Improved Privacy-Preserving Bayesian Network Parameter Learning on Vertically Partitioned Data*

Zhiqiang Yang Rebecca N. Wright

Computer Science Department
Stevens Institute of Technology
Hoboken, NJ 07030 USA

{zyang|rwright}@cs.stevens.edu

Abstract

Privacy concerns often prevent different parties from sharing their data in order to carry out data mining applications on their joint data. Privacy-preserving data mining seeks to address this by enabling parties to jointly compute a data mining algorithm on distributed data without sharing their data. In this paper, we address a particular data mining problem, that of learning the parameters of Bayesian network on a vertically partitioned database. We provide a simple privacy-preserving protocol for learning the parameters of Bayesian network on vertically partitioned databases. In comparison to the previously known solution for this problem (Meng, Sivakumar, and Kargupta, 2004), our solution provides better performance, full privacy, and complete accuracy. In combination with our previous work on privacy-preserving learning of Bayesian network structure on vertically partitioned databases, this work provides a complete privacy-preserving protocol for learning Bayesian networks (both structure and parameters) on vertically partitioned data, with very little overhead beyond computing the structure alone.

1 Introduction

Data mining provides the power to discover valuable patterns and trends from vast amount of data, and is playing an important role in knowledge discovery. Traditional data mining algorithms are based on the centralized model where the data is gathered into a central site. With the increased use of computers and networks in all aspects of everyday life, more and more data is collected by different parties. Some of this data is sensitive: its improper use can harm data subjects, data owners, data users, or other relevant parties. Many databases include sensitive data—e.g., databases containing financial transactions, health records and intelligence records from government agencies.

An increasing number of data mining applications deal with multiple databases held by different data owners. For example, different banks want to share their data to detect patterns of fraudulent activities, and government agencies and airlines desire to cooperate to detect patterns of terrorist behavior between their databases. However, privacy concerns prevent either party from sharing their raw data with the other. In some cases, regulations also prohibit different organizations from sharing their data with each other, e.g., HIPAA [HIP98] and the European privacy directives [Par95, Par98].

In recognition of these concerns, a growing class of *privacy-preserving* data mining algorithms have been developed to compute various data mining algorithms without requiring any single party to have access to the entire original dataset (e.g., [AS00, LP02, CIK$^+$01, ESAG02, VC02, KV03, AES03, DZ03, VC04, FNP04, WY04, MSK04]). In the strongest privacy setting, no party should learn anything about any other party's data beyond what it can infer from its own data and the resulting data mining product.

In principle, the elegant and general paradigm of secure multiparty computation provides privacy-preserving solutions for any distributed computation [GMW87, Yao86]. However, these solutions, while efficient on small inputs, are not efficient enough to be used for very large inputs such as typical databases for data mining applications. Instead, privacy-preserving data mining has concentrated on providing specific efficient solutions for specific data mining problems of interest.

*This work was supported by the National Science Foundation under grant number CCR-0331584.

Appears in *Pre-Proceedings of International Workshop on Privacy Data Management*, April 8-9, 2005, Tokyo, Japan. Held in conjunction with ICDE '05.

In previous work, we initiated the study of privacy-preserving computation of Bayesian networks [WY04]. Bayesian networks provide a concise way to represent mutual dependencies among different features in an application domain. They have proven to be a very useful tool in data mining and knowledge discovery. A Bayesian network consists of two parts: the network structure and the network parameters. The problem of learning a Bayesian network across multiple distributed databases has been addressed for horizontally and vertically partitioned data respectively [Yam97, CSK03], motivated by efficiency concerns rather than privacy concerns. Accordingly, they provide solutions that are more efficient than sending the data to a single location, but do not provide complete privacy. In particular, the solution for vertically partitioned data requires the parties to send some of their data elements either to each other or to a third party.

In [WY04], we presented a cryptographic privacy-preserving protocol for learning the structure of Bayesian network in a binary-valued database partitioned vertically between two parties. Meng, Sivakumar, and Kargupta presented a privacy-preserving method for learning the parameters of a Bayesian network [MSK04], assuming the network structure is already known, again for a binary-valued database partitioned vertically between two parties.

In this paper, we provide an alternate solution to that of [MSK04] for computing the parameters of a Bayesian network for a vertically partitioned database. In contrast to their solution, ours is more private, more efficient, and more accurate. In particular, our solution provides complete privacy, in that the only thing the parties learn about each other's inputs is the desired output, and completely accurate, in that the parameters computed are exactly what they would be if the data was centralized. In addition, our solution works for both binary and nonbinary discrete data. We provide a more detailed comparison of the two solutions in Section 4. As we also discuss in Section 4, our parameter-learning solution has the desirable property that it can be run simultaneously with the structure-learning solution of [WY04] with very little additional overhead over just learning the structure. All of these results are in the *honest-but-curious* model, described in more detail in Section 2.3.

In combination with our previous work on privacy-preserving learning of Bayesian network structure [WY04] on vertically partitioned databases, this work provides a complete privacy-preserving protocol for learning Bayesian networks (both structure and parameters) on vertically partitioned data.

In Section 2, we give some necessary background, including a brief review of Bayesian networks and of secure two-party computation. Our privacy-preserving Bayesian network parameter learning protocol is described in Section 3. We provide further discussion in Section 4 and conclude in Section 5.

2 Background

In this section, we give a brief overview of Bayesian networks and Bayesian network parameter learning. We also review some existing cryptographic tools that our solution makes use of: secure two-party computation and privacy-preserving scalar product protocols.

2.1 An Overview of Bayesian Networks. A Bayesian network (BN) is a graphical model that encodes probabilistic relationships among variables of interest, which can then be used for data analysis. Bayesian networks have been widely used in knowledge discovery and data mining applications [Hec95, CH92]. A simple example is shown in Figure 1.

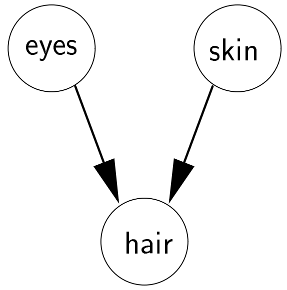
Formally, a Bayesian network for a set V of m variables is a pair $(B_{\mathrm{s}}, B_{\mathrm{p}})$, where the *network structure* $B_{\mathrm{s}} = (V, E)$ is a directed acyclic graph (DAG) whose nodes are the variables in V, and the *network parameters* B_{p} consist of descriptions of local probability distributions associated with each variable, described further below. We use a variable name v_i to denote both the variable and its corresponding node, and π_i to denote the parents of node v_i in B_{s}. The graph B_{s} describes conditional independence assertions about variables in V: the absence of an edge between nodes 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 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. Then the network parameters $B_{\mathrm{p}} = \{\Pr[v_i = v_{i_k} \mid \pi_i =$



hair	eyes, skin			
	brown, fair	brown, dark	blue, fair	blue, dark
red	1/3	1/8	1/6	1/2
blond	1/3	1/8	2/3	1/6
brown	1/3	3/4	1/6	1/3

Figure 1: A simple example showing a simple Bayesian network structure and the parameter table for node hair with parents eyes and skin.

$\phi_{ij}] : v_i \in V,\ 1 \le j \le q_i,\ 1 \le k \le d_i\}$ 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; that is $\Pr[v_i = v_{i_k} \mid \pi_i = \phi_{ij}] = \Pr[v_i = v_{i_k}]$. Figure 1 illustrates a simple example with a table showing the network parameters associated with the single node hair.

2.2 Learning Bayesian Network Parameters. As just discussed, the parameters $B_{\rm p}$ can be written as a table for each variable $v_i \in V$ describing the conditional distribution of v_i given the values of its parents π_i. Thus, the table for a variable is dependent on the variable, its parents, and the combined database. Figure 1 shows a simple hypothetical example, in which a node hair, describing hair color, has two parents eyes, describing eye color, and skin, describing skin color.

Given a database D (in the centralized setting, so the entire database is known to a single party), and given a network structure representing the conditional dependencies of the variables, each entry in a parameter table can be determined by counting how many of the records in the database have the specified values. (Note that this treats the data as if it *exactly* represents the underlying distribution. This assumption typically produces reasonable results if the dataset is large and the samples are independent.)

For example, the first entry in the table of Figure 1 is

$$\Pr[\mathsf{hair} = \mathsf{red} \mid \mathsf{eyes} = \mathsf{brown}, \mathsf{skin} = \mathsf{fair}],$$

and would be computed as

$$\frac{(\#\text{ records with } \mathsf{hair} = \mathsf{red}, \mathsf{eyes} = \mathsf{brown}, \mathsf{skin} = \mathsf{fair})}{(\#\text{ records with } \mathsf{eyes} = \mathsf{brown}, \mathsf{skin} = \mathsf{fair})}.$$

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 $\alpha_{ij} = \sum_{k=1}^{d_i} \alpha_{ijk}$. Hence, any table entry (i.e., one considering the kth instantiation of variable v_i and the jth instantiation of π_i) is computed as:

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

In Section 3.1, we show how this computation can be carried out in a distributed, privacy-preserving way when the database is vertically partitioned.

2.3 Secure Two-party Computation and Privacy-Preserving Data Mining. Secure two-party computation, introduced by Yao [Yao86], is a very elegant and general paradigm showing that, in principle, any distributed function can be efficiently computed using a distributed protocol in such a way that no party learns anything beyond its own input and output. Here, "efficient" means with a polynomially-bounded overhead over doing the corresponding centralized computation. Intuitively, the privacy properties of secure two-party computation (or secure multiparty computation, if there are more than two parties) are that no party learns any more information about other parties' data by engaging in the protocol than it would if it were simply handed the computed result. These definitions can be formalized (e.g. [Gol04]).

While this kind of general solution has often been considered largely theoretical, Malkhi et al. recently demonstrated it is quite efficient in practice for small inputs [MNPS04]. However, in the setting of data mining, inputs are typically very large, and stronger efficiency requirements must be met: an unspecified polynomial computation in the size of a large database is usually not acceptable. Rather, in the setting of a table-structured database, computations should be perhaps linear or even sublinear in the number of database rows or records, while larger bounds in the number of database columns or variables may be permitted.

Hence, work on privacy-preserving data mining has strived to provide the "ideal" privacy that secure two-party computation (or secure multiparty computation, if there are more parties) would provide, but at substantially better efficiency than the general solution provides. Sometimes, tradeoffs are made between privacy, accuracy, and efficiency.

In this paper, as in much of the privacy-preserving data mining work, we assume the parties are *honest-but-curious*. That is, the argument for privacy is made assuming that the participants follow their specified instructions. This seems an appropriate model for parties who wish to collaborate and can both benefit from engaging properly in the computation, but do not wish to fully trust each other with their data. Cryptographic techniques can also be used to provide privacy even if parties are behaving maliciously, but this additional security typically comes at too high a cost. (There are exceptions, such as [FNP04], which provides a private and very efficient set intersection computation that is secure even if the parties are malicious.)

Our privacy-preserving parameter computation makes black-box use of a secure two-party computation, such as [Yao86], but only on inputs whose size is sufficiently small (in particular, independent of the size of the database). In this way, we can obtain ideal privacy while maintaining reasonable efficiency. We also make black-box use of a particular privacy-preserving protocol, for scalar products, as we now discuss.

2.4 Privacy-Preserving Scalar Product Protocol. The scalar product, or inner product, of two vectors is frequently used computations in data mining applications. Given two vectors $\mathbf{z} = (x_1, \ldots, x_n)$ and $\mathbf{z}' = (y_1, \ldots, y_n)$ of the same length, their scalar product is $\mathbf{z} \cdot \mathbf{z}' = \sum_{i=1}^{n} x_i y_i$.

A privacy-preserving scalar product protocol is an important primitive for privacy-preserving data mining. Several such protocols have been proposed (e.g. [CIK$^+$01, DA01, VC02, LKR03, SWY04, WY04, FNP04, GLLM04, MSK04]), with varying degrees of security. A nice overview of the problem and the security properties of some solutions appears in [GLLM04]. In a privacy-preserving scalar product protocol, Alice holds $\mathbf{z}$ and Bob holds $\mathbf{z}'$. One or both parties are supposed to learn $\mathbf{z} \cdot \mathbf{z}'$. Ideally, neither party should learn anything about the other party's input beyond what is implied by his or her own vector and his or her result (where his or her result is either the scalar product or nothing, depending on whether the party was supposed to learn it), though some of the proposed protocols provide only weaker notions of privacy.

Our privacy-preserving protocol for learning Bayesian network parameters uses a fully secure privacy-preserving scalar product protocol in which only one party learns the result as a black-box building block. We only require use of the protocol for binary data, even if the database consists of non-binary data. We note that this can be done (e.g. as described in [CIK$^+$01, SWY04, GLLM04]) with complete cryptographic privacy using $O(kn)$ computation and communication, where n is the number of elements in the vectors and k is a security parameter representing the bit length of certain cryptographic keys. A reasonable value in practice for k is 1024.

In our case, we also make black-box use of a privacy-preserving scalar product protocol where, at the end, Alice and Bob learn *additive secret shares* of the resulting scalar product. For example, if Alice holds $\mathbf{z}$, Bob holds $\mathbf{z}'$, and the scalar product $\mathbf{z} \cdot \mathbf{z}'$ is known to be less than M, then Alice learns r_A and Bob learns r_B, where r_A and r_B are random integers, called *shares*, between 0 and $M - 1$ such that $(r_A + r_B) \bmod M = \mathbf{z} \cdot \mathbf{z}'$. Therefore, together Alice and Bob "know" $\mathbf{z} \cdot \mathbf{z}'$, but individually neither learns any information about its value. We call such a protocol a *privacy-preserving scalar product share protocol*. Most of the above protocols, including that of [SWY04, GLLM04], can be modified to act as privacy-preserving scalar product share protocols with no performance penalty.

Assuming the scalar product protocol is fully private, our resulting Bayesian network parameter-learning protocol is fully private as well.

3 Privacy-Preserving Protocol for Learning Bayesian Network Parameters

In this section, we present our main result: a privacy-preserving protocol for learning the Bayesian network parameters B_{p} for a distributed database that is vertically partitioned between two parties.

Alice and Bob jointly hold a vertically partitioned database. Specifically, Alice has D_A and Bob has D_B where D_A has the variables $V_A = \{a_1, ..., a_r\}$ and D_B has the variables $V_B = \{b_1, ..., b_t\}$, where the sets are assumed to be disjoint. Thus, $r + t = m$, and the variable set is $V = V_A \cup V_B$. Each database has n rows. There is a common *data handle* that links the rows in two databases (and that is not treated as a node of the Bayesian network). Without loss of generality, we assume that the row index is the data handle 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, $D = D_A \bowtie D_B$.

We assume that a Bayesian network B_s representing the joint data is already known to each party. (If the data is binary, this network structure could itself be computed in a privacy-preserving manner using [WY04].) Now the parties wish to learn the parameters B_p of the Bayesian network B_s. We seek a privacy-preserving solution, such that each party learns the parameters B_s, but learns "nothing else" about the other database. For example, returning to the example of Figure 1, suppose Alice owns the eyes variable and the skin variable and Bob owns the hair variable. Then Alice learns the parameter table for hair, so she learns that a third of the records for which she sees eyes = brown, skin = fair have hair = red. But she should not learn any information about which third of those records it is.

Accordingly, we describe a privacy-preserving procedure for determining the value of a single entry of the parameter table for a single variable. In order to learn the entire set B_p of parameters, this procedure would be carried out for each entry of the table for each variable. Our computation produces the identical value as would be computed if the entire database were available in a single location, but it does so without revealing anything beyond the information in the table entries to the two parties.

3.1 Computing the Parameters. Given a network structure, a (jointly held) database, a specific variable v_i with parents π_i, and instantiations v_{i_k} and ϕ_{ij}, respectively, we show a privacy-preserving protocol to compute the corresponding parameter value. As described in Section 2.2, the probabilities for the parameters are computed by considering the counts of various instantiations occurring in the data. Specifically, the parameter table entry we must compute is

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

In the privacy-preserving setting, we must provide a way to compute this value, α_{ijk}/α_{ij}, without sharing the databases or revealing any intermediate values such as α_{ijk} and α_{ij} (unless such values can be computed from the parameter table itself, in which case revealing them does not constitute a privacy breach).

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} if the variables in $(\{v\} \cup \pi_i) \cap V_A$ 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).

As was made use of by [WY04], 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 protocols (see Section 2.4). If a privacy-preserving scalar product share protocol is used, then this results in privacy-preserving computations for Alice and Bob to learn shares of α_{ijk} and α_{ij}. We describe this process in more detail below.

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

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} and α_{ij} from her own data, so she can compute α_{ijk}/α_{ij} 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}. In order for the parties to privately compute α_{ijk}, Alice prepares the vector $\mathbf{z}_A = \mathsf{compat}_A(v_{i_k}, \phi_{ij})$ and Bob prepares the vector $\mathbf{z}_B = \mathsf{compat}_B(v_{i_k}, \phi_{ij})$. They then use a privacy-preserving scalar product protocol for Alice to learn $\alpha_{ijk} = \mathbf{z}_A \cdot \mathbf{z}_B$. (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 computes α_{ijk}/α_{ij}, which she then also announces to Bob.

3. **The general case: the parent nodes are divided between Alice and Bob.** In the general case, Alice and Bob must jointly compute both α_{ijk} and α_{ij}. Alice prepares $\mathbf{z}_A = \mathsf{compat}_A(v_{i_k}, \phi_{ij})$ and $\mathbf{z}'_A = \mathsf{compat}_A(\phi_{ij})$. Bob prepares $\mathbf{z}_B = \mathsf{compat}_B(v_{i_k}, \phi_{ij})$ and $\mathbf{z}'_B = \mathsf{compat}_B(\phi_{ij})$. Using a privacy-preserving scalar product share protocol twice, Alice and Bob compute shares of $\mathbf{z}_A \cdot \mathbf{z}_B$ and $\mathbf{z}'_A \cdot \mathbf{z}'_B$. That is, Alice learns random shares s_A and r_A and Bob learns random shares s_B and r_B such that $(s_A + s_B) \bmod M = \alpha_{ijk}$ and $(r_A + r_B) \bmod M = \alpha_{ij}$ (where M is a parameter of the underlying scalar product share protocol). Thus, the desired parameter is $((s_A+s_B) \bmod M)/((r_A+r_B) \bmod M)$. In order to carry out this computation without revealing anything about s_A and r_A to Bob, or s_B and r_B to Alice, we make use of general secure two-party computation (see Section 2.3). 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 (and also independent of the database size n).

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.

3.2 Putting It All Together. In order to learn all the parameters B_{p}, Alice and Bob compute each entry of the table for each variable using the method just described in Section 3.1, as demonstrated in Figure 2.

> **Input:** Alice and Bob have a vertically partitioned database. 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 from Section 3.1. In the general case, this involves:
>
> a. Alice and Bob each create a vector to use for computing shares of α_{ijk} and α_{ij}.
>
> b. Alice and Bob run a two-party privacy-preserving scalar product share protocol to learn secret shares of α_{ijk} and α_{ij}.
>
> c. Alice and Bob run a secure two-party computation to compute the desired parameter α_{ijk}/α_{ij}.

Figure 2: Protocol Bayes

THEOREM 3.1. *Assuming the privacy-preserving scalar product and product share protocols and the secure two-party computation are correct and private, the Protocol Bayes is correct and private.*

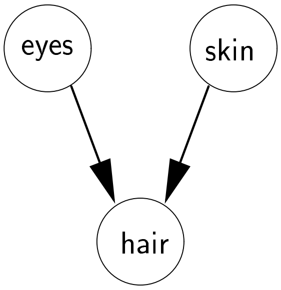
Proof (sketch): Assume the privacy-preserving scalar product and product share protocols and the secure two-party computation are correct and private. Then the correctness of Protocol Bayes is clear, because the values computed are precisely the desired parameters α_{ijk}/α_{ij}.

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 protecting via secret sharing, which reveals no additional information to the parties. ■

EFFICIENCY OF OUR SOLUTION. Let $d = \max d_i$ be the maximum number of possible values any variable takes on. Recall that n is the number of records in the database, while m is the number of variables. Let u be the maximum number of parents any node in the Bayesian network has (so $u \leq m - 1$). Typically $u \ll m \ll n$; also, typically d is fairly small (e.g., $d = 2$ for binary data), or can be made so by grouping values together. Additionally, k is a security parameter describing the length of cryptographic keys. Current values in practice for k would be 1024 or 2048, so for large databases, $k \ll n$.

In our solution, each parameter entry requires at least one of Alice and Bob to read through their entire database. Assuming all cases are the general case, which is the worst case for efficiency, the parameter entry requires Alice and Bob to each read through their databases to prepare the appropriate vector. The privacy-preserving scalar product share protocol requires kn computation and communication, and its results are two shares each, all of length k. Alice and Bob then engage in a secure two-party computation to compute the division of the sums of their respective shares, at a cost that is polynomial in k.

The number of parameter table entries for each variable is the number of possible instantiations of the variable and its parents—i.e., at most $d^{(u+1)}$. Hence, the total number of parameter table entries for the entire network is at most $md^{(u+1)}$, and the total computation and communication required is $O(md^{(u+1)}(kn + \mathsf{poly}(k)))$. Thinking of d, k, and u as constants since



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

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

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

Figure 3: Modified example from Figure 1. All parameters are shown.

they are likely to be small and essentially fixed in practice, this is $O(mn)$.

4 Discussion

We first provide a detailed comparison of the privacy, efficiency, and accuracy of our solution and that of [MSK04] (which we refer to as MSK). We show that our solution performs better than MSK in all three aspects. We then provide some additional discussion, including open problems.

4.1 Comparison to 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 methods such as we discuss in Section 2.4.

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). Due to MSK's leaked counts, 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.

Efficiency. As we discussed in Section 3.2, our solution runs in time $O(md^{(u+1)}(kn+\mathsf{poly}(k)))$. Taking $d = 2$ for purposes of comparison (since MSK assumes the data is binary-valued), this is $O(m2^{(u+1)}(kn + \mathsf{poly}(k)))$. In contrast, MSK runs in time $O(m(2^{(u+1)} + n^2))$. In particular, for a fixed security parameter k and maximum number u of parents of any node, as the database grows large enough that $k \ll n$, our efficiency grows linearly in n, while MSK grows as n^2.

We note that source of the quadratic growth in 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 a ideally-private cryptographic scalar product protocol such as the one we use, their performance would improve to $O(m(2^{(u+1)} + kn))$, a moderate efficiency improvement over our solution.

Accuracy. As shown in Section 3.2, our 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.

Privacy. As shown in Section 3.2, our solution provides ideal privacy, in the sense that the parties learn nothing about each other's 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 easily be 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 Figures 3 and 4.

In this example, Alice owns the variable eyes, while

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

Figure 4: Example showing that counts leak more information than parameters.

Bob owns skin and hair. In this case, each variable has two possible values, so the data can be considered binary. 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 Figure 3.

Figure 4 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 additionally given the count information, Alice 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 illustrates a general problem that can arise in more complex databases and Bayesian networks as well. It follows that our solution is inherently more private than MSK.

4.2 Additional Discussion. In this section, we discuss practical optimizations and some additional privacy issues.

PRACTICAL OPTIMIZATIONS. We note a few potential practical optimizations. For example, in order to reduce the number of rounds of communication, the parameters can all 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 entry in the tables. Additionally, if multiple nodes share the same set of parents, the same intermediate values can be reused for both. We also note that since the sum of shares in fact creates valid shares of the sum, the computation of each α_{ij} could be replaced by a sum of shares instead of a preparation of vectors and scalar product, again eliminating some passes through the database, as well as a scalar product computation.

We also point out that if this protocol is used in combination with our privacy-preserving Bayesian structure-learning protocol [WY04], then the shared α_{ijk} and α_{ij} values needed here are already computed there. In this case, the only additional overhead to compute the parameters is the secure two-party computation to divide each shared α_{ijk} by the shared α_{ij}.

ADDITIONAL PRIVACY ISSUES. One concern is that even if a privacy-preserving computation of Bayes network parameter is used, the resulting parameters—particularly when taken together with one party's database—reveal 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 privacy-preserving way, a phenomenon discussed nicely by Kantarcioglu, Jin, and Clifton [KJC04].

To limit this leakage, it might be preferable, for example, to have the parameter table 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 alternative would be to have the parties only learn shares of the resulting parameters, not the actual parameters. This points to an open research direction, which is how to design mechanisms such that the parties can then use the Bayesian network and shared parameters in a privacy-preserving interactive way to carry out classification or whatever task they seek to perform.

5 Conclusion

We have presented a privacy-preserving protocol for learning the parameters of Bayesian network in a database partitioned vertically between two parties.

Our protocol uses a privacy-preserving scalar product protocol and general secure two-party computation (for a simple function) as building blocks. Our solution provides ideal privacy and full accuracy. Compared to the previous solution [MSK04], our protocol is more private, more efficient, and more accurate.

In combination with our previous work on privacy-preserving learning of Bayesian network structure [WY04] on vertically partitioned databases, we can provide a complete privacy-preserving protocol for learning Bayesian networks with very little additional overhead over the structure computation alone.

References

[AES03] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 86–97. ACM Press, 2003.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450. ACM Press, May 2000.

[CH92] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347, 1992.

[CIK+01] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright. Selective private function evaluation with applications to private statistics. In *Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing*, pages 293–304. ACM Press, 2001.

[CSK03] R. Chen, K. Sivakumar, and H. Kargupta. Learning Bayesian network structure from distributed data. In *Proc. of the SIAM International Data Mining Conference*, San Franciso, USA, 2003.

[DA01] W. Du and M. Atallah. Privacy-preserving cooperative statistical analysis. In *Proc. 17th Annual Computer Security Applications Conference*, pages 103–110, 2001.

[DZ03] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505–510. ACM Press, 2003.

[ESAG02] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228. ACM Press, 2002.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004*, LNCS 3027, pages 1–19. Springer-Verlag, 2004.

[GLLM04] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the Seventh Annual International Conference in Information Security and Cryptology*, LNCS. Springer-Verlag, 2004. to appear.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proc. of the 19th Annual ACM Conference on Theory of Computing*, pages 218–229. ACM Press, 1987.

[Gol04] O. Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[Hec95] D. Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, Redmond, Washington, 1995. Revised June 96.

[HIP98] HIPAA. The health insurance portability and accountability act of 1996, October 1998. Available at `www.cms.hhs.gov/hipaa`.

[KJC04] M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2004.

[KV03] M. Kantarcioglu and J. Vaidya. Privacy preserving naive Bayes classifier for horizontally partitioned data. In *IEEE Workshop on Privacy Preserving Data Mining*, 2003.

[LKR03] 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 Engineering Department, University of Maryland, Baltimore County, 2003.

[LP02] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[MNPS04] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. In *Proc. Usenix Security Symposium 2004*, 2004.

[MSK04] D. Meng, K. Sivakumar, and H. Kargupta. Privacy-sensitive Bayesian network parameter learning. In *Proc. of the Fourth IEEE International Conference on Data Mining*, Brighton, UK, 2004.

[Par95] 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 Journal of the European Communities*, page 31, 1995.

[Par98] European Parliament. Directive 97/66/EC of the European Parliament and of the Council of 15 December 1997 concering the processing of personal data and the protection of privacy in the telecommunications sector. *Official Journal of the European Communities*, pages 1–8, 1998.

[SWY04] H. Subramaniam, R. N. Wright, and Z. Yang. Experimental analysis of privacy-preserving statistics computation. In *Proc. of the VLDB Worshop on*

Secure Data Management, pages 55–66, August 2004.

[VC02] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM Press, 2002.

[VC04] J. Vaidya and C. Clifton. Privacy preserving naive Bayes classifier on vertically partitioned data. In *2004 SIAM International Conference on Data Mining*, 2004.

[WY04] R. N. Wright and Z. Yang. Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–718. ACM Press, 2004.

[Yam97] K. Yamanishi. Distributed cooperative Bayesian learning strategies. In *Proc. of the 10th Annual Conference on Computational Learning Theory*, pages 250–262. ACM Press, 1997.

[Yao86] A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.