms have been proposed for different data mining applications, including decision trees on randomized data [3], association rules mining on randomized data [37], [14], association rules mining across multiple databases [40], [23], clustering [41], [21], [20], naive Bayes classification [24], [42], and privacy-preserving collaborative filtering [7]. Additionally, several solutions have been proposed for privacy-preserving versions of simple primitives that are very useful for designing privacy-preserving data mining algorithms. These include finding common elements [15], [2] computing scalar products [6], [4], [40], [39], [15], [16], and computing correlation matrices [30].

In principle, the elegant and powerful paradigm of secure multiparty computation provides cryptographic solutions for protecting privacy in any distributed computation [17], [46]. The definition of privacy is that no more information is leaked than in an "ideal" model in which each party sends her input to a trusted third party who carries out the computation on the received inputs and sends the appropriate results back to each party. Because, generally, there is no third party that all participating parties trust and because such a party would become a clear single target for attackers, secure multiparty computation provides privacy-preserving protocols that eliminate the need for a trusted third party while ensuring that each party learns nothing more than he or she would in the ideal model. However, the complexity of the general secure multiparty computation is rather high for computations on large data sets. More efficient privacy-preserving solutions can often be designed for specific distributed computations. Our work is an example of such a solution (in our case, for an ideal functionality that also computes both the desired Bayesian network and the order in which the edges were added, as we discuss further in Section 8.2). We use general two-party computation as a building block for some smaller parts of our computation to design a tailored, more efficient, solution to Bayesian network learning.

The field of distributed data mining provides distributed data mining algorithms for different applications [28], [38], [22] which, on minor modification, may provide privacy-preserving solutions. Distributed Bayesian network learning has been addressed for both vertically partitioned data and horizontally partitioned data [9], [8], [44]. These algorithms were designed without privacy in mind and, indeed, they require parties to share substantial amounts of information with each other. In Section 8.3, we briefly describe an alternate privacy-preserving Bayesian network structure-learning solution based on the solutions of Chen et al. [9], [8] and compare that solution to our main proposal.

Meng et al. [32] provide a privacy-preserving technique for learning the parameters of a Bayesian network in vertically partitioned data. We provide a detailed comparison of our technique for parameter learning to theirs in Section 7, where we show that our solution provides better accuracy, efficiency, and privacy.

3 Review of Bayesian Networks and the $K2$ Algorithm

In Section 3.1, we give an introduction to Bayesian networks. In Section 3.2, we briefly introduce the $K2$ algorithm for learning a Bayesian network from a set of data.

3.1 Bayesian Networks

A Bayesian network (BN) is a graphical model that encodes probabilistic relationships among variables of interest [11]. This model can be used for data analysis and is widely used in data mining applications. Formally, a Bayesian network for a set V of m variables is a pair $(B_{\mathrm{s}}, B_{\mathrm{p}})$. The *network structure* $B_{\mathrm{s}} = (V, E)$ is a directed acyclic graph whose nodes are the set of variables. The *parameters* B_{p} describe local probability distributions associated with each variable. The graph B_{s} represents conditional independence assertions about variables in V: An edge between two nodes denotes direct probabilistic relationships between the corresponding variables. Together, B_{s} and B_{p} define the joint probability distribution for V. Throughout this paper, we use v_i to denote both the variable and its corresponding node. We use π_i to denote the parents of node v_i in B_{s}. The absence of an edge between v_i and v_j denotes conditional independence between the two variables given the values of all other variables in the network.

For bookkeeping purposes, we assume there is a canonical ordering of variables and their possible instantiations which can be extended in the natural way to sets of variables. We denote the jth unique instantiation of V by V_j. Similarly, we denote the kth instantiation of a variable v_i by v_{i_k}. Given the set of parent variables π_i of a node v_i in the Bayesian network structure B_{s}, we denote the jth unique instantiation of π_i by ϕ_{ij}. We denote the number of unique instantiations of π_i by q_i and the number of unique instantiations of v_i by d_i.

Given a Bayesian network structure B_{s}, the joint probability for any particular instantiation V_ℓ of all the variables is given by:

$$\Pr[V = V_\ell] = \prod_{v_i \in V} \Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}],$$

where each k and j specify the instantiations of the corresponding variables as determined by V_ℓ. The network parameters

$$B_{\mathrm{p}} = \{\Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}] : v_i \in V, 1 \le j \le q_i, 1 \le k \le d_i\}$$

are the probabilities corresponding to the individual terms in this product. If variable v_i has no parents, then its parameters specify the marginal distribution of v_i: $\Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}] = \Pr[v_i = v_{i_k}]$.

3.2 $K2$ Algorithm

Determining the BN structure that best represents a set of data is NP-hard [10], so heuristic algorithms are typically used in practice. One of the most widely used structure-learning algorithms is the $K2$ algorithm [11], which we use as the starting point of our distributed privacy-preserving algorithm. The $K2$ algorithm is a greedy heuristic approach to efficiently determining a Bayesian network representation of probabilistic relationships between variables from a data set containing observations of those variables.

The $K2$ algorithm starts with a graph consisting of nodes representing the variables of interest, with no edges. For each node in turn, it then incrementally adds edges whose addition most increases the score of the graph, according to a specified score function. When the addition of no single parent can increase the score or a specified limit of parents has been reached, this algorithm stops adding parents to that node and moves onto the next node.

In the $K2$ algorithm, the number of parents for any node is restricted to some maximum u. Given a node v_i, $\mathrm{Pred}(v_i)$ denotes all the nodes less than v_i in the node ordering. D is a database of n records, where each record contains a value assignment for each variable in V. The $K2$ algorithm constructs a Bayesian network structure B_{s} whose nodes are the variables in V. Each node $v_i \in V$ has a set of parents π_i.

More generally, we define α_{ijk} to be the number of records in D in which variable v_i is instantiated as v_{i_k} and π_i is instantiated as ϕ_{ij}. Similarly, we define α_{ij} to be the number of records in D in which π_i is instantiated as ϕ_{ij}. We note that, therefore,

$$\alpha_{ij} = \sum_{k=1}^{d_i} \alpha_{ijk}. \tag{1}$$

In constructing the BN structure, the $K2$ algorithm uses the following score function $f(i, \pi_i)$ to determine which edges to add to the partially completed structure:

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(d_i - 1)!}{(\alpha_{ij} + d_i - 1)!} \prod_{k=1}^{d_i} \alpha_{ijk}! \tag{2}$$

We refer to all possible α_{ijk} and α_{ij} that appear in (2) as α-*parameters*. The $K2$ algorithm [11] is as follows:

Input: An ordered set of m nodes, an upper bound u on the number of parents for a node, and a database D containing n records.

Output: Bayesian network structure B_{s} (whose nodes are the m input nodes and whose edges are as defined by the values of π_i at the end of the computation)

For $i = 1$ **to** m
{
  $\pi_i = \emptyset$;
  $P_{\mathrm{old}} = f(i, \pi_i)$;
  KeepAdding = true;
  While KeepAdding and $|\pi_i| < u$
  {
    let z be the node in $\mathrm{Pred}(x_i) - \pi_i$ that maximizes $f(i, \pi_i \cup \{z\})$;
    $P_{\mathrm{new}} = f(i, \pi_i \cup \{z\})$;
    If $P_{\mathrm{new}} > P_{\mathrm{old}}$
      $P_{\mathrm{old}} = P_{\mathrm{new}}$;
      $\pi_i = \pi_i \cup \{z\}$;
    Else KeepAdding = false;
  }
}

4 Security Model and Problem Formalization

We formally state our security model in Section 4.1. We formalize the privacy-preserving distributed learning Bayesian network problem in Section 4.2. The security of our solution relies on the composition of privacy-preserving protocols, which is introduced in Section 4.3.

4.1 Security Model

Security in distributed computation is frequently defined with respect to an *ideal model* [18]. In the ideal model for privacy-preserving Bayesian networks, two parties send their databases to a *trusted third party* (TTP). The TTP then applies a Bayesian network learning algorithm on the combination of the two databases. Finally, the learned BN model is sent to the two parties by the trusted third party. In the ideal model, the two parties only learn the global BN (their objective) and nothing else. A distributed computation that does not make use of a TTP is then said to be secure if the parties learn nothing about each other's data during the execution of the protocol that they would not learn in the ideal model.

In this paper, we design a privacy-preserving solution for two parties to learn a BN using a secure distributed computation. Ideally, the parties should learn nothing more than in the ideal model. In our case, in order to obtain security with respect to an ideal model, we must also allow the ideal model to reveal the order in which an iterative algorithm adds edges to the Bayesian network (as this is revealed to Alice and Bob in our solution).

Following standard distributed cryptographic protocols, we make the distinction between *passive* and *active* adversaries [18]. Passive adversaries (often called semihonest adversaries) only gather information and do not modify the behavior of the parties. Such adversaries often model attacks that take place only after the execution of the protocol has completed. Active adversaries (often called malicious) cause the corrupted parties to execute arbitrary operations of the adversary's choosing, potentially learning more about the other party's data than intended. In this work, as in much of the existing privacy-preserving data mining literature, we suppose the parties in our setting are

semihonest adversaries. That is, they correctly follow their specified protocol, but they keep a record of all intermediate computation and passed messages and may use those to attempt to learn information about each other's inputs.

4.2 Problem Formalization

In the distributed two-party setting we consider, a database D consisting only of categorical variables is vertically partitioned among Alice and Bob. Alice and Bob hold confidential databases D_A and D_B, respectively, each of which can be regarded as a relational table. Each database has n rows. The variable sets in D_A and D_B are denoted by V_A and V_B, respectively. There is a common ID that links the rows in two databases owned by those two parties. Without loss of generality, we assume that the row index is the common ID that associates the two databases—that is, Alice's rows and Bob's rows represent the same records, in the same order, but Alice and Bob each have different variables in their respective "parts" of the records. Thus, $D = D_A \bowtie D_B$. Alice has D_A and Bob has D_B, where D_A has the variables $V_A = \{a_1, \ldots, a_{m_a}\}$ and D_B has the variables $V_B = \{b_1, \ldots, b_{m_b}\}$. (The sets D_A and D_B are assumed to be disjoint.) Hence, $m_a + m_b = m$ and the variable set is $V = V_A \cup V_B$. We assume the domains of databases D are public to both parties. We also assume the variables of interest are those in the set $V = V_A \cup V_B$. That is, Alice and Bob wish to compute the Bayesian network of the variables in their combined database $D_A \bowtie D_B$ without revealing any individual record and ideally not revealing any partial information about their own databases to each other except the information that can be derived from the final Bayesian network and their own database. However, our solution does reveal some partial information in that it reveals the order in which edges were added in the process of structure learning. The privacy of our solution is further discussed in Section 8.2.

4.3 Composition of Privacy-Preserving Protocols

In this section, we briefly discuss the composition of privacy-preserving protocols. In our solution, we use the composition of privacy-preserving subprotocols in which all intermediate outputs from one subprotocol that are inputs to the next subprotocol are computed as secret shares (see Section 5). In this way, it can be shown that if each subprotocol is privacy-preserving, then the resulting composition is also privacy-preserving [18], [5]. (A fully fleshed out proof of these results requires showing simulators that relate the information available to the parties in the actual computation to the information they could obtain in the ideal model.)

5 Cryptographic Preliminaries

In this section, we introduce several cryptographic preliminaries that are used to construct the privacy-preserving protocols for learning BN on vertically partitioned data.

5.1 Secure Two-Party Computation

Secure two-party computation, introduced by Yao [46] is a very general methodology for securely computing *any function*. Under the assumption of the existence of the collections of enhanced trapdoor permutations, Yao's solution provides a solution by which any polynomial-time computable (randomized) function can be securely computed in polynomial time. (In practice, a block cipher such as AES [12] is used as the enhanced trapdoor permutation, even though it is not proven to be one.) Essentially, the parties compute an encrypted version of a combinatorial circuit for the function and then they evaluate the circuit on encrypted values. A nice description of Yao's solution is presented in Appendix B of [29].

In our setting, as in any privacy-preserving data mining setting, general secure two-party computation would be too expensive to use for the entire computation if the data set is large. However, it is reasonable for functions that have small inputs and circuit representation, as was recently demonstrated in practice by the Fairplay system that implements it [31]. We use general secure two-party computation as a building block for several such functions.

5.2 Secret Sharing

In this work, we make use of secret sharing and, specifically, *2-out-of-2* secret sharing. A value x is "shared" between two parties in such a way that neither party knows x, but, given both parties' shares of x, it is easy to compute x. In our case, we use additive secret sharing in which Alice and Bob share a value x modulo some appropriate value N in such a way that Alice holds a, Bob holds b, and x is equal (not just congruent, but equal) to $(a + b) \bmod N$. An important property of this kind of secret sharing is that if Alice and Bob have shares of x and y, then they can each locally add their shares modulo N to obtain shares of $x + y$.

5.3 Privacy-Preserving Scalar Product Share Protocol

The scalar product of two vectors $\mathbf{z} = (z_1, \ldots, z_n)$ and $\mathbf{z}' = (z'_1, \ldots, z'_n)$ is $\mathbf{z} \cdot \mathbf{z}' = \sum_{i=1}^{n} z_i z'_i$. A privacy-preserving scalar product shares protocol where both parties hold each vector, respectively, and both parties learn secret shares of the product result. We only require the use of the scalar product protocol for binary data, even if the database consists of nonbinary data. This can be done with complete cryptographic privacy based on any additive homomorphic encryption scheme [6], [39], [16] such as the Paillier encryption scheme [34], which is secure assuming that it is computationally infeasible to determine composite residuosity classes. The protocol produces two shares whose sum modulo N (where N is appropriately related to the modulus used in the encryption scheme) is the target scalar product. To avoid the modulus introducing differences in computations, the modulus should be larger than the largest possible outcome of the scalar product.

5.4 $\ln x$ and $x \ln x$ Protocols

Lindell and Pinkas designed an efficient two-party privacy-preserving protocol for computing $x \ln x$ [29]. In the protocol, two parties have inputs v_1 and v_2, respectively, and we define $x = v_1 + v_2$. The output for this protocol is that two parties obtain random values w_1 and w_2, respectively, such that $w_1 + w_2 = x \ln x$. With the same techniques, the two parties can also compute secret shares for $\ln x$. Both protocols are themselves privacy-preserving and produce secret shares as their results.

6 Privacy-Preserving Bayesian Network Structure Protocol

In this section, we present a privacy-preserving protocol to learn the Bayesian network structure from a vertically partitioned database. We start in Sections 6.1 and 6.2 by describing a modified $K2$ score function and providing some experimental results for it. We describe several new privacy-preserving subprotocols in Sections 6.3, 6.4, 6.5, and 6.6. In Section 6.7, we combine these into our overall privacy-preserving solution for Bayesian network structure. Bayesian network parameters are discussed later in Section 7.

6.1 Our Score Function

We make a number of changes to the score function that appear not to substantially affect the outcome of the $K2$ algorithm and that result in a score function that works better for our privacy-preserving computation. Since the score function is only used for comparison purposes, we work instead with a different score function that has the same relative ordering. We then use an approximation to that score function. Specifically, we make three changes to the score function $f(i, \pi_i)$: We apply a natural logarithm, we take Stirling's approximation, and we drop some bounded terms.

First, we apply the natural logarithm to $f(i, \pi_i)$, yielding $f'(i, \pi_i) = \ln f(i, \pi_i)$ without affecting the ordering of different scores:

$$f'(i,\pi_i) = \sum_{j=1}^{q_i} \left(\ln(d_i - 1)! - \ln(\alpha_{ij} + d_i - 1)!\right) + \sum_{j=1}^{q_i}\sum_{k=1}^{d_i} \ln \alpha_{ijk}!. \tag{3}$$

Next, we wish to apply Stirling's approximation on $f'(i, \pi_i)$. Recall that Stirling's approximation says that, for any $\ell \geq 1$, we have $\ell! = \sqrt{2\pi\ell}\left(\frac{\ell}{e}\right)^{\ell} e^{\epsilon_\ell}$, where $\epsilon(\ell)$ is determined by Stirling's approximation and satisfies $\frac{1}{12\ell+1} < \epsilon(\ell) < \frac{1}{12\ell}$. However, if any α_{ijk} is equal to 0, then Stirling's approximation does not apply to $\alpha_{ijk}!$. As a solution, we note that, if an α_{ijk} is changed from 0 to 1 in (3), the outcome is unchanged because $1! = 0! = 1$. Hence, we replace any α_{ijk} that is 0 with 1. Specifically, we define $\beta_{ijk} = \alpha_{ijk}$ if α_{ijk} is not 0 and $\beta_{ijk} = 1$ if α_{ijk} is 0. Either way, we define $\beta_{ij} = \alpha_{ij}$. (This is simply so that we may entirely switch to using βs instead of having some βs and some αs.) We refer to β_{ijk} and β_{ij} for all possible i, j, and k as β parameters. Replacing α parameters with β parameters in (3), we have

$$f'(i,\pi_i) = \sum_{j=1}^{q_i} \left(\ln(d_i - 1)! - \ln(\beta_{ij} + d_i - 1)!\right) + \sum_{j=1}^{q_i}\sum_{k=1}^{d_i} \ln \beta_{ijk}! \tag{4}$$

Taking $\ell_{ij} = \beta_{ij} + d_i - 1$, we apply Stirling's approximation to (4), obtaining:

A	0.99	0.01						
S	0.5	0.5						
T	0.1	0.9	0.9	0.1				
L	0.3	0.6	0.7	0.4				
B	0.1	0.8	0.9	0.2				
E	0.9	0.1	0.1	0.01	0.1	0.9	0.9	0.9
X	0.2	0.6	0.8	0.4				
D	0.9	0.1	0.1	0.01	0.1	0.9	0.9	0.9

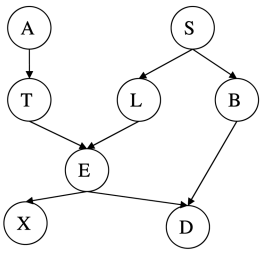


Fig. 1. The Bayesian network parameters and structure for the Asia model.

$$f'(i,\pi_i) \approx \sum_{j=1}^{q_i}\left(\sum_{k=1}^{d_i}\left(\frac{1}{2}\ln\beta_{ijk} + \beta_{ijk}\ln\beta_{ijk} - \beta_{ijk} + \epsilon(\beta_{ijk})\right) - \left(\frac{1}{2}\ln\ell_{ij} + \ell_{ij}\ln\ell_{ij} - \ell_{ij} + \epsilon(\ell_{ij})\right)\right) + q_i\ln(d_i-1)! + \frac{q_i(d_i-1)}{2}\ln 2\pi. \tag{5}$$

Finally, dropping the bounded terms $\epsilon_{\ell_{ij}}$ and $\epsilon_{\beta_{ijk}}$, pulling out $q_i(d_i - 1)$, and setting

$$\mathrm{pub}(d_i, q_i) = q_i(d_i - 1) + q_i \ln(d_i - 1)! + \frac{q_i(d_i-1)}{2}\ln 2\pi,$$

we obtain our score function $g(i, \pi_i)$ that approximates the same relative ordering as $f(i, \pi_i)$:

$$g(i,\pi_i) = \sum_{j=1}^{q_i}\left(\sum_{k=1}^{d_i}\left(\frac{1}{2}\ln\beta_{ijk} + \beta_{ijk}\ln\beta_{ijk}\right) - \left(\frac{1}{2}\ln\ell_{ij} + \ell_{ij}\ln\ell_{ij}\right)\right) + \mathrm{pub}(d_i, q_i). \tag{6}$$

A main component of our privacy-preserving $K2$ solution is showing how to compute $g(i, \pi_i)$ in a privacy-preserving manner, as described in the remainder of this section. First, we provide some experimental results to provide some evidence that f and g produce similar results.

6.2 Experimental Results of Our Score Function

We tested our score function on two different data sets in order to validate that it produces an acceptable approximation to the standard $K2$ algorithm. The first data set, called the Asia data set, includes one million instances. It is generated from the commonly used Asia model.[1] The Bayesian network for the Asia model is shown in Fig. 1. This model has eight variables: Asia, Smoking, Tuberculosis, Lung cancer, Bronchitis, Either, X-ray, and Dyspnoea, denoted by {A, S, T, L, B, E, X, D}.

The second data set is a synthetic data set with 10,000 instances, including six variables denoted 0 to 5. All six variables are binary, either true or false. Variables 0, 1, and 3 were chosen uniformly at random. Variable 2 is the XOR of variables 0 and 1. Variable 4 is the product of variables 1 and 3. Variable 5 is the XOR of variables 2 and 4.

On those two data sets, we tested the $K2$ algorithms with both score functions f and g. For both the Asia data set and

1. http://www.cs.huji.ac.il/labs/compbio/LibB/programs.html#GenInstance.

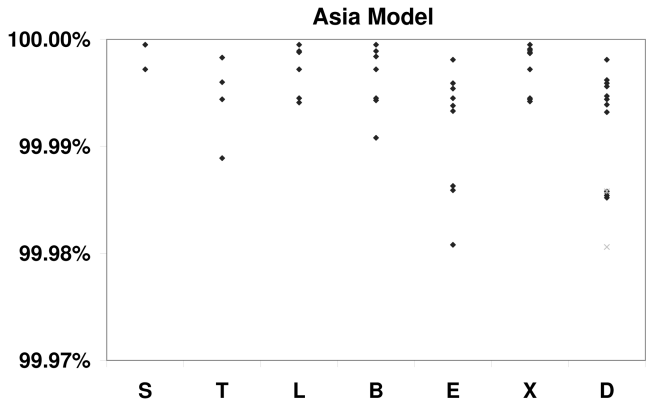


Fig. 2. The ratio of g to $\ln f$ in the Asia model.

the synthetic data set, the $K2$ algorithm generates the same structures whether f or g is used as the score function.

We further compare the difference of g and $\ln f$ for both data sets as computed by the $K2$ algorithm. (Recall that g is our approximation to $\ln f$.) In total, the $K2$ algorithm computes 64 scores on the Asia data set and 30 scores on the synthetic data set. Fig. 2 shows the ratios of each g to the corresponding $\ln f$. The X-axis represents different variables and the Y-axis represents the ratios of g to $\ln f$ that are computed for choosing the parents at each node. For instance, 14 scores for node D are computed to choose the parents of D. In the Asia model, all g scores are within 99.8 percent of $\ln f$. The experimental results illustrate that, for those two data sets, the g score function is a good enough approximation to the f score function for the purposes of the $K2$ algorithm. Kardes et al. [26] have implemented our complete privacy-preserving Bayesian network structure protocol and are currently carrying out additional experiments.

6.3 Privacy-Preserving Computation of α Parameters

In this section, we describe how to compute secret shares of the α parameters defined in Section 3.2 in a privacy-preserving manner. Recall that α_{ijk} is the number of records in $D = D_A \bowtie D_B$, where v_i is instantiated as v_{i_k} and π_i is instantiated as ϕ_{ij} (as defined in Section 3.2), and recall that q_i is the number of unique instantiations that the variables in π_i can take on. The α parameters include all possible α_{ijk} and α_{ij} that appear in (2) in Section 3.2.

Given instantiations v_{i_k} of variable v_i and ϕ_{ij} of the parents π_i of v_i, we say a record in D is *compatible with ϕ_{ij} for Alice* if the variables in $\pi_i \cap V_A$ (i.e., the variables in π_i that are owned by Alice) are assigned as specified by the instantiation ϕ_{ij} and we say the record is *compatible with v_{i_k} and ϕ_{ij} for Alice* if the variables in $(\{v_i\} \cup \pi_i) \cap V_A$ are assigned as specified by the instantiations v_{i_k} and ϕ_{ij}. Similarly, we say a record is *compatible for Bob* with ϕ_{ij}, or with v_{i_k} and ϕ_{ij}, if the relevant variables in V_B are assigned according to the specified instantiation(s).

We note that α_{ijk} can be computed by determining how many records are compatible for both Alice and Bob with v_i and π_i. Similarly, α_{ij} can be computed by determining how many records are compatible for both Alice and Bob with π_i. Thus, Alice and Bob can determine α_{ijk} and α_{ij} using privacy-preserving scalar product share protocols (see Section 5) such that Alice and Bob learn secret shares of α_{ijk} and α_{ij}. We describe this process in more detail below.

We define the vector $\mathrm{compat}_A(\phi_{ij})$ to be the vector $(x_1, \ldots, x_n)$ in which $x_\ell = 1$ if the ℓth database record is compatible for Alice with ϕ_{ij}; otherwise, $x_\ell = 0$. We analogously define $\mathrm{compat}_A(v_{i_k}, \phi_{ij})$, $\mathrm{compat}_B(\phi_{ij})$, and $\mathrm{compat}_B(v_{i_k}, \phi_{ij})$. Note that, given the network structure and i, j, k, Alice can construct $\mathrm{compat}_A(\phi_{ij})$ and $\mathrm{compat}_A(v_{i_k}, \phi_{ij})$ and Bob can construct $\mathrm{compat}_B(\phi_{ij})$ and $\mathrm{compat}_B(v_{i_k}, \phi_{ij})$. Then, $\alpha_{ij} = \mathrm{compat}_A(\phi_{ij}) \cdot \mathrm{compat}_B(\phi_{ij})$ and $\alpha_{ijk} = \mathrm{compat}_A(v_{i_k}, \phi_{ij}) \cdot \mathrm{compat}_B(v_{i_k}, \phi_{ij})$. However, the parties cannot, in general, learn α_{ijk} and α_{ij} as this would violate privacy.

Note that in the degenerate case, all variables for v_i and π_i belong to one party, who can locally compute the corresponding α parameters without any interaction with the other party. The following protocol computes α_{ijk} parameters for the general case in which variables including v_i and π_i are distributed among two parties:

Input: D_A and D_B held by Alice and Bob, respectively, values $1 \le i \le m$, $1 \le j \le q_i$, and $1 \le k \le d_i$, plus the current value of π_i and a particular instantiation ϕ_{ij} of the variables in π_i are commonly known to both parties.

Output: Two secret shares of α_{ijk}.

1) Alice and Bob generate $\mathrm{compat}_A(v_{i_k}, \phi_{ij})$ and $\mathrm{compat}_B(v_{i_k}, \phi_{ij})$, respectively.
2) By taking $\mathrm{compat}_A(v_{i_k}, \phi_{ij})$ and $\mathrm{compat}_B(v_{i_k}, \phi_{ij})$ as two inputs, Alice and Bob execute the privacy-preserving scalar product share protocol of Section 5 to generate the secret shares of α_{ijk}.

By running the above protocol for all possible combinations i, j, and k, Alice and Bob can compute secret shares for all α_{ijk} parameters in (2). Since $\alpha_{ij} = \sum_{k=1}^{d_i} \alpha_{ijk}$, Alice and Bob can compute the secret shares for a particular α_{ij} by simply adding all their secret shares of α_{ijk} together.

Theorem 1. *Assuming both parties are semihonest, the protocol for computing α parameters is privacy-preserving.*

Proof. Since the scalar product share protocol is privacy-preserving, the privacy of each party is protected. Each party only learns secret shares of each α-parameter and nothing else about individual records of the other party's data. □

6.4 Privacy-Preserving Computation of β Parameters

We now show how to compute secret shares of the β parameters of (6). As described earlier in Section 6.3, Alice and Bob can compute secret shares for α_{ijk} and α_{ij}. We denote these shares by $\alpha_{ijk} = a_{ijk} + b_{ijk}$ and $\alpha_{ij} = a_{ij} + b_{ij}$, where a_{ijk}, a_{ij} and b_{ijk}, b_{ij} are secret shares held by Alice and Bob, respectively. Since β_{ij} is equal to α_{ij} (by definition), the secret shares of β_{ij} are a_{ij} and b_{ij}.

Recall that $\beta_{ijk} = \alpha_{ijk}$ if α_{ijk} is not 0; otherwise, $\beta_{ijk} = 1$. However, neither Alice nor Bob knows the value of each α_{ijk} because each only has a secret share of each α_{ijk}. Hence, neither of them can directly compute the secret shares of

β_{ijk} from α_{ijk}. (The direct exchange of their secret shares would incur a privacy breach.)

We use general secure two-party computation to generate the secret shares of β_{ijk}. That is, Alice and Bob carry out a secure version of the following algorithm. Given that the algorithm is very simple and has small inputs, Yao's secure two-party computation of it can be carried out privately and efficiently [46], [31].

Input: a_{ijk} and b_{ijk} held by Alice and Bob.
Output: Rerandomized a_{ijk} and b_{ijk} to Alice and Bob, respectively.

If $(a_{ijk} + b_{ijk} == 0)$
Rerandomize a_{ijk} and b_{ijk} s.t. $a_{ijk} + b_{ijk} = 1$;
Else Rerandomize a_{ijk} and b_{ijk} s.t. $a_{ijk} + b_{ijk} = \alpha_{ijk}$;

That is, Alice and Bob's inputs to the computation are two secret shares of α_{ijk}. They obtain two new secret shares of β_{ijk}.

6.5 Privacy-Preserving Score Computation

Our goal in this subprotocol is to privately compute two secret shares of the output of the $g(i, \pi_i)$ score function. There are five kinds of subformulas to compute in the $g(i, \pi_i)$ score function:

1. $\ln \beta_{ijk}$,
2. $\beta_{ijk} \ln \beta_{ijk}$,
3. $\ln (\beta_{ij} + d_i - 1)$,
4. $(\beta_{ij} + d_i - 1) \ln(\beta_{ij} + d_i - 1)$, and
5. $\text{pub}(d_i, q_i)$.

To compute two secret shares of $g(i, \pi_i)$ for Alice and Bob, the basic idea is to compute two secret shares of each subformula for Alice and Bob and then Alice and Bob can add their secret shares of the subformulas together to get the secret shares of $g(i, \pi_i)$. The details of how to compute the secret shares are addressed below.

Since d_i is public to each party, secret shares of $\beta_{ij} + d_i - 1$ can be computed by Alice (or Bob) adding $d_i - 1$ to her secret share of β_{ij} such that Alice holds $a_{ij} + d_i - 1$ and Bob holds b_{ij} as the secret shares for $\beta_{ij} + d_i - 1$. Hence, items 1 and 3 in the above list can be written as $\ln a_{ijk} + b_{ijk}$ and $\ln (a_{ij} + d_i - 1) + b_{ij}$. Then, the problem of computing secret shares for items 1 and 3 above can be reduced to the problem of computing two secret shares for $\ln x$, where x is secretly shared by two parties. The $\ln x$ problem can be solved by the privacy-preserving $\ln x$ protocol of Lindell and Pinkas [29].

Similarly, the problem of generating two secret shares for items 2 and 4 above can be reduced to the problem of computing secret shares of $x \ln x$ in a privacy-preserving manner, which again is solved by Lindell and Pinkas [29]. In item 5 above, q_i and d_i are known to both parties, so they can be computed by either party.

After computing secret shares for items 1, 2, 3, 4, and 5 above, Alice and Bob can locally add their respective secret shares to compute secret shares of $g(i, \pi_i)$. Because each subprotocol is privacy-preserving and results in only secret shares as intermediate results, the computation of secret shares of $g(i, \pi_i)$ is privacy-preserving.

6.6 Privacy-Preserving Score Comparison

In the $K2$ algorithm specified in Section 3, Alice and Bob need to determine which of a number of shared values is maximum. That is, we require the following privacy-preserving comparison computation:

Input: $(r_{a_1}, r_{a_2}, \ldots, r_{a_x})$ held by Alice and $(r_{b_1}, r_{b_2}, \ldots, r_{b_x})$ held by Bob.
Output: i such that $r_{a_i} + r_{b_i} \geq r_{a_j} + r_{b_j}$ for $1 \leq j \leq x$.

In this case, x is at most $u + 1$, where u is the restriction on the number of possible parents for any node and, in any case, no larger than m, the total number of variables in the combined database. Given that generally m will be much smaller than n, this can be privately and efficiently computed using general secure two-party computation [46], [31].

6.7 Overall Privacy-Preserving Solution for Learning Bayesian Network Structure

Our distributed privacy-preserving structure-learning protocol is shown in Fig. 3. It is based on the $K2$ algorithm, using the variable set of the combined database $D_A \bowtie D_B$, but executes without revealing the individual data values and the sensitive information of each party to the other. Each party learns only the BN structure plus the order in which edges were added (which in turn reveals which edge had maximum score at each iteration).

In the original $K2$ algorithm, all the variables are in one central site, while, in our setting, the variables are distributed in two sites. Hence, we must compute the score function across two sites. Remembering that $\ell_{ij} = \beta_{ij} + d_i - 1$, we can see from (6) that the score relies on the β parameters.

Other than the distributed computation of the scores and their comparison, our control flow is as given in the $K2$ algorithm. (For efficiency reasons, it is preferable to combine the comparisons that determine which possible parent yields the highest score with the comparison to determine if this score is higher than the current score, but logically the two are equivalent.) Note that this method leaks relative score values by revealing the order in which the edges were added. Formally, in order for the protocol to be considered privacy-preserving, we therefore consider it to be a protocol for computing Bayesian network structure *and the order in which edges were added by the algorithm*.

The protocol does not reveal the actual scores or any other intermediate values. Instead, we use privacy-preserving protocols to compute the secret shares of the scores. We divide the BN structure-learning problem into smaller subproblems and use the earlier described privacy-preserving subprotocols to compute shares of the β parameters (Section 6.4) and the scores (Section 6.5) in a privacy-preserving way, and to compare the resulting scores in a privacy-preserving way (Section 6.6). Overall, the privacy-preserving protocol is executed jointly between Alice and Bob as shown in Fig. 3. It has been fully implemented by Kardes et al. [26]. Privacy and performance issues are further discussed in Section 8.

Theorem 2. *Assuming the subprotocols are privacy-preserving, the protocol to compute Bayesian network structure reveals*

Input: An ordered set of m nodes, an upper bound u on the number of parents for a node, both known to Alice and Bob, and a database D containing n records, vertically partitioned between Alice and Bob.

Output: Bayesian network structure B_s (whose nodes are the m input nodes, and whose edges are as defined by the values of π_i at the end of the protocol)

Using a fixed ordering of variables in V, Alice and Bob execute the following steps for each node v_i. Initially, each node has no parent. After Alice and Bob run the following steps for node v_i, its parents are the nodes in π_i.

Step 1: Alice and Bob execute the privacy-preserving score protocol to compute the secret shares of $g(i, \pi_i)$ and $g(i, \pi_i \cup \{z\})$ for any $z \in \mathsf{Pred}(v_i) - \pi_i$.

Step 2: Alice and Bob execute the privacy-preserving score comparison protocol to compute which of the scores in Step 1 is maximum.

Step 3: If $g(i, \pi_i)$ is maximum, Alice and Bob go to the next node v_{i+1} to run from Step 1 until Step 3. If one z generates the maximum score in Step 2, then z is added as the parent of v_i s.t. $\pi_i = \pi_i \cup \{z\}$ and Alice and Bob go back to Step 1 with the same node v_i.

Fig. 3. Structure-learning protocol.

nothing except the Bayesian network structure and order in which the nodes are added.

Proof. Besides the structure itself, the structure-learning protocol reveals only the order information because each of the subprotocols is privacy-preserving, they are invoked sequentially, and they only output secret shares at each step. □

7 Privacy-Preserving Bayesian Network Parameters Protocol

In this section, we present a privacy-preserving solution for computing Bayesian network parameters on a database vertically partitioned between two parties. Assuming the BN structure is already known, Meng et al. presented a privacy-preserving method for learning the BN parameters [32], which we refer to as MSK. In this section, we describe an alternate solution to MSK. In contrast to MSK, ours is more private, more efficient, and more accurate. In particular, our parameter-learning solution provides complete privacy, in that the only information the parties learn about each other's inputs is the desired output, and complete accuracy, in that the parameters computed are exactly what they would be if the data were centralized. In addition, our solution works for both binary and nonbinary discrete data. We provide a more detailed comparison between the two solutions in Section 7.2.

As we discuss further in Section 8.1, it is possible to run our structure-learning protocol and parameter-learning protocol together for only a small additional cost over just the structure-learning protocol.

7.1 Privacy-Preserving Protocol for Learning BN Parameters

Recall the description of Bayesian network parameters in Section 3.1. Given Bayesian network structure B_s, the network parameters are the conditional probabilities $B_p = \{\Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}] : v_i \in V, 1 \le j \le q_i, 1 \le k \le d_i\}$. If variable v_i has no parents, then its parameters specify the marginal distribution of v_i:

$$\Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}] = \Pr[v_i = v_{i_k}].$$

Note that these parameters can be computed from the α parameters as follows:

$$\Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}] = \frac{\alpha_{ijk}}{\alpha_{ij}}. \quad (7)$$

Earlier, in Section 6.3, we described a privacy-preserving protocol to compute secret shares of α_{ijk} and α_{ij}. Now, we need to extend this to allow the parties to compute the value α_{ijk}/α_{ij} without sharing their data or revealing any intermediate values such as α_{ijk} and α_{ij} (unless such values can be computed from the BN parameters themselves, in which case, revealing them does not constitute a privacy breach). We consider three cases separately:

1. **One party owns all relevant variables.** In the degenerate case, one party (say, Alice) owns all of the relevant variables: $\{v_i\} \cup \pi_i$. In this case, she can compute α_{ijk}/α_{ij} locally and announce the result to Bob.
2. **One party owns all parents, other party owns node.** In the next simplest case, one party (again, say Alice) owns all the variables in π_i and the other party (Bob) owns v_i. In this case, Alice can again directly compute α_{ij} from her own data. Alice and Bob can compute the secret shares of α_{ijk} using the protocol described in Section 6.3. Bob then sends his share of α_{ijk} to Alice so she can compute α_{ijk}. (In this case, it is not a privacy violation for her to learn α_{ijk} because, knowing α_{ij}, she could compute α_{ijk} from the final public parameter α_{ijk}/α_{ij}.) From α_{ijk} and α_{ij}, Alice then computes α_{ijk}/α_{ij}, which she also announces to Bob.
3. **The general case: The parent nodes are divided between Alice and Bob.** In the general case, Alice and Bob have secret shares for both α_{ijk} and α_{ij} such that $a_{ijk} + b_{ijk} = \alpha_{ijk}$ and $a_{ij} + b_{ij} = \alpha_{ij}$ (where these additions are modular additions in a group depending on the underlying scalar product share protocol used in Section 6.3). Thus, the desired parameter is $(a_{ijk} + b_{ijk})/(a_{ij} + b_{ij})$. In order to carry out this computation without revealing anything about a_{ijk}

Input: A database D is vertically partitioned between Alice and Bob. Alice holds D_A and Bob holds D_B. A global Bayesian network B_{s} over $D_A \bowtie D_B$ is known to both parties.

Output: The parameters B_{p} of the Bayesian network B_{s}.

For each variable v_i in B_{s} and for each possible instantiation of v_i and π_i, Alice and Bob carry out the computation in the appropriate case discussed above. In the general case, this involves:

a. Alice and Bob run the privacy-preserving protocol for computing α-parameters introduced in Section 6.3 to learn secret shares of α_{ijk} and α_{ij}.

b. Alice and Bob run a secure two-party computation to compute the desired parameter α_{ijk}/α_{ij}.

Fig. 4. Parameter-learning protocol.

and a_{ij} to Bob or b_{ijk} and b_{ij} to Alice, we make use of general secure two-party computation. Note that this is sufficiently efficient here because the inputs are values of size k, independent of the database size n, and because the function to compute is quite simple.

Note that cases 1 and 2 could also be handled by the general case, but the simpler solutions provide a practical optimization as they require less computation and communication. In order to learn all the parameters B_{p}, Alice and Bob compute each parameter for each variable using the method just described above, as demonstrated in Fig. 4.

Theorem 3. *Assuming the privacy and correctness of the protocol for computing α parameters is privacy-preserving and the secure two-party computation protocol, the parameter-learning protocol is correct and private.*

Proof. The correctness of that protocol is clear because the values computed are precisely the desired parameters α_{ijk}/α_{ij}.

Privacy is protected because, in each case, we only reveal values to a party that are either part of the final output or are straightforwardly computable from the final output and its own input. All other intermediate values are protected via secret sharing, which reveals no additional information to the parties. □

7.2 Comparison with MSK

For a data set containing only binary values, the MSK solution showed that the count information required to estimate the BN parameters can be obtained as a solution to a set of linear equations involving some inner products between the relevant different feature vectors. In MSK, a random projection-based method is used to securely compute the inner product.

In this section, we provide a detailed comparison of the privacy, efficiency, and accuracy of our parameter-learning solution and MSK. We show that our solution performs better in efficiency, accuracy, and privacy than MSK. The primary difference between our solution and MSK is that MSK computes the parameter probabilities by first computing the counts of the various possible instantiations of nodes and their parents. As we discuss below, this approach inherently leaks more information than the parameters alone. In addition, they use a secure "pseudo inner product" to compute those counts, using a method that is less efficient, less accurate, and less private than cryptographic scalar product protocols (such as those discussed in Section 5).

As we discuss further below, replacing the pseudo inner product of MSK with an appropriate cryptographic scalar product would improve MSK to have somewhat better efficiency than our solution and complete accuracy (as our solution does). Our solution remains more private than the modified MSK, so, in some sense, this suggests that our solution and the modified MSK solution represent an efficiency/privacy tradeoff.

7.2.1 Efficiency

Let $d = \max d_i$ be the maximum number of possible values any variable takes on, κ be a security parameter describing the length of cryptographic keys used in the scalar product protocol, and u be the maximum number of parents any node in the Bayesian network has. (Thus, $u \leq m - 1$ and, typically, $u \ll m \ll n$). Our solution runs in time $O(md^{(u+1)}(\kappa n + \kappa^2))$. Taking $d = 2$ for purposes of comparison (since MSK assumes the data is binary-valued), this is $O(m2^{(u+1)}(\kappa n + \kappa^2))$. In contrast, MSK runs in time $O(m(2^{(u+1)} + n^2))$. In particular, for a fixed security parameter κ and maximum number u of parents of any node, as the database grows large enough that $\kappa \ll n$, our efficiency grows linearly in n, while MSK grows as n^2.

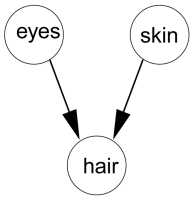
We note that the source of the quadratic growth of MSK is their secure pseudo inner product as, for an input database with n records, it requires the parties to produce and compute with an $n \times n$ matrix. If this step were replaced with an ideally private cryptographic scalar product protocol such as the one we use, their performance would improve to $O(m(2^{(u+1)} + \kappa n))$, a moderate efficiency improvement over our solution.

7.2.2 Accuracy

Our parameter-learning solution provides complete accuracy in the sense that we faithfully produce the desired parameters. The secure pseudo inner product computation of MSK introduces a small amount of computational error. Again, replacing this step with a perfectly accurate cryptographic scalar product can provide perfect accuracy.

7.2.3 Privacy

Our parameter-learning solution provides ideal privacy in the sense that the parties learn nothing about each other's



eyes: $\Pr[\text{eyes} = \text{brown}] = \Pr[\text{eyes} = \text{blue}] = 1/2$

skin: $\Pr[\text{skin} = \text{fair}] = \Pr[\text{eyes} = \text{dark}] = 1/2$

hair:

	eyes, skin			
hair	brown, fair	brown, dark	blue, fair	blue, dark
blond	1/2	1/2	1/2	1/2
brown	1/2	1/2	1/2	1/2

Fig. 5. One example of BN structure and parameters.

inputs beyond what is implied by the Bayesian parameters and their own inputs. MSK has two privacy leaks beyond ideal privacy. The first comes from the secure pseudo inner product computation, but again this could be avoided by using an ideally private scalar product protocol instead. The second, however, is intrinsic to their approach. As mentioned earlier, they compute the parameter probabilities by first computing the counts of the various possible instantiations of nodes and their parents. As they point out, the probabilities can be easily computed from the counts, so this does not affect the correctness of their computation. However, *the reverse is not true*—in general, the counts leak more information than the probabilities because different counts can give rise to the same probabilities. We illustrate this by a simple example, as shown in Figs. 5 and 6.

In this example, Alice owns the variable eyes, while Bob owns skin and hair. The Bayesian network, consisting of both the given structure (which we assume is given as part of the input to the problem) and the parameters (which are computed from the input databases), are shown in Fig. 5.

Fig. 6 shows two quite different kinds of databases, DB1 and DB2, that are both consistent with the computed Bayesian network parameters and with a particular setting for Alice's values for eyes. Both databases have 16 records. For eyes, half the entries are brown and half are blue; similarly, for skin, half the entries are fair and half are dark. The difference between DB1 and DB2 lies with hair and its relation to the other variables. One can easily verify that both DB1 and DB2 are consistent with the computed Bayesian network parameters and with a particular setting for Alice's values for eyes. Hence, given only the parameters and her own input, Alice would consider both databases with counts as shown in DB1 and with counts as shown in DB2 possible (as well as possibly other databases). However, if Alice is given additional count information, she can determine that either DB1 or DB2 is not possible, substantially reducing her uncertainty about Bob's data values. Although this example is simple and rather artificial, it suffices to demonstrate the general problem.

8 Discussion

We analyze the performance and privacy issues of the proposed solution in Sections 8.1 and 8.2. In Section 8.3, we discuss a possible alternate solution.

8.1 Performance Analysis

We have presented privacy-preserving protocols for learning BN structure and parameters (Sections 6 and 7, respectively). Rather than running these sequentially to learn a Bayesian network from a data set, these two protocols can be combined so that the BN parameters can be computed with a constant overhead over the computation of the BN structure. This is because the secret shares of α_{ijk} and α_{ij} needed in the parameter protocol are already computed in the structure protocol. Hence, the only additional overhead to compute the parameters is the secure two-party computation to divide the shared α_{ijk} by the shared α_{ij}.

Further, we note a few more potential practical optimizations. For example, in order to reduce the number of rounds of communication, the α parameters can be computed in parallel, rather than in sequence. This allows all the vectors for a given set of variables to be computed in a single pass through the database, rather than multiple passes. Similarly, shares of each α_{ij} need only be computed once, rather than once for each BN parameter. Additionally, if multiple nodes share the same set of parents, the same intermediate values can be reused multiple times.

As discussed above, the dominating overhead of our solution comes from computing the BN structure. Hence, the overall overhead of our solution depends on the database size n, the number m of variables, and the limit u on the number of possible parents for any node. Like the original $K2$ algorithm, our Structure Protocol requires computation that is exponential in u (in order to compute the α parameters for all possible $O(2^u)$ instantiations of the set of parents of a given node). In the $K2$ algorithm, the inner loop runs $O(mu)$ times. Each time the inner loop is executed, there are $O(u)$ scores to compute, each requiring $O(m2^u)$ α parameters to be computed. In our solution, the computation of each

instantiation			counts	counts
eyes	skin	hair	in DB1	in DB2
brown	fair	blond	2	1
brown	fair	brown	2	1
brown	dark	blond	2	3
brown	dark	brown	2	3
blue	fair	blond	2	3
blue	fair	brown	2	3
blue	dark	blond	2	1
blue	dark	brown	2	1

Fig. 6. Example showing that counts leak more information than parameters.

α-parameter, including the scalar product share protocol, requires $O(n)$ communication and computation. This is the only place that n comes into the complexity. Everything else, including computing β parameters from α parameters, combining β parameters into the score, and the score comparison, can be done in computation and communication that is polynomial in m and 2^u.

8.2 Privacy Analysis

In our solution, each party learns the Bayesian network, including the structure and the parameters, on the joint data without exchanging their raw data with each other. In addition to the Bayesian network, each party also learns the relative order in which edges are added into the BN structure. While this could be a privacy breach for some settings, it seems a reasonable privacy/efficiency trade-off that may be acceptable in many settings.

We note that the BN parameters contain much statistical information about each database, so another concern is that, even if a privacy-preserving computation of Bayesian network parameters is used, the resulting BN model—particularly when taken together with one party's database—reveals quite a lot of information about the other party's database. That is, the *result* of the privacy-preserving computation may itself leak too much information, even if the computation is performed in a completely privacy-preserving way, a phenomenon discussed nicely by Kantarcioglu et al. [25]. To limit this leakage, it might be preferable, for example, to have the parameters associated with a variable v_i revealed only to the party owning the variable. By using a secure two-party computation that gives the result only to the appropriate party, our solution can easily be modified to do this.

Another option would be to have the parties learn secret shares of the resulting parameters, not the actual parameters. This suggests an open research direction, which is to design mechanisms that allow the parties to use the Bayesian network and shared parameters in a privacy-preserving interactive way to carry out classification or whatever task they seek to perform.

8.3 Possible Alternate Solution

Chen et al. present efficient solutions for learning Bayesian networks on vertically partitioned data [8], [9]. In their solutions, each party first learns local BN models based on his own data, then sends a subset of his data to the other party. A global BN model is learned on the combination of the communicated subsets. (The computation can be done by either party.) Finally, the final BN model is learned by combining the global BN model and each party's local BN model. Those solutions are very efficient both in computation and communication, but, obviously, they were not designed with privacy in mind as each party has to send part of his data to the other party. Further, these solutions suffer a trade-off between the quality of the final BN model and the amount of communicated data: The more of their own data the parties send to each other, the more accurate the final BN model will be.

By combining our proposed solution with the solutions in [8], [9], we can achieve a new solution that provides privacy, as discussed in Section 8.2, together with a trade-off between performance and accuracy. The basic idea is that, first, each party locally learns a model on his or her own data and chooses the appropriate subset of his or her data according to the methods of [8], [9]. Rather than sending the selected subset of data to the other party, both parties then run our solutions described in Sections 6 and 7 on the chosen subset of their data to privately learn the global BN model on their data subsets. Finally, each party publishes his or her local BN models and the parties combine the global BN model with their local models to learn the final BN model following the methods of [8], [9]. This solution suffers a similar trade-off between performance and accuracy as the solutions of [8], [9], but with improved privacy as parties no longer send their individual data items to each other.

Acknowledgments

The authors thank Raphael Ryger for pointing out the need for introducing the β parameters. They also thank Onur Kardes for helpful discussions. Preliminary versions of parts of this work appeared in [43] and [45]. This work was supported by the US National Science Foundation under grant number CNS-0331584.

References

[1] D. Agrawal and C. Aggarwal, "On the Design and Quantification of Privacy Preserving Data Mining Algorithms," *Proc. 20th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems,* pp. 247-255, 2001.

[2] R. Agrawal, A. Evfimievski, and R. Srikant, "Information Sharing across Private Databases," *Proc. 2003 ACM SIGMOD Int'l Conf. Management of Data,* pp. 86-97, 2003.

[3] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," *Proc. 2000 ACM SIGMOD Int'l Conf. Management of Data,* pp. 439-450, May 2000.

[4] M. Atallah and W. Du, "Secure Multi-Party Computational Geometry," *Proc. Seventh Int'l Workshop Algorithms and Data Structures,* pp. 165-179, 2001.

[5] R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols," *J. Cryptology,* vol. 13, no. 1, pp. 143-202, 2000.

[6] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R.N. Wright, "Selective Private Function Evaluation with Applications to Private Statistics," *Proc. 20th Ann. ACM Symp. Principles of Distributed Computing,* pp. 293-304, 2001.

[7] J. Canny, "Collaborative Filtering with Privacy," *Proc. 2002 IEEE Symp. Security and Privacy,* pp. 45-57, 2002.

[8] R. Chen, K. Sivakumar, and H. Kargupta, "Learning Bayesian Network Structure from Distributed Data," *Proc. SIAM Int'l Data Mining Conf.,* pp. 284-288, 2003.

[9] R. Chen, K. Sivakumar, and H. Kargupta, "Collective Mining of Bayesian Networks from Distributed Heterogeneous Data," *Knowledge Information Syststems,* vol. 6, no. 2, pp. 164-187, 2004.

[10] D.M. Chickering, "Learning Bayesian Networks is NP-Complete," *Learning from Data: Artificial Intelligence and Statistics V,* pp. 121-130, 1996.

[11] G. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data," *Machine Learning,* vol. 9, no. 4, pp. 309-347, 1992.

[12] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard.* Springer-Verlag, 2002.

[13] V. Estivill-Castro and L. Brankovic, "Balancing Privacy against Precision in Mining for Logic Rules," *Proc. First Int'l Data Warehousing and Knowledge Discovery,* pp. 389-398, 1999.

[14] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy Preserving Mining of Association Rules," *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 217-228, 2002.

[15] M. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," *Advances in Cryptology—Proc. EUROCRYPT 2004,* pp. 1-19, Springer-Verlag, 2004.

[16] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On Private Scalar Product Computation for Privacy-Preserving Data Mining," *Information Security and Cryptology—Proc. ICISC*, vol. 3506, pp. 104-120, 2004.

[17] O. Goldreich, S. Micali, and A. Wigderson, "How to Play ANY Mental Game," *Proc. 19th Ann. ACM Conf. Theory of Computing*, pp. 218-229, 1987.

[18] O. Goldreich, *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge Univ. Press, 2004.

[19] The Health Insurance Portability and Accountability Act of 1996, http://www.cms.hhs.gov/hipaa, 1996.

[20] G. Jagannathan, K. Pillaipakkamnatt, and R.N. Wright, "A New Privacy-Preserving Distributed k-Clustering Algorithm," *Proc. Sixth SIAM Int'l Conf. Data Mining*, 2006.

[21] G. Jagannathan and R.N. Wright, "Privacy-Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 593-599, 2005.

[22] E. Johnson and H. Kargupta, "Collective, Hierarchical Clustering from Distributed, Heterogeneous Data," *Lecture Notes in Computer Science*, vol. 1759, pp. 221-244, 1999.

[23] M. Kantarcioglu and C. Clifton, "Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data," *Proc. ACM SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD '02)*, pp. 24-31, June 2002.

[24] M. Kantarcioglu and J. Vaidya, "Privacy Preserving Naive Bayes Classifier for Horizontally Partitioned Data," *Proc. IEEE Workshop Privacy Preserving Data Mining*, 2003.

[25] M. Kantarcioglu, J. Jin, and C. Clifton, "When Do Data Mining Results Violate Privacy?" *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 599-604, 2004.

[26] O. Kardes, R.S. Ryger, R.N. Wright, and J. Feigenbaum, "Implementing Privacy-Preserving Bayesian-Net Discovery for Vertically Partitioned Data," *Proc. Proc. Int'l Conf. Data Mining Workshop Privacy and Security Aspects of Data Mining*, 2005.

[27] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the Privacy Preserving Properties of Random Data Perturbation Techniques," *Proc. Third IEEE Int'l Conf. Data Mining*, pp. 99-106, 2003.

[28] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, "Collective Data Mining: A New Perspective towards Distributed Data Mining," *Advances in Distributed and Parallel Knowledge Discovery*, AAAI/MIT Press, 2000.

[29] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining," *J. Cryptology*, vol. 15, no. 3, pp. 177-206, 2002.

[30] K. Liu, H. Kargupta, and J. Ryan, "Multiplicative Noise, Random Projection, and Privacy Preserving Data Mining from Distributed Multi-Party Data," Technical Report TR-CS-03-24, Computer Science and Electrical Eng. Dept., Univ. of Maryland, Baltimore County, 2003.

[31] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—A Secure Two-Party Computation System," *Proc. 13th Usenix Security Symp.*, pp. 287-302, 2004.

[32] D. Meng, K. Sivakumar, and H. Kargupta, "Privacy-Sensitive Bayesian Network Parameter Learning," *Proc. Fourth IEEE Int'l Conf. Data Mining*, pp. 487-490, 2004.

[33] D.E. O'Leary, "Some Privacy Issues in Knowledge Discovery: The OECD Personal Privacy Guidelines," *IEEE Expert*, vol. 10, no. 2, pp. 48-52, 1995.

[34] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residue Classes," *Advances in Cryptography—Proc. EUROCRYPT '99*, pp. 223-238, 1999.

[35] European Parliament, "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data," *Official J. European Communities*, p. 31 1995.

[36] European Parliament, "Directive 97/66/EC of the European Parliament and of the Council of 15 December 1997 Concerning the Processing of Personal Data and the Protection of Privacy in the Telecommunications Sector," *Official J. European Communities*, pp. 1-8, 1998.

[37] S. Rizvi and J. Haritsa, "Maintaining Data Privacy in Association Rule Mining," *Proc. 28th Very Large Data Bases Conf.*, pp. 682-693, 2002.

[38] S. Stolfo, A. Prodromidis, S. Tselepis, W. Lee, D. Fan, and P. Chan, "JAM: Java Agents for Meta-Learning over Distributed Databases," *Knowledge Discovery and Data Mining*, pp. 74-81, 1997.

[39] H. Subramaniam, R.N. Wright, and Z. Yang, "Experimental Analysis of Privacy-Preserving Statistics Computation," *Proc. Very Large Data Bases Worshop Secure Data Management*, pp. 55-66, Aug. 2004.

[40] J. Vaidya and C. Clifton, "Privacy Preserving Association Rule Mining in Vertically Partitioned Data," *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 639-644, 2002.

[41] J. Vaidya and C. Clifton, "Privacy-Preserving k-Means Clustering over Vertically Partitioned Data," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 206-215, 2003.

[42] J. Vaidya and C. Clifton, "Privacy Preserving Naive Bayes Classifier on Vertically Partitioned Data," *Proc. 2004 SIAM Int'l Conf. Data Mining*, 2004.

[43] R.N. Wright and Z. Yang, "Privacy-Preserving Bayesian Network Structure Computation on Distributed Heterogeneous Data," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 713-718, 2004.

[44] K. Yamanishi, "Distributed Cooperative Bayesian Learning Strategies," *Information and Computation*, vol. 150, no. 1, pp. 22-56, 1999.

[45] Z. Yang and R.N. Wright, "Improved Privacy-Preserving Bayesian Network Parameter Learning on Vertically Partitioned Data," *Proc. Int'l Conf. Data Eng. Int'l Workshop Privacy Data Management*, Apr. 2005.

[46] A. Yao, "How to Generate and Exchange Secrets," *Proc. 27th IEEE Symp. Foundations of Computer Science*, pp. 162-167, 1986.

Zhiqiang Yang received the BS degree from the Department of Computer Science at Tianjin University, China, in 2001. He is a currently a PhD candidate in the Department of Computer Science at the Stevens Institute of Technology. His research interests include privacy-preserving data mining and data privacy.

Rebecca Wright received the BA degree from Columbia University in 1988 and the PhD degree in computer science from Yale University in 1994. She is an associate professor at Stevens Institute of Technology. Her research spans the area of information security, including cryptography, privacy, foundations of computer security, and fault-tolerant distributed computing. She serves as an editor of the *Journal of Computer Security* (IOS Press) and the *International Journal of Information and Computer Security* (Inderscience) and was previously a member of the board of directors of the International Association for Cryptologic Research. She was program chair of Financial Cryptography 2003 and the 2006 ACM Conference on Computer and Communications Security and has served on numerous program committees. She is a member of the IEEE and the IEEE Computer Society.