In industrial and government settings, there is often a need to perform statistical analyses that require data stored in multiple distributed databases. However, the barriers to literally integrating these data can be substantial, even insurmountable. In this article we show how tools from information technology—specifically, secure multiparty computation and networking—can be used to perform statistically valid analyses of distributed databases. The common characteristic of these methods is that the owners share sufficient statistics computed on the local databases in a way that protects each owner’s data from the other owners. Our focus is on horizontally partitioned data, in which data records rather than attributes are spread among the databases. We present protocols for securely performing regression, maximum likelihood estimation, and Bayesian analysis, as well as secure construction of contingency tables. We outline three current research directions: a software system implementing the protocols, secure EM algorithms, and partially trusted third parties, which reduce incentives for owners to be dishonest.

KEY WORDS: Data confidentiality; Distributed databases; Secure multiparty computation.

1. INTRODUCTION

Many scientific, business, or government investigations require statistical analyses that integrate data stored in multiple distributed databases. For example, in Section 3.2 we discuss a regression analysis on chemical databases owned by multiple pharmaceutical companies. The integrated analysis is more informative in identifying molecular features influencing biological activity compared with separate analyses of the individual databases.

At the same time, the barriers to actually integrating multiple databases are numerous. In industrial settings, proprietary data is a principal impediment to integration. Scale is another barrier; despite advances in networking technology, the only sure way to move multiple terabytes of data overnight may be FedEx. In such settings as official statistics (Karr, Lin, Reiter, and Sanil 2004, 2005b; Sanil, Karr, Lin, and Reiter 2004a,b) or homeland security (Karr et al. 2006), where database owners may be states or multiple federal agencies and confidentiality of data subjects is paramount, laws often prohibit moving or sharing data.

In this article we show that how for many analyses it is not necessary to move or share individual data records. Instead, using techniques from computer science known generically as secure multiparty computation (SMPC), the database owners can share summaries (in many cases, sufficient statistics) of the data anonymously, but in a way to allow statistically valid analyses.

The article is organized as follows. Section 2 introduces SMPC and presents the one concrete version—secure summation—that we require. Section 3 presents some analyses for horizontally partitioned data, including data integration, regression, construction of contingency tables, maximum likelihood for exponential families, and Bayesian analyses. Section 4 gives a brief discussion of vertically partitioned data. Section 5 describes three directions of ongoing research at the National Institute of Statistical Sciences (NISS), and Section 6 gives some conclusions.

1.1 Problem Formulation

Consider a global database that is partitioned among a number of owners—for concreteness, companies or government agencies. These database owners wish to perform a statistically valid analysis of the global database but without ever actually creating it. Reasons why creating it may be difficult or impossible were mentioned earlier and range from protecting proprietary data to scale to confidentiality. The constraints preclude the use of trusted third parties. The process must produce the same answer as would have been obtained from the global database. In addition, all objects that statisticians customarily consider part of the analysis must be calculated. To illustrate, for a regression, not only the estimated coefficients, but also their standard errors, measures of model fit, and even characteristics of residuals should be provided.

© 2007 American Statistical Association and the American Society for Quality
TECHNOMETRICS, AUGUST 2007, VOL. 49, NO. 3
DOI 10.1198/004017007000000209
The protocols that we describe protect the owners from one another in the sense that whereas each owner can compare the global analysis to the same analysis on its own data, it cannot attribute any characteristics of the difference to specific other databases. The extent to which the individual data subjects are protected is not yet understood (see Sec. 6).

Finally, implementations must be both computationally feasible and secure from tampering by malicious owners or external parties.

1.2 Data Partitioning Models

For the remainder of this article, a database is a flat file in which rows represent data subjects and columns represent attributes. We focus on horizontally partitioned data in which the data subjects are partitioned among the databases, and each owner has the same attributes for all of its subjects. This is illustrated Figure 1(a). For vertically partitioned data, the attributes rather than the subjects are partitioned among the databases, as shown in Figure 1(b). We discuss more complex partitions in Section 5.2.

Both partitioning models engender significant metadata issues. For horizontally partitioned data, for example, the database owners need to ascertain that their sets of data subjects are in fact disjoint and must have the attributes in the same order and in the same units. Vertically partitioned data entails the much more difficult problem of securely linking records across the databases.

2. SECURE MULTIPARTY COMPUTATION

In this section we give a brief introduction to SMPC. General references are the work of Goldwasser (1997) and Yao (1982).

2.1 Generalities

Consider \( K \) data owners with values \( v_1, \ldots, v_K \) that wish to evaluate a known function \( f \), which is typically symmetric in its arguments, at these values, subject to three constraints:

C1. The correct value \( f(v_1, \ldots, v_K) \) is obtained and made known to all owners.
C2. Each owner \( j \) learns no more about the other owners’ values \( V_{-j} = \{ v_k : k \neq j \} \) than it can deduce from \( v_j \) and \( f(v_1, \ldots, v_K) \).
C3. No trusted third party is part of the process.

Our model of a trusted third party is an organization, person, or computer to which the owners would submit their values \( v_j \), would calculate \( f(v_1, \ldots, v_K) \), and would inform the owners of the result. The challenge is that C1 and C2 say that the process must be as effective as if there were a trusted third party, whereas C3 forbids this. In Section 5.1 we allow a central server (see Fig. 4) that has knowledge of the order in which database owners input information, but not of data values. We describe a stronger relaxation in Section 5.3.

The computer science literature contains a large number of articles on the theory of SMPC. Few of these describe implemented algorithms or functioning software systems. Some procedures for SMPC involve cryptography, whereas others depend on some form of randomization. The latter are exemplified by secure summation, which we discuss in Section 2.2.

Nearly all protocols for SMPC assume that the owners are semi-honest. Specifically, this means that they perform agreed-on computations correctly and that they use their true data. If the protocol is iterative, then they are permitted to retain the results of intermediate computations. In Section 5.3 we examine the consequences of owners that violate the assumption of semi-honesty and propose one way to mitigate these consequences.

2.2 Secure Summation

In this article we use one form of SMPC: secure summation. In the notation of Section 2.1,

\[
f(v_1, \ldots, v_K) = v_1 + \cdots + v_K.
\]

Denote this sum by \( V \).

The secure summation protocol (Benaloh 1987) is depicted graphically in Figure 2. Assume for simplicity that the \( v_k \) are integers. The protocol comprises the following steps:

- **Owner 1:** \( v_1 = 29 \) and \( R = 1003 \)
- **Owner 2:** \( v_2 = 5 \)
- **Owner 3:** \( v_3 = 153 \)
- **Final step:** calculate and share \( V = (168 - R) \mod 1024 = 187 \)

Figure 2. Pictorial representation of secure summation.
• Initialization. Owner 1 generates (and retains) a very large random integer \( R \), adds \( R \) to its value \( v_1 \), and sends the sum \( R + v_1 \) to owner 2.

• Iteration. Because \( R \) is random, owner 2 learns effectively nothing about \( v_1 \) from \( R + v_1 \). It simply adds its value \( v_2 \) to \( R + v_1 \), sends the result to owner 3, and so on.

• Sharing. Finally, owner 1 receives \( R + v_1 + \cdots + v_K = R + V \) from owner \( K \), subtracts \( R \), and shares the result \( V \) with the other owners.

After completing the protocol, each owner \( j \) knows only \( v_j \) and \( V \), from which it can calculate \( V_{-j} = \sum_{k \neq j} v_k \). However, at least in the absence of external knowledge, it cannot resolve \( V_{-j} \) into its components \( v_k, k \neq j \), much less associate these with specific other owners.

Figure 2 depicts an extra layer of protection. Suppose that \( V \) is known to lie in the range \([0, m]\), where \( m \) is a very large number, say \( 2^{100} \), known to all of the owners. Then \( R \) can be chosen randomly from \([0, \ldots, m - 1]\) and all computations can be performed modulo \( m \), which provides added protection to owners with large values of \( v_j \) that are early in the process.

As an illustration of this, suppose that the owners have income data and wish to compute the global average income. Let \( n_j \) be the number of records in owner \( j \)'s database and let \( I_j \) be the sum of their incomes. The quantity to be computed is

\[
\bar{I} = \frac{\sum_j I_j}{\sum_j n_j},
\]

(1)

the numerator of which can be computed using secure summation on the \( I_j \)'s and the denominator computed using secure summation on the \( n_j \)'s. Each owner can then calculate \( \bar{I} \).

Although the secure summation protocol is simple, implementation presents a number of challenges. For example, neither another owner nor an outsider should be able to masquerade as an owner, nor should the process be visible to, or corruptible by, outsiders. In addition, secure summation is vulnerable to collusion among a subset of the database owners, although there are known ways (Benaloh 1987) to address this problem. The secure computation system (SCS) described in Section 5.1 implements these protections and more.

3. SECURE ANALYSIS OF HORIZONTALLY PARTITIONED DATA

In this section we describe various secure analyses for horizontally partitioned data, including data integration, regression, construction of contingency tables, maximum likelihood estimation, and calculation of posterior distributions for Bayesian analyses. The common theme is to use secure summation to calculate sufficient statistics that are additive across the databases.

A significant shortcoming of the approach is that the analysis to be performed must be specified beforehand, together with any transformations of attributes. Such statistical procedures as exploratory data analysis and determining appropriate transformations of attributes simply do not fit the paradigm very well.

Moreover, although we do not emphasize it here, significant preprocessing effort is required. For example, all database owners need to have data attributes in the same units and in the same order. It may be necessary to eliminate (duplicate) data records that appear in more than one database, which is not straightforward without revealing details of the databases.

3.1 Secure Data Integration

The formulation in Section 1.1 prohibits creation of the global database as a means of performing the analysis. However, there may be situations (see Sec. 3.3 for one) in which the owners are willing to create and share the global database, provided that the sources of data elements are protected. For example, owners who are retailers may be willing to share information about their customers, including the customers’ identities, provided that they do not reveal who is whose customer.

Of course, from the global and its own database, each owner can recognize customers who are not its own, but an owner should not be able to infer which other owner’s (or owners’) customers they are.

There are circumstances in which such secure data integration (SDI) is simply not possible. The most apparent of these is when data values themselves reveal the source. This would happen if, for example, the owners were state education agencies in the United States and the data were student-level data containing ZIP code of residence.

We now sketch a protocol for SDI. Conceptually it is straightforward, provided that the star topology and encryption mechanisms of the NISS SCS described in Section 5.1 are used. The owners incrementally contribute data in a random round-robin order known to a central server but not to them. Figure 4 in Section 5.1 illustrates the star topology of the system. The circulating database is assumed to be encrypted using a key known to the clients but not the server, and encryption/decryption steps are omitted from the description. The SDI protocol is as follows:

• Initial round. Each owner receives from the server a circulating database. It records all elements in it, adds to the database a random number of its records (the need for this randomization and issues associated with it are discussed later), and sends the result back to the server.

• Intermediate rounds. Each successive time that an owner receives the circulating database, it first removes elements that it put on during the preceding round. It can recognize these, and removing them is safe because they have already been recorded by all the other owners. It then records all other data in the database that have been added by other owners. Finally, it adds a random subset of its remaining data and returns the database to the server.

• Penultimate round. Each owner removes its own previous data and records other owners’ previous data as for intermediate rounds, but now puts in all of its remaining data. Which round is the penultimate round may be owner-dependent.

• Final round. Following its penultimate round, each owner removes the data that it inserted at that time and sends the database back to the server. The last owner to do this will be left with an empty database, and the process will be complete. Until that happens, each owner will continue to receive the database, will record data added by other owners, and send the database on.

Note that because of the round-robin system, between any two times that an owner receives the database, all other owners (who have not exhausted their own data) will have contributed
to it, so there is no means of limiting which other owners may have contributed the new records. The “remove what was put in on the previous round” achieves computational feasibility by controlling the size of the database; the size of circulating database is approximately constant rather than growing linearly with the number of rounds. With \( n \) the size of the global database, this reduces the communication overhead from \( O(n^2) \) to \( O(n) \).

We judge the effectiveness of the protocol using Bayesian methods. Assume that the database owners know the sizes \( n_j \) of each other’s databases. (If they were state agencies, then this would be plausible, because the sizes would often be public information.) Owner \( j \)'s (naive) prior distribution on the source of any record not from its own database is

\[
P(\text{Source} = \text{owner } k) = \frac{n_k}{\sum_{\ell \neq j} n_\ell}.
\]  

(2)

The extent to which observing the protocol enables an owner to improve its estimates of records’ sources—posterior distributions, given what is observed about the protocol—over the priors in (2) measures the lack of protection of sources.

In early rounds, randomizations that add a constant fraction of each owner’s records are revealing: the larger the pool received by an owner, the more likely it is to contain records from large databases. At the other extreme, if each owner were to put in the same (expected) number of records on each round, then in late rounds, only owners with large databases would be contributing. In empirical experiments, moderate numbers of rounds and multinomial distributions appear to address both considerations.

This protocol violates the strict interpretation of condition C3 in Section 2, because the server is trusted by the owners to know the order in which they contribute to the data pool. The encryption described in Section 5.1 prevents the server from learning any data values. Moreover, no owner can know even its place (i.e., first, second, …, last) in the order. Without this protection, the second owner in the process would know that all records that it received in the initial round came from a single database, and might use relationships deduced from those records to identify their owner.

### 3.2 Secure Regression

Assume that the data consist of \( p + 1 \) numerical attributes of each data subject, so that owner \( j \)'s data on its \( n_j \) subjects consist of \( p \) predictors \( x^j \) and a response \( y^j \). The owners wish to fit the usual linear model

\[
y = X\beta + \epsilon
\]  

(3)
to the “global” data

\[
X = \begin{bmatrix} x^1 \\ \vdots \\ x^K \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} y^1 \\ \vdots \\ y^K \end{bmatrix}.
\]

We embed the constant term of the regression in the first predictor by putting \( x_0^j \equiv 1 \) for all \( j \). Typically, the predictors and response would be centered at mean values (or standardized). Because the means (and standard deviations) are the global ones, a preliminary round of secure computation is needed to compute them.

We assume for simplicity that \( \text{cov}(\epsilon) = \sigma^2 I \), in which case the least squares estimator for \( \beta \) is

\[
\hat{\beta} = (X^T X)^{-1} X^T y.
\]  

(4)
The crucial point is that the global \((p + 1) \times (p + 1)\) matrix

\[
[X y]^T [X y] = \begin{bmatrix} X^T X & X^T y \\ y^T X & y^T y \end{bmatrix},
\]

from which \( \hat{\beta} \) can be calculated using (4), is additive over the owners,

\[
[X y]^T [X y] = \sum_{k=1}^{K} [X^k y^k]^T [X^k y^k];
\]  

(5)

therefore, \([X y]^T [X y]\) can be computed entrywise using secure summation. In the SCS described in Section 5.1, a single secure summation is performed on \((p + 1) \times (p + 1)\) matrices, whose entries are initialized with independently generated random numbers.

We illustrate with a data set of 1,318 chemical compounds, in which the response is water solubility and the 91 predictors are a constant and 90 chemical features of the compounds (Karr et al. 2005a). Four companies were created with databases containing 499, 572, 16(1), and 231 compounds. This example mimics real-world heterogeneity, where each company’s database contains compounds with features that are absent from all compounds in the other companies’ databases. This sharpens the incentive for each company to participate, because it can learn about features for which it has no data. Of course, company 3 has greatest incentive to participate, because it cannot even do the regression on its own.

There is no need to illustrate that the secure regression protocol produces the correct answer; it is guaranteed to do so if the database owners are semi-honest. It is more pertinent to ask what the database owners gain and lose from the global analysis. Figure 3 compares that analysis with single-company analyses. Panels (a), (b), and (c) are scatterplots of the 91 regression coefficients for companies 1, 2, and 4 (y-axis) against the coefficients for the global (four-company) regression (x-axis). Coefficients with y-values of 0 correspond to features missing from each company’s database. Not surprisingly, the match between each of company 1, 2, and 4’s coefficients and the global coefficients depends on the size of its database; the larger the database, the better the match.

A natural question, if the companies knew the sizes of each others’ databases, is whether companies 1, 2, and 4 should allow company 3 to participate, because it has so little data. Figure 3(d) sheds some light on this. There the x-axis again contains the coefficients for the four-company regression, and the y-axis contains those for the regression for companies 1, 2, and 4, excluding company 3. Although close, the coefficients are not identical, showing that companies 1, 2, and 4 do gain from including company 3.

Other issues are relevant to this example. In particular, there is no inherent way to handle data quality that varies across companies, although a secure weighted regression can be performed
in the same manner as a secure ordinary regression, provided that weights can be agreed on.

Calculation of \( \hat{\beta} \) is only part of a valid, useful regression. Various other objects can be calculated from \( [Xy]^T[Xy] \) or by using secure summation directly. These include the coefficient of determination \( R^2 \), the least squares estimate \( \hat{\sigma}^2 \) of the error variance, and the “hat” matrix \( H = X(X^T X)^{-1} X^T \), which can be used to identify outliers (Karr et al. 2005b, 2006). It is also possible to use the secure data integration algorithm of Section 3.1, together with methods for constructing privacy-preserving synthetic residuals in ordinary regressions (Reiter 2003), to create secure synthetic residuals (Karr et al. 2006).

Other analyses that are sufficiently similar to regression can be performed in the same way. For example, the constant variance assumption preceding (4) can be relaxed. Analyses using adaptive regression splines have been treated by Ghosh, Reiter, and Karr (2007).

### 3.3 Secure Construction of Contingency Tables

The algorithm for secure data integration described in Section 3.1 has an important, but indirect application: securely constructing contingency tables containing counts or sums. Let \( \mathcal{D} \) be a database containing categorical attributes \( A_1, \ldots, A_J \). The associated contingency table is the \( J \)-dimensional array \( T \) defined by

\[
T(a_1, \ldots, a_J) = \# \{ r \in \mathcal{D} : r_1 = a_1, \ldots, r_J = a_J \},
\]

where each \( a_j \) is a possible value of the categorical attribute \( A_j \) (e.g., if \( A_1 \) corresponds to gender, then possible values of \( a_1 \) are...
are “female” and “male”), \#\{\cdot\} denotes “cardinality of \cdot,” and \( r_i \) is the \( i \)th attribute of record \( r \). The \( J \)-tuple \((a_1, \ldots, a_J)\) is called the cell coordinates. More generally, contingency tables may contain sums of numerical variables rather than counts; the procedure described below works in either case.

An array is not a feasible data structure for tables with very large numbers of cells. Large tables are invariably sparse, however, with relatively few cells having nonzero counts. For instance, the table associated with the U.S. Census long form, which contains 52 questions, has more than \( 10^{15} \) cells, but at most approximately \( 10^6 \) (the number of households in the U.S.) of these are nonzero. The sparse representation of a table is the data structure of (cell coordinate, cell count) pairs

\[
\{ (a_1, \ldots, a_J, T(a_1, \ldots, a_J)) : T(a_1, \ldots, a_J) \neq 0 \},
\]

Algorithms that use this sparse representation data structure have been developed that support virtually all important table operations (Moore and Lee 1998).

Consider now the problem of securely building a contingency table from databases \( D_1, \ldots, D_K \) containing the same categorical attributes for disjoint sets of data subjects. Given the tools described in Sections 3.1 and 2.2, this process is straightforward. The steps are as follows:

1. List nonzero cells. Use secure data integration to build the list \( L \) of cells with nonzero counts. The “databases” being integrated in this case are the owners’ individual lists of cells with nonzero counts. The protocol in Section 3.1 allows each owner to not reveal in which cells it has data.
2. Count nonzero cells. For each cell in \( L \), use secure summation to determine the associated count or sum.

### 3.4 Secure Maximum Likelihood Estimation

Suppose now that the owners’ databases partition a global database \( \{x_i\} \) modeled as independent samples from an unknown density \( f(\theta, \cdot) \) belonging to an exponential family. Specifically, suppose that

\[
\log f(\theta, x) = \sum_{\ell=1}^{L} c_\ell(x)d_\ell(\theta), \tag{7}
\]

Then, under the independence assumption, the global log-likelihood function is

\[
\log L(\theta, x) = \sum_{\ell=1}^{L} d_\ell(\theta) \left( \sum_{k=1, x_i \in D_k}^{K} c_\ell(x_i) \right), \tag{8}
\]

where \( D_k \) is the database of owner \( k \).

Assuming that the owners have agreed in advance on the model (7), they can use secure summation to compute each of the \( L \) terms within the brackets in (8), and then each can maximize the likelihood function by whatever means it wishes.

In principle, a secure (iterative) Newton–Raphson algorithm for numerical maximization of a nonexponential family likelihood function is possible. Assume that \( \theta = (\theta_1, \ldots, \theta_m) \), and let \( \ell \) be the log-likelihood function. The straightforward aspect is that secure summation can be used to compute the gradient vector

\[
\nabla \ell(\theta_0) = \left( \sum_{k=1}^{K} \sum_{x_i \in D_k} \frac{\partial f(\theta, x_i)/\partial \theta_1}{f(\theta, x_i)} \right)_1, \ldots,
\]

\[
\sum_{k=1}^{K} \sum_{x_i \in D_k} \frac{\partial f(\theta, x_i)/\partial \theta_m}{f(\theta, x_i)} \right)_\theta_0 \tag{9}
\]

and Hessian matrix \( D^2 \ell(\theta_0) \) of \( \ell \) at a given parameter value \( \theta_0 \). From these, each database owner can compute a Newton–Raphson step,

\[
\Delta \theta = -[D^2 \ell(\theta_0)]^{-1} \nabla \ell(\theta_0), \tag{10}
\]

and a new value \( \theta' = \theta_0 + \Delta \theta \), and the process can proceed iteratively.

There are complications, however. The owners need agreed-on expressions for \( \nabla \ell \) and \( D^2 \ell \) or algorithms to compute these expressions that perform identically on all owners’ machines. In addition, before the process can proceed safely from one iteration to the next, there must be a way to verify that all owners have the same value for \( \Delta \theta \) in (10). In contrast, secure summation presumes only that the owners’ machines can add, while secure regression requires that each machine be able to perform matrix inversions. However, the matrix inversion for secure regression occurs only locally and at the end of the process. An owner whose inversion algorithm is flawed does not calculate the correct answer, but this does not prevent other owners from calculating the correct answer. Failure of any owner to perform secure Newton–Raphson, on the other hand, affects the results for all owners. Finally, there must be an agreed-on criterion for convergence of the algorithm.

Note also that once convergence is reached, (9) can be used for computing estimated Fisher information matrices through the substitution estimator \( \hat{I} = \nabla \ell(\hat{\theta}_{\text{MLE}})\nabla \ell(\hat{\theta}_{\text{MLE}})^T \).

### 3.5 Secure Posterior Distributions

Any owner who wishes also can perform Bayesian analyses, with whatever prior distribution \( \pi \) it chooses, because the relevant posterior distribution is computable from \( \pi \) and the likelihood function \( L(\theta, x) \) of (8).

### 4. VERTICALLY PARTITIONED DATA

Secure analysis of vertically partitioned data is substantially more complex than analysis of horizontally partitioned data. Here we outline some of the issues and approaches, focusing on linear regression.

The preprocessing alluded to at the beginning of Section 3 is more complex than for horizontally partitioned data. Records in multiple databases must be aligned with one another, which requires a common primary key (e.g., social security numbers) for the databases. (Contrast this with Sec. 3, in which database keys are immaterial.) The records that the databases have in common must be determined, and a decision must be made about how to handle incomplete records (see Sec. 5.2). Even at this stage, owners are yielding information to one another by revealing which records appear in whose databases.
Even assuming that these problems can be handled, statistical analysis is not simple. One approach to regression (Sanil et al. 2004a) is to use secure matrix products to compute off-diagonal blocks in the “full data” covariance matrix. Each such block involves only two database owners, A and B, that have disjoint sets of p and q attributes, for the same n data subjects. Together, their data matrix is

\[ \begin{bmatrix} X^A & X^B \end{bmatrix} \]

We assume that the two data matrices are of full rank; if not, then the owners remove linearly dependent columns.

The owners now need to compute securely and share the \((p \times q)\)-dimensional matrix \((X^A)^T X^B\). An optimal protocol in the sense of Section 2 ensures that neither learns more about the other’s data by using the protocol than it would learn if an omniscient third party were to tell it the result. From the perspective of fairness, the protocol should be symmetric in the amount of information exchanged. A protocol that achieves both of these goals (at least approximately) is as follows.

First, owner A generates a set of g orthonormal vectors \(Z_1, Z_2, \ldots, Z_g \in \mathbb{R}^n\) such that \(Z_i^T X_i = 0\) for all \(i, j\), and sends the matrix \(Z = [Z_1 Z_2 \ldots Z_g]\) to B. Methods for choosing g and generating Z has been presented by Sanil et al. (2004a). Second, B computes \(W = (I - ZZ^T)X^B\), where I is an identity matrix, and sends W to A. Finally, A calculates, and shares with B, \((X^A)^T W = (X^A)^T (I - ZZ^T)X^B = (X^A)^T X^B\).

The second equality holds because \((X^A)^T Z_i = 0\) for all \(i, j\).

It might appear that \(B^0\)’s data can be learned exactly because A knows both \(W\) and \(Z\). However, \(W\) has rank \((n - g) = (n - 2p)/2\), and thus A cannot invert it to obtain \(X^B\).

This protocol does involve some loss of protection for each owner’s data. To assess how much protection is lost, we note first that (even if there were a trusted third party), each owner learns, from \((X^A)^T X^B\), \(pq\) constraints on the other’s data, one for each element of this matrix. The protocol itself provides owner A about the additional knowledge that \(X^B\) lie in the \(g \approx (n/2)\)-dimensional subspace given by \(W = (I - ZZ^T)X^B\).

Thus owner A has a total of \(g + pq\) constraints on \(X^B\). Assuming, realistically, that \(n \gg pq\), owner A thus knows the approximately \((n/2)\)-dimensional subspace in which \(X^B\) lie.

The same is true for owner B’s knowledge of \(X^A\); thus, at a mechanical level, the protocol is symmetric in the information exchanged. At higher levels, however, symmetry can break down. For example, if A holds the response but none of its other attributes is a good predictor, whereas the attributes held by B are good predictors, then A learns more about B’s data than vice versa.

Application of the secure matrix product protocol to conduct secure linear regression analysis is straightforward. Altering notation for simplicity, let the matrix of all variables in the possession of the owners be \(D = [D_1, \ldots, D_p]\), with

\[ D_i = \begin{bmatrix} d_{1i} & \cdots & d_{pi} \\ \vdots & \ddots & \vdots \\ d_{ni} & \cdots & d_{ni} \end{bmatrix}, \quad 1 \leq i \leq p. \]
overhead (Ben-Or, Goldwasser, and Wigderson 1988). Second, authentication and encryption are much more efficient with the server. Finally, process management is vastly simplified.

On the other hand, because the server is not an owner, it must not be able to “read” any of the information that it is passing between clients, which leads (as described later) to dual encryption. In fact, in the SCS the server is not even aware which protocol the clients are performing.

We next present a simplified description of SCS functionality. We assume that whatever secure protocol is being performed requires only one contribution from each client. In terms of Section 3, this means everything except secure data integration and the first stage of secure contingency tables.

Set-up. Before actually performing the analysis, the owners/clients must agree on who is participating, which protocol (e.g., secure regression) will be performed, metadata issues, a time at which the protocol is to be performed, and a symmetric clients-only encryption key, which is not revealed to the server. The time and IP addresses of clients are transmitted to the server by means of a reservation system.

Log-in. At the reserved time, the server listens for clients to log in. As each client logs in, it receives the server’s public key, to use when sending messages to the server. It also generates a (private key, public key) pair for encryption of messages from the server to it, and sends the latter to the server.

Initiation. When all clients have logged in, the server randomly selects an order for the clients and sends an “initiate protocol” message to the first client. The structure and encryption of messages from the server to the client is shown in Figure 5(a). The entire message is encrypted using the client’s public key, which prevents it from being read by the other clients or an outsider. The tag is generated by the server and used for authentication: the server will not respond to any message not containing the correct tag. The client message contains information passed from the server to the client. Examples are “initiate protocol” and “protocol completed.” The remainder of the message is empty at the start of the process, or else (as described later) has been constructed and encrypted by another client. It is not readable by the server.

Client-Side Processing. When it receives a message from the server, a client decrypts it using its own private key, sets the tag aside, parses the client message, and acts accordingly. For example, if the message is “initiate protocol” and the clients have agreed to perform secure summation, then the client would generate the random number $R$, add its $v_j$ to it, construct the payload (which is simply $R + v_j$), set operation to “secure summation: add value,” which tells the next client what to do, concatenate the two, and encrypt the result with the clients-only key. It would then concatenate the saved tag, a server message and the encrypted operation and payload; encrypt that entire object with the server’s public key; and send it to the server.

When there are no problems, the server message will be of the form “client operation successful.” Were there a problem, for example, if a client in secure regression could not locate its data file, server message would inform the server that there had been a problem. The prototype version of the SCS makes no attempt to recover from errors, and the server merely informs the clients that the process has terminated unsuccessfully.

Succeeding clients receive a tag and client message of the form “continue,” decrypt the operation and payload, perform the indicated operation (e.g., “secure summation: add value”) to generate a new payload (e.g., the payload it received plus its $v_j$), encrypt the two with the clients-only key, and then proceed as described earlier.

The initiating client recognizes that the computation phase of the protocol is complete when it receives a second message from the server. Because it knows the protocol, it knows that it should then remove random initializers (e.g., subtract $R$ from $V + R$ to obtain $V$), set operation to “read,” set payload to $V$, and proceed.

The initiating client recognizes that the entire protocol is complete when it receives a third message from the server, in which case it sets server message to “protocol complete,” sends one final message to the server, and shuts itself down. The server then tells the remaining clients that the protocol is complete, which they acknowledge and shut themselves down.

Server-Side Processing. When it receives a message (purporting to be) from a client, the server first decrypts it using its private key. It then compares the tag to that of the last message that it sent out and, if the two are not identical, either ignores the message and waits for the “correct” message or declares a problem. If the tag is correct, then the server parses the server message, and acts accordingly. In most cases, the message will be “client operation successful” or “protocol complete.” As noted previously, in the current SCS, if the server message is “client encountered problem,” then the server terminates the protocol.

The server continues sending messages to clients in the same order (cyclically) until the server message is “protocol complete.” Note that the server is never able to read either the operation or the payload.

The client software for the SCS consists of a number of graphical user interfaces (GUIs) and computational engines. A full description of these will appear elsewhere (Vera, Fulp, and Karr 2007), but Figure 6 shows those associated with secure regression.

---

Figure 5. Encryption used by the SCS. (a) Encryption of server-to-client messages. (b) Encryption of client-to-server messages. Portions of messages encrypted with the clients-only key, which are shaded in gray, cannot be read by the server.
5.2 Secure EM Algorithms for Complex Data Partitions

For complex data partitions, such as that shown in Figure 7, one approach is for the database owners to base inferences solely on the data records that all owners have in common. One obvious problem is that there may be no such records, or too few to be useful. Also, there are security issues involved in determining which records are common across the databases. In any event, this approach sacrifices information from records common to some, but not all of the databases, which can result in inferences that are inefficient and potentially biased (Little and Rubin 2002).

An alternative approach is to view complicated data partitions as incomplete data sets—the global database is construed as a rectangular data set with missing values in those records not common to all parties—and then develop secure versions of techniques used for analyzing incomplete data sets. One such technique is to specify a joint distribution for the complete data and then use the EM algorithm (Dempster, Laird, and Rubin 1977) to estimate the parameters of that distribution. For some models (e.g., distributions from the exponential family), the EM algorithm requires only sufficient statistics. If those sufficient statistics can be calculated using SMPC, then a secure EM algorithm is feasible, as we now show.

For simplicity, assume that the data follow a multivariate normal distribution. One problem, of course, is to verify this assumption when the owners do not share data values. We further assume that the owners share globally unique identifiers of the records in their databases, to identify records that are common to multiple databases. These identifiers are shown in Figure 7. Finally, we assume that matching on these unique identifiers can be done without error.

For the multivariate normal distribution, the sufficient statistics are sums, sums of squares, and sums of cross-products of the data values. All of these can be computed securely by the following protocol.

Let $M$ be the number of incomplete data (“data missingness”) patterns in the global database $D$. For example, in Figure 7, $M = 5$; partitioning the attributes into four blocks (corresponding to owners 1, {2, 3}, 4, and 5), there are five patterns: blocks 3 and 4, block 3, blocks 2 and 3, block 4, and no blocks. For $m = 1, \ldots, M$, let $D_m$ be the set of all data elements with missingness pattern $m$.

To begin the secure EM protocol, the owners group data records by missingness patterns, which is possible because they have shared unique identifiers. After this initial cooperation, each owner knows the values of $m$ for all records and the values of the data for the records in its database.

The owners next compute and share two tables of summary statistics needed by the EM algorithm. The first table has $M$ rows corresponding to the missingness patterns and $p$ columns corresponding to (all of) the attributes in the global database. The entry in the table for row $m$ and column $j$ is the sum of the observed $y_{ij}$ for those records with the missingness pattern associated with row $m$. When there are no common attributes, each sum is computed by only one owner. When there are common attributes, the sum is computed using secure summation. In the case of common attributes and common records, the owners cooperate to ensure that each record is represented only once in each $\sum y_{ij}$.
The second table has \( M \) rows corresponding to the missingness patterns and \( p(p+1)/2 \) columns corresponding to the inner products of all \( p \) variables in the data set, including the sums of squares. The entry in the table for row \( m \) and the column associated with attributes \((j,k)\) is the \( \sum_{j|\not\in k} \) for those records with the missingness pattern of row \( m \). With no common attributes, each cross-product entry in the table is derived from a single dot product involving only two database owners. When there are common attributes, the owners cooperate to ensure that each record enters the \( \sum_{j|\not\in k} \) one time for each \((j,k)\). The table has many structural 0, because there are no dot products between the missing and observed data. The owners compute the dot products using a secure dot product protocol (Du and Zhan 2002; Sanil et al. 2004a), which allows owners to perform dot products without sharing attribute values.

Once each owner has the two tables of summary statistics, it has all of the information needed to run the EM algorithm independently of other owners. (For details of the E-steps and M-steps for a multivariate normal model, see Schafier 1997.) Further, inference from the data (e.g., fitting regression models) is then possible without additional error.

The secure EM protocol is vulnerable not only to the usual risk of the owners’ not being semi-honest, but also to confidentiality risks associated with sharing globally unique identifiers. In addition, missingness patterns associated with small numbers of attributes are problematic. To illustrate in the multivariate normal setting, for any missingness pattern with \( q \) attributes, there are \( q + q(q + 1)/2 \) equations involving the records in that pattern. When the number of records in that pattern is less than or equal to \( q + q(q + 1)/2 \), the owners can solve these equations for the data values associated with that pattern. To protect confidentiality, the owners may have to exclude missing-data patterns with small numbers of records from the EM, although this could bias parameter estimates. Finally, the secure EM protocol does not guard against risks arising when sensitive attributes owned by different owners are nearly colinear.

A possibly deeper difficulty is that EM algorithms are based on the assumption that the incomplete data are missing at random. But Figure 7 makes it clear that the missingness is structural, with attributes missing in blocks. Further research is needed to address this issue.

5.3 Partially Trusted Third Parties

Here we consider more carefully problematic aspects of the semi-honesty assumption in Section 2.1, and outline a path to mitigate them. In the secure regression setting of Section 3.2, suppose that all owners other than \( j \) are semi-honest, but that instead of contributing \( [X'y']^T [X'y'] \) to the summation in (5), owner \( j \) puts in 0. Then what every other owner receives at the end of the process and thinks is \( [X'y']^T [X'y] \) is in fact

\[
(Xy)^T [Xy] - j = \sum_{k \not\in j} [X'y]^T [X'y].
\]

(14)

Owner \( j \) can then merely add \( [X'y]^T [X'y] \) to obtain the correct \( [X'y]_j^T [X'y] \) and proceed to perform the correct regression, leaving other owners unaware that they do not have the correct regression.

Active deception can cause more severe problems. If instead of \( [X'y]^T [X'y] \) or 0, owner \( j \) adds false values \([X'y^f]^T [X'y^f] \), then what each other owner thinks is \( [X'y]^T [X'y] \) is now

\[
\sum_{k=1,k\neq j}^K [X'y]^T [X'y] + [X'y^f]^T [X'y^f].
\]

Owner \( j \) obtains the correct value of \( [X'y]^T [X'y] \) by subtracting \( [X'y^f]^T [X'y^f] \) and adding \( [X'y]^T [X'y^f] \), and then can calculate the correct regression. Unless \( [X'y^f]^T [X'y^f] \) is egregiously false, the other owners will be unaware that they have a completely bogus regression.

These examples show that the secure regression protocol in Section 3.2, viewed as a multiplayer game, is not a Nash equilibrium. Each owner has unilateral incentive to not be semi-honest, which is effective if the others are semi-honest.

The concept of PTTPs, currently under development at NISS, may be able to reduce unilateral incentives to cheat in situations where the results of interest are (nonadditive) functions of two or more other quantities. For secure regression, PTTPs work because \( \hat{\beta} \) is a function of \( X'yX \) and \( X'y \), which can be computed independently by secure summation. They also would work for secure averages such as (1), with the number and denominator computed using secure summation.

A PTTP is in effect a dataless owner that initializes secure summations, receives the results, calculates quantities of interest, and shares only the final result of the computations with the database owners. To illustrate in the setting of secure regression, we restrict attention to regression coefficients \( \hat{\beta} \). The PTTP protocol is then as follows:

- **Initialization.** The PTTP creates a matrix \( R \) of dimensions \( (p+1) \times (p+1) \), where \( p \) is the number of predictors, each of which is a very large random number.
- **Iteration.** Using either an ordinary secure summation protocol (Sec. 2.2) or the SCS of Section 5.1, the database owners each add their \( [X'y]^T [X'y] \).
- **Computation.** When it receives back \( R + [X'y]^T [X'y] \) from the final owner to contribute, the PTTP subtracts \( R \) and then calculates \( \hat{\beta} \) using (4).
- **Dissemination.** The PTTP disseminates \( \hat{\beta} \) to the owners. It does not disseminate \( [X'y]^T [X'y] \).

The major advantage of the PTTP protocol is that because the computation of \( \hat{\beta} \) is hidden from the database owners, it is harder to undo the effects of cheating. Let \( \hat{\beta}_{\not\in j} \) denote the (true) coefficients for the regression involving all owners other than \( j \). If instead of adding \( [X'y]^T [X'y] \), owner \( j \) adds \( [X'y^f]^T [X'y^f] \) and then receives false coefficients \( \hat{\beta}_{\not\in j} \), to recover true coefficients, \( j \) must somehow remove the effects of \( [X'y^f]^T [X'y^f] \) from \( \hat{\beta}_{\not\in j} \) and then “add” the effects of \( [X'y]^T [X'y] \).
invertible \( p \times p \) matrix not known to the PTTP, and the regression could be performed using the PTTP and the \( Z' \). Finally, each owner would multiply the \( \beta \) received from the PTTP by \( B \) to recover the true estimated coefficients.

6. CONCLUSIONS AND DISCUSSION

In this article we have outlined an approach to valid statistical analysis of distributed data that does not require actually integrating the data. Instead, it is based on anonymized sharing of database-specific sufficient statistics in such a way that no database owner can disentangle the individual contributions of the other owners. We have presented underlying abstractions of SMPC, as well as illustrative protocols for regression, contingency tables, and exponential family maximum likelihood. We have described a prototype software system together with initial approaches to complex data partitioning and reducing incentives to “cheat.”

Many research challenges remain, several of which have been noted here. One of the most central of these is to link our approach to traditional concerns in statistical disclosure limitation (Willenborg and de Waal 2001), which focus on protecting the identities of individual data records and sensitive attributes within them.

ACKNOWLEDGMENTS

This research was supported by National Science Foundation grants EIA-0131884 and SES-0345441 to the National Institute of Statistical Sciences and DMS-0112069 to the Statistical and Applied Mathematical Sciences Institute. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors thank the referees and guest editor for many valuable comments on the manuscript.

[Received December 2005. Revised August 2006.]

REFERENCES


This article has been cited by:

