NISS

Secure, Privacy-Preserving Analysis of Distributed Databases

Alan F. Karr, William J. Fulp, Francisco Vera,
S. Stanley Young, Xiaodong Lin and Jerome P. Reiter

Technical Report Number 158
December 2005

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

Secure, Privacy-Preserving Analysis of Distributed Databases

Alan F. Karr, William J. Fulp, Francisco Vera, S. Stanley Young
National Institute of Statistical Sciences
Research Triangle Park, NC 27709–4006, USA
karr@niss.org, wfulp@stat.cmu.edu, veraf@niss.org, young@niss.org

Xiaodong Lin
University of Cincinnati
Cincinnati, OH
linxd@math.uc.edu

Jerome P. Reiter
Duke University
Durham, NC 27708 USA
jerry@stat.duke.edu

December 1, 2005

Abstract

There is clear value, in both industrial and government settings, derived from performing statistical analyses that, in effect, integrate data in multiple, distributed databases. However, the barriers to actually integrating the data can be substantial or even insurmountable. Corporations may be unwilling to share proprietary databases such as chemical databases held by pharmaceutical manufacturers, government agencies are subject to laws protecting confidentiality of data subjects, and even the sheer volume of the data may preclude actual data integration.

In this paper, we show how tools from modern information technology—specifically, secure multiparty computation and networking—can be used to perform statistically valid analyses of distributed databases. The common characteristic of the methods we describe is that the owners share sufficient statistics computed on the local databases in a way that protects each owner from the others. That is, while each owner can calculate the "complement" of its contribution to the analysis, it cannot discern which other owners contributed what to that complement.

Our focus is on horizontally partitioned data: the data records rather than the data attributes are spread among the owners. We present protocols for secure regression, contingency tables, maximum likelihood and Bayesian analysis. For low-risk situations, we describe a secure data integration protocol that integrates the databases but prevents owners from learning the source of data records other than their own. Finally, we outline three current research directions: a software system implementing the protocols, secure EM algorithms, and partially trusted third parties, which reduce incentives to owners not to be honest.

1 Introduction

Many scientific or business investigations require statistical analyses that "integrate" data stored in multiple, distributed databases. At the same time, the barriers to actually integrating the databases are numerous. In this paper, we describe 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 multi-party computation*, the participating database owners can share summaries (in many cases, sufficient statistics) of the data anonymously, but in a way that the analysis can be performed in a statistically valid manner.

To illustrate, as in the example in §3.2, a regression analysis on integrated chemical databases to identify molecular features influencing biological activity is more insightful than individual analyses. In this setting, proprietary data are the principal impediment to integration. Scale is another barrier: despite advances in networking technology, the sure only way to move a terabyte of data from point A today to point B tomorrow may be FedEx.

By contrast, in the "official statistics" setting of Karr et al. (2004), Karr et al. (2005c), Sanil et al. (2004a) and Sanil et al. (2004b), and the homeland security setting of Karr et al. (2005b), where database owners may be states or multiple federal agencies, confidentiality of data subjects is paramount, and laws prohibit moving or sharing data.

The paper is organized in the following manner. §2 introduces the computer science abstraction of secure multi-party computation (SMPC) that underlies our methods, and presents the one concrete version—secure summation—that we require. §3 presents a number of analyses for "horizontally partitioned data" (defined in §1.2), including data integration, regression, contingency tables, maximum likelihood for exponential families and Bayesian analyses. In §4 we describe three directions of ongoing research at the National Institute of Statistical Sciences (NISS). Conclusions appear in §5.

1.1 Problem Formulation

Consider a "global" database that is partitioned among a number of "owners." For concreteness, the owners can be thought of as companies (as in the example in §3.2) or, in official statistics contexts, as 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 are mentioned above: they range from protecting proprietary data to scale of the data to confidentiality. In particular, the "without ever actually creating [the global database]" proviso precludes use of either human or electronic trusted third parties. "[S]tatistically valid" means just that: the same answer is obtained as would have been obtained from the global database. It also means that all objects statisticians customarily think of as part of the analysis must be calculated. To illustrate, for a regression (§3.2), not only the estimated coefficients but also their standard errors, measures of model fit and even characteristics of residuals must be provided.

The owners wish protection from one another in the sense that while each owner can compare the results of the global analysis to, for instance, the results of performing the same analysis on its own data, it should not be able to attribute any characteristics of the difference to specific other database owners. Whether the analysis places data subjects at risk is not yet understood (see §5).

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

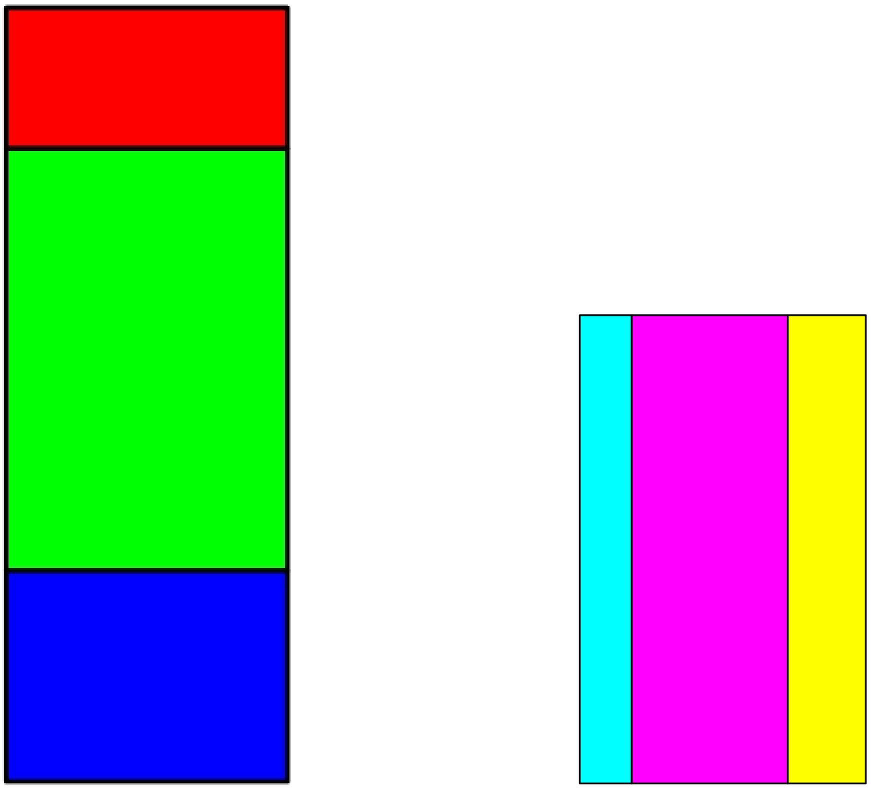


Figure 1: Principal models for data partitioning. Data subjects are rows, and attributes are columns. *Left:* horizontally partitioned data—data subjects are partitioned among database owners. *Right:* Vertically partitioned data—attributes are partitioned among database owners.

1.2 Data Partitioning Models

For the remainder of this paper, "database" means a flat file of rows representing data subjects $\times$ columns representing attributes. There are two "structured" data partitioning models. For *horizontally partitioned data*, on which this paper focuses, the data subjects are partitioned among the database owners, and each has the same attributes for all of its subjects. This is illustrated in the left-hand panel of Figure 1. For *vertically partitioned data*, it is the attributes rather than the subjects that are partitioned among the database owners, as in the center panel in Figure 1. Regression for vertically partitioned data is treated in Sanil et al. (2004a) and Sanil et al. (2004b). Finally, of course, there are more complex partitionings, as discussed and illustrated in in §4.2.

Both structured 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 need to have the attributes in the same order and in the same units. For vertically partitioned data, there is the much more difficult problem of linking records across the databases.

2 Secure Multi-Party Computation

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

2.1 Generalities

Consider K data owners (parties, in the computer science literature) with values $v_1, \ldots, v_K$ who wish to evaluate a known function f at these values subject to three constraints:

C1: The correct value $f(v_1, \ldots, v_K)$ is obtained and known to all owners.

C2: No owner j knows more about the other owners' values $\mathcal{V}_{-j} = \{v_k : k \neq j\}$ than it can deduce from v_j and $f(v_1, \ldots, v_K)$.

C3: No trusted third party—human or machine—is part of the process.

The challenge is that **C1** and **C2** say that the process must be as effective as if there were a trusted third party, while **C3** forbids this.

There is also an implicit fourth constraint that no owner can "disentangle" the other owners' contributions to $f(v_1, \ldots, v_K)$. That is, while owner j may be able to learn something about $\mathcal{V}_{-j}$, it should be ignorant about the nature of others' contributions to $f(v_1, \ldots, v_K)$. In most cases, as for secure summation (§2.2), this constraint is satisfied automatically because f is symmetric in its K arguments.

The computer science literature contains a large number of papers on the theory of SMPC. Fewer of these, it seems, describe implemented algorithms, let alone functioning software systems. Some procedures for SMPC involve encryption, while others depend on some form of randomization. The latter are exemplified by secure summation.

As an example of the former, suppose two owners have databases of individuals indexed by social security numbers and they want to determine those individuals who are in both databases. Then one owner can form a polynomial with roots at (and only at) the social security numbers in its database, which it then sends to the second owner in "open"—the values of the coefficients—rather than factored form. Because of the computational (near-)impossibility of factoring the polynomial, the second owner cannot recover the zeros, so the first owner's data are in effect encrypted. The second owner can, however, evaluate the polynomial at the social security numbers in its database, and those for which the polynomial evaluates to zero are in both databases. To complete the process, it informs the first owner of the results.[1]

Nearly all protocols for SMPC assume that the owners are *semi-honest*. Specifically, this means that they perform agreed-upon computations correctly, and that they use their true data. If the protocol is iterative, they are permitted, however, to retain the results of intermediate computations. In §4.3, we examine the consequences of owners that are not semi-honest, and propose one way to mitigate them.

2.2 Secure Summation

In this paper we use only the simplest version of SMPC, namely secure summation. In the notation of §2.1,

$$f(v_1, \ldots, v_K) = v_1 + \ldots + v_K.$$

[1]This example is not realistic in the sense that there are only 10^{10} social security numbers, so owner 2 could just evaluate the polynomial at all social security numbers and learn exactly who is in owner 1's database. There are ways around this problem, but they obscure the essence of the process.

Denoting this sum by V and $\sum_{k \neq j} v_k$ by V_{-j}, some of the generalities in §2.1 become more concrete. Were there a trusted third party, owner j would know *only* v_j and V, from which it can calculate V_{-j}. 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.

The secure summation protocol (Benaloh, 1987), which is depicted graphically in Figure 2, is straightforward in principle. Assume for simplicity that the v_k are integers.

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. Since 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 $R + v_1 + \ldots + v_K = R + V$ from owner K, subtracts R, and shares the result V with the other owners.

Figure 2 contains 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}, that is known to all the owners. Then R can be chosen randomly from $\{0, \ldots, m-1\}$ and all computations performed modulo m.

As a simple but illustrative application, 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 I_j be the sum of their incomes. The quantity to be computed is

$$\bar{I} = \frac{\sum_j I_j}{\sum_j n_j}, \tag{1}$$

whose numerator can be computed using secure summation on the I_j's, and whose denominator can be computed using secure summation on the n_j's.

Despite the simplicity of the protocol, a production quality implementation of secure summation 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 outsiders. The Secure Computation System (SCS) described in §4.1 implements these protections and more. In particular, the SCS hides from the owners the order in which they contribute their v_k's, which prevents collusion. If the order is known, then the owners before and after j, by comparing the values one sends to j and the other receives from j, can determine v_j.

3 Analysis of Horizontally Partitioned Data

In this section, we describe a variety of analyses for horizontally partitioned data.

3.1 Secure Data Integration

The problem formulation in §1.1 prohibits creation of the global database as a means of solving the problem. There may be, of course, "low-risk" situations—see §3.3 for one—in which the owners are willing to create and share the global database, but still wish to protect the sources of data elements. For example, owners who are retailers may be willing to share information about their customers as long as they don't reveal who is whose customer. Of course from the global and its local database, each owner can recognize which customers are not its own, but it should not be able to infer which other owner's customers they are.

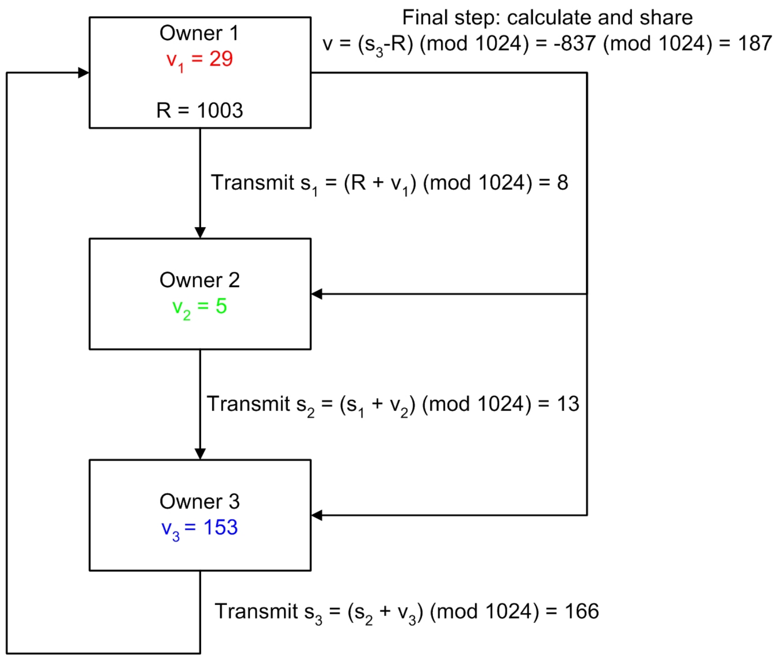

Owner 1
v_1 = 29
R = 1003
Final step: calculate and share
v = (s_3-R) (mod 1024) = -837 (mod 1024) = 187
Transmit s_1 = (R + v_1) (mod 1024) = 8
Owner 2
v_2 = 5
Transmit s_2 = (s_1 + v_2) (mod 1024) = 13
Owner 3
v_3 = 153
Transmit s_3 = (s_2 + v_3) (mod 1024) = 166

Figure 2: Pictorial representation of secure summation.

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, for example, if the owners were state education agencies in the US 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" of the NISS Secure Computation System described in §4.1 is employed, so that owners are not aware of the order in which they are contributing data elements. More specifically, the owners incrementally contribute data in a random round-robin order known to a central server to not to them. Here are details, with the owners numbered in that random order:

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

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

Penultimate round. Each owner removes own previous data and records other owners' previous data as for **Intermediate Rounds**, but now puts in *all its remaining data*. Which round is the penultimate round may be owner-dependent.

Final round. Following its penultimate round, each owner removes the data it inserted then, 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 an owner receives the database *all other owners* will have contributed to it (at least those who have not exhausted their own data), so there is no means of attributing new records to other owners. The "remove what you put in on the previous round" component of the protocol controls the size of the database in order to achieve computational feasibility.

The randomizations required by this protocol matter. As a standard for judging them, suppose that the database owners knew the sizes of each other's databases,[2] so that the naive prior distribution on the source of a record is the size of that database divided by the total of the sizes. In reality, of course, each owner would attempt to attribute only records other than its own, and would remove the size of its database from the denominator. So finally, the extent to which observing the protocol[3] enables an owner to improve its estimates—which now are posterior distributions given what is observed about the protocol—over the naive ones measures the lack of protection of sources.

In early rounds, randomizations that add a constant fraction of each owner's records are revealing, so owners must be putting in the same expected numbers of records. But if "same expected numbers of records" were maintained as the process proceeds, then in late rounds only owners with large databases would be contributing. The precise nature of this tradeoff is still being worked out.

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

$$y = X\beta + \epsilon, \tag{2}$$

to the "global" data

$$X = \begin{bmatrix} X^1 \\ \vdots \\ X^K \end{bmatrix} \qquad \text{and} \qquad y = \begin{bmatrix} y^1 \\ \vdots \\ y^K \end{bmatrix}.$$

We embed the constant term of the regression in the first predictor by putting $X_1^j \equiv 1$ for all j. To illustrate the subtleties of analysis of distributed data, the usual strategy of centering the predictors and response at mean values does not work, at least not directly. The means in this case are the global means, which are not available, although they could be calculated with another round of secure computation.

Under the condition that $\mathrm{Cov}(\varepsilon) = \sigma^2 I$, the least squares estimator for β is of course

$$\hat{\beta} = (X^T X)^{-1} X^T y. \tag{3}$$

Let $n = \sum n_j$ be the size of the global database. Then the crucial point is that the global $(p+1) \times (p+1)$ matrix

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

[2]If they were state agencies, this would be a plausible assumption, since the sizes would be likely to be public information.

[3]Recall that semi-honesty allows retention of intermediate results.

from which $\hat{\beta}$ can be calculated using (3), is additive over the owners:

$$[X\ y]^T[X\ y] = \sum_{k=1}^{K} [X^k\ y^k]^T[X^k\ y^k]. \tag{4}$$

Figure 3 illustrates this for $X^T X$: each X^j is $n_j \times p$.

Therefore, $[X\ y]^T[X\ y]$ can be computed *entrywise* using secure summation, and each owner can then calculate $\hat{\beta}$ using (3).

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

Figure 4 summarizes the results. The first three panels 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 zero 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, at least if the companies knew the sizes of each others' databases,[4] is whether companies 1, 2 and 4 should allow company 3 to participate, since it has so little data. The lower right-hand panel of Figure 4 sheds some light on this. There, the x-axis again contains the coefficients for the four-company regression and the y-axis those for the regression for companies 1, 2 and 4, excluding company 3. Though close, the coefficients are not identical, so companies 1, 2 and 4 do gain from including company 3.

From a statistical perspective, calculation of $\hat{\beta}$ is only part of "performing" a valid, useful regression. A variety of other objects can be calculated from $[X\ y]^T[X\ y]$, or using secure summation directly. These include the coefficient of determination R^2, the least squares estimate S^2 of the error variance σ^2 and the "hat" matrix $H = X(X^T X)^{-1} X^T$, which can be used to identify outliers (Karr et al., 2005b,c). It is also possible to use the secure data integration algorithm of §3.1, together with methods for constructing (privacy-preserving) synthetic residuals in ordinary regressions (Reiter, 2005), to create secure synthetic residuals (Karr et al., 2005b).

3.3 Secure Contingency Tables

The algorithm for secure data integration described in §3.1 has an important indirect application—securely constructing contingency tables containing counts or sums.

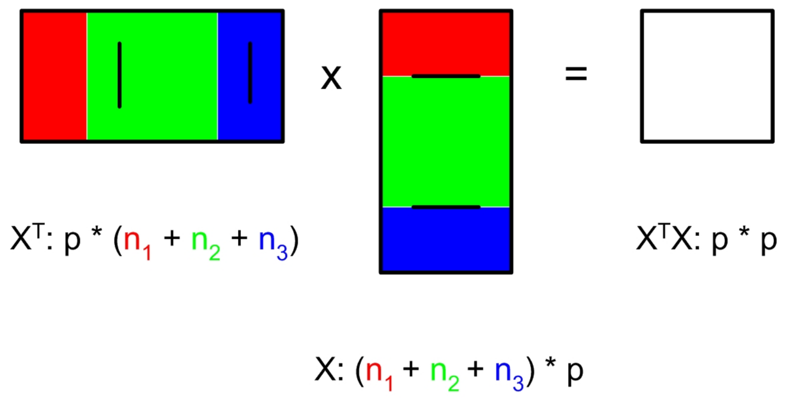
Let $\mathcal{D}$ be a database containing only categorical attributes $A_1, \ldots, A_J$. The associated contingency table is the J-dimensional array T defined by

$$T(a_1, \ldots, a_J) = \#\{r \in \mathcal{D} : r_1 = a_1, \ldots, r_J = A_J\}, \tag{5}$$

where each a_i is a possible value of the categorical attribute A_i[5], $\#\{\cdot\}$ denotes "cardinality of $\cdot$" and r_i is the

[4]Which our protocol does not require.

[5]For example, if A_1 corresponds to gender, then possible values of a_1 are "female" and "male."



X^T: p * (n_1 + n_2 + n_3)
x
=
X^TX: p * p
X: (n_1 + n_2 + n_3) * p

Figure 3: Pictorial representation of the computation of the global $X^T X$ for secure regression.

ith attribute of record i. 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. The table T is a near-universal sufficient statistic, for example, for fitting log-linear models (Bishop et al., 1975).

While (5) defines a table as an array, this is not a feasible data structure for tables very large numbers of cells. Fortunately large tables are invariably sparse, with relatively few cells having non-zero counts. For example, the table associated with the US Census "long form," which contains 52 questions, has more than 10^{15} cells (1 gigabyte = 10^9) but at most approximately 10^8 (the number of households in the US) of these are non-zero. The *sparse representation* of a table is the data structure of (cell coordinate, cell count) pairs

$$\Big\{\big(a_1, \ldots, a_J, T(a_1, \ldots, a_J)\big) : T(a_1, \ldots, a_J) \neq 0\Big\}.$$

Algorithms that use the sparse representation data structure have been developed for virtually all important table operations.

Consider now the problem of securely building a contingency table from databases $\mathcal{D}_1, \ldots, \mathcal{D}_K$ containing the same categorical attributes for disjoint sets of data subjects. Given the tools described in §3.1 and 2.2, this process is straightforward. The steps:

1. **List of Non-Zero Cells:** Use secure data integration to build the list $\mathcal{L}$ of cells with non-zero counts. The "databases" being integrated in this case are the owners' individual lists of cells with non-zero counts. The protocol in §3.1 allows each owner not to reveal in which cells it has data in.

2. **Non-Zero Cell Counts:** For each cell in $\mathcal{L}$, use secure summation to determine the associated count (or sum).

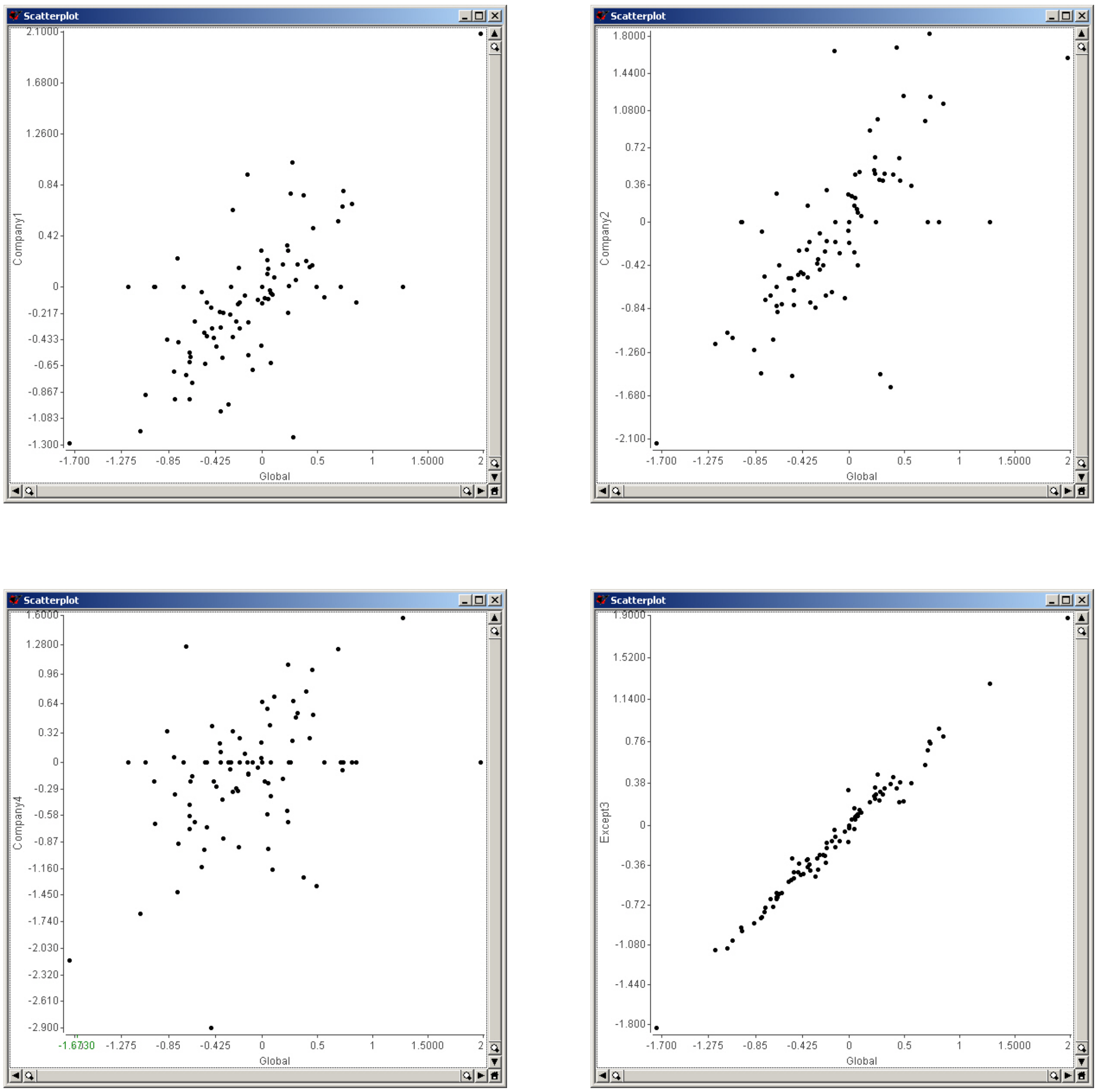


Figure 4: Scatterplots for the example discussed in §3.2. In all plots, the regression coefficients for the four-company regression appear on the x-axis. *Upper left:* y-axis contains regression coefficients for company 1 alone. *Upper right:* y-axis contains regression coefficients for company 2 alone. *Lower left:* y-axis contains regression coefficients for company 4 alone. *Lower right:* y-axis contains regression coefficients for the regression for companies 1, 2 and 4, but excluding company 3.

3.4 Secure Maximum Likelihood and Bayes

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

$$\log f(\theta, x) = \sum_{\ell=1}^{L} c_\ell(x) d_\ell(\theta). \tag{6}$$

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

$$\log L(\theta, x) = \sum_{\ell=1}^{L} d_\ell(\theta) \left[\sum_{k=1}^{K} \sum_{x_i \in \mathcal{D}_k} c_\ell(x_i) \right], \tag{7}$$

where $\mathcal{D}_k$ is the database of owner k.

Assuming that the owners have agreed in advance on the model (6), they can simply use secure summation to compute each of the L terms in (7), and then each can maximize the likelihood function by whatever means it wishes.

Any owner that wishes can also perform Bayesian analyses, with whatever prior distribution π it chooses, since the relevant posterior distribution is computable from π and the likelihood function $L(\theta, x)$.

4 New Directions

In this section we describe three directions of ongoing research at NISS.

4.1 The NISS Secure Computation System

As alluded to in §2.2, a serious implementation of protocols such as that for secure regression (§3.2) presents significant challenges. A prototype Secure Computation System (SCS) developed by NISS attempts to address many of these.

The fundamental difference between the SCS and the simple protocols in §3 is that the SCS uses a network-based server-client model implemented as a star topology, illustrated in Figure 5, which operates on the public internet. The database owners are the clients, but they never communicate directly with each other. Rather, they communicate only with the server. There are multiple reasons for this. First, the server can hide from the clients the order in which they provide information, preventing collusion. Second, authentication and encryption are much more efficient with the server. And finally, process management is vastly simplified.

On the other hand, since the server is not an owner, it must not be able to "read" any of the information it is passing between clients, which leads, as described below, to a dual encryption. In fact, in the SCS the server is not even aware which protocol the clients are performing.

Here is 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 §3, this means everything except secure data integration and the first stage of secure contingency tables.

Set-Up. Prior to actually performing the analysis, the owners/clients must agree on who is participating, which protocol will be performed, 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 not-yet-implemented reservation system.

Log In. At the reserved time, the server listens for clients to log in. As each client logs in, it receives from the server 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 server the latter.

Initiation. When all clients, as determined by the reservation, have logged in, the server randomly selects an order for the clients, and sends "Initiate protocol" message to the first client.

The structure and encryption of messages from the server to the client is shown in the top panel in Figure 6. The entire message is encrypted using the client's public key, which prevents its 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 below, 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, unbeknowst to the server, to perform secure summation, 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, 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 clients would be informed by the server that the process had terminated unsuccessfully.

Succeeding clients receive a Tag and Client Message of the form "Continue," decrypt the Operation and Payload, perform the indicated operation (Example: "Secure summation: add value") to generate a new payload (Example: the payload it received plus its v_j), encrypt the two with the clients-only key, and then proceed as described above.

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 (Example: 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 it sent out, and if the

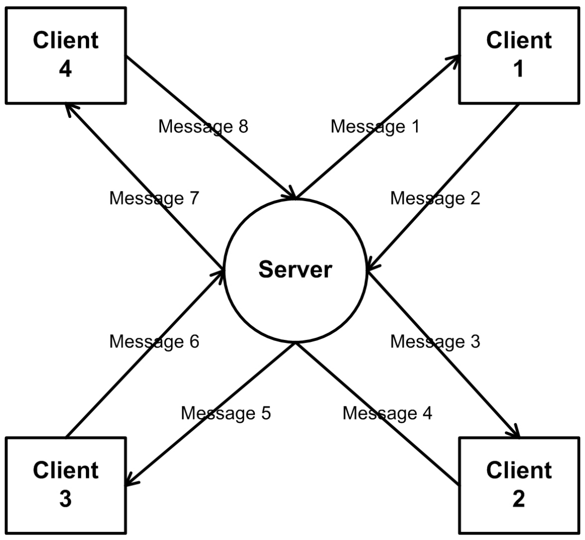


Figure 5: Star topology structure of the SCS illustrated with four clients. Messages are never passed directly from one client to another but rather only from the server to a client, or *vice versa*.

two are not identical, either ignores the message and waits for the "correct" message, or declares a problem. If the Tag is correct, the server parses the Server Message, and acts accordingly. In most cases, the message would be "Client operation successful" or "Protocol complete." As noted previously, in the current SCS, if the Server message is "Client encountered problem," the server would terminate the protocol.

The server continues sending messages to clients in the same order (cyclically) until Server Message is "Protocol complete." Note that the server is never able to read either the Operation or the Payload.

Table 1 illustrates the entire process for the secure summation example in Figure 2.

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 et al., 2005), but Figure 7 shows those associated with secure regression.

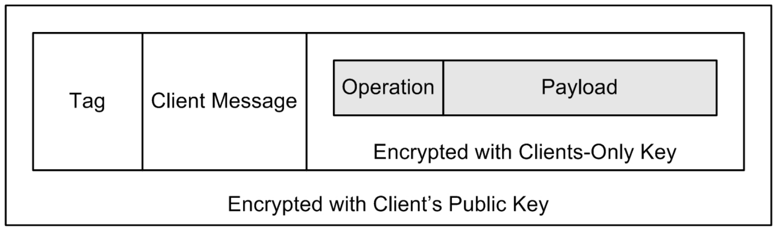
4.2 Secure EM for Complex Data Partitions

For complex data partitions such as that in Figure 8, 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. And 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 to develop secure versions of techniques commonly used for analyzing incomplete data sets. To analyze incomplete data, a typical strategy is to specify a joint distribution for the complete data, and then to use the EM algorithm (Dempster et al., 1977) to estimate the parameters of that distribution. Doing so is complicated in our setting because the database owners do not share individual data values. However,

Sender→ Recipient	Tag	Client or Server Message	Operation	Payload
Server → Client1	Tag1	Initiate Protocol	∅	∅
Client1 → Server	Tag1	Success	Secure sum: add value	8
Server → Client2	Tag2	Proceed	Secure sum: add value	8
Client2 → Server	Tag2	Success	Secure sum: add value	13
Server → Client3	Tag3	Proceed	Secure sum: add value	13
Client3 → Server	Tag3	Success	Secure sum: add value	166
Server → Client1	Tag4	Proceed	Secure sum: add value	166
Client1 recognizes that all clients have contributed				
Client1 → Server	Tag4	Success	Secure sum: read sum	187
Server → Client2	Tag5	Proceed	Secure sum: read sum	187
Client2 → Server	Tag5	Success	Secure sum: read sum	187
Server → Client3	Tag6	Proceed	Secure sum: read sum	187
Client3 → Server	Tag6	Success	Secure sum: read sum	187
Server → Client1	Tag7	Proceed	Secure sum: read sum	187
Client1 recognizes that all clients have read sum				
Client1 → Server	Tag8	Protocol complete	∅	∅
Client1 shuts down				
Server → Client2	Tag9	Protocol complete	∅	∅
Client2 → Server	Tag9	Completion acknowledged	∅	∅
Client2 shuts down				
Server → Client3	Tag10	Continue	∅	∅
Client3 → Server	Tag10	Completion acknowledged	∅	∅
Client3 shuts down				
Server recognizes that all clients have shut down, and shuts itself down				

Table 1: SCS implementation of secure summation for the example in Figure 2, assuming that the clients are numbered 1, 2, 3. Neither Operation nor Payload is visible to the server.


Tag
Client Message
Operation
Payload
Encrypted with Clients-Only Key
Encrypted with Client's Public Key

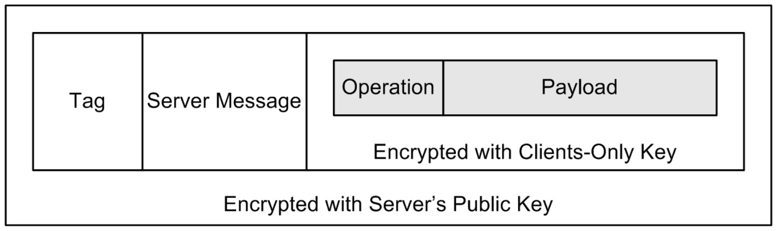

Tag
Server Message
Operation
Payload
Encrypted with Clients-Only Key
Encrypted with Server's Public Key

Figure 6: Encryption used by the SCS. *Top:* Encryption of server-to-client messages. *Bottom:* 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.

for some models—for example 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 illustrate.

For simplicity, assume that the data follow a multivariate normal distribution. One problem, of course, is that this assumption is difficult to verify when the owners do not share data values.[6] We further assume that the owners share *globally unique* identifiers (for example, social security numbers) of the records in their databases, which is necessary in order to identify records that are common to multiple data sets. These identifiers are shown in Figure 8. Finally, we assume that matching on these unique identifiers can be done without error.

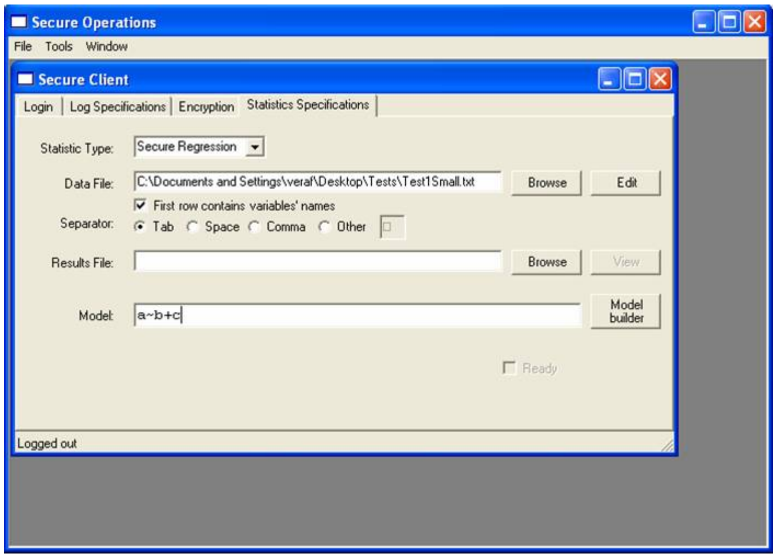
For the multivariate normal distribution, the sufficient statistics are sums, sums of squares, and sums of cross-products of the data values. All of these can be computed securely by the following protocol.

Let M be the number of incomplete data ("data missingness") patterns in the global database $\mathcal{D}$. For example, in Figure 8, $M = 5$: partitioning the attributes into four blocks, the patterns are as shown in Table 2. For $m = 1, \ldots, M$, let $\mathcal{D}_m$ 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 since 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) the attributes in the global database. The entry in the table for row m and column j is the sum of the observed y_j

[6]For regressions (§3.2), only the distribution of the errors in the regression need be normally distributed, which each owner can check. NISS is researching more flexible methods of secure computation.



Statistic Type: Secure Regression

Data File: C:\Documents and Settings\veraf\Desktop\Tests\Test1Small.txt

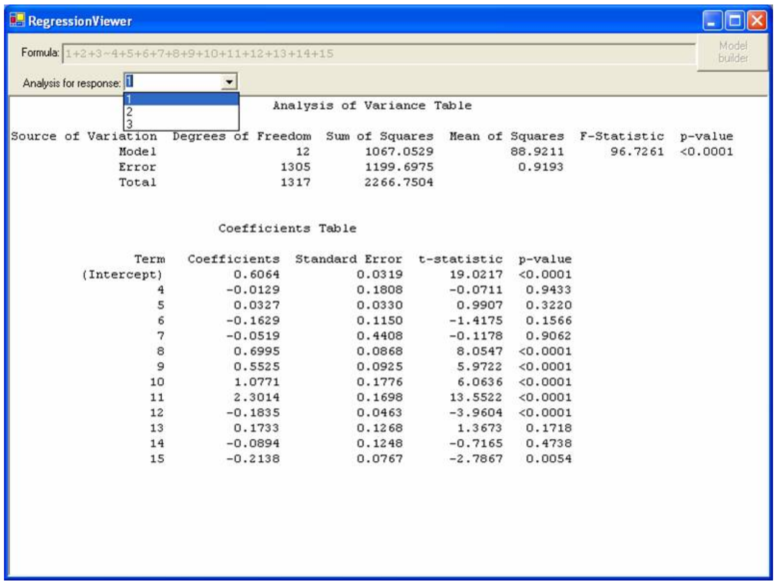
[✓ First row contains variables' names]

Separator: [✓ Tab] Space Comma Other

Results File:

Model: a~b+c

Ready



Formula: 1+2+3~4+5+6+7+8+9+10+11+12+13+14+15

Analysis for response: 1

```
                                Analysis of Variance Table

Source of Variation  Degrees of Freedom  Sum of Squares  Mean of Squares  F-Statistic  p-value
              Model                  12       1067.0529          88.9211      96.7261  <0.0001
              Error                1305       1199.6975           0.9193
              Total                1317       2266.7504


                          Coefficients Table

               Term  Coefficients  Standard Error  t-statistic  p-value
        (Intercept)        0.6064          0.0319      19.0217  <0.0001
                  4       -0.0129          0.1808      -0.0711   0.9433
                  5        0.0327          0.0330       0.9907   0.3220
                  6       -0.1629          0.1150      -1.4175   0.1566
                  7       -0.0519          0.4408      -0.1178   0.9062
                  8        0.6995          0.0868       8.0547  <0.0001
                  9        0.5525          0.0925       5.9722  <0.0001
                 10        1.0771          0.1776       6.0636  <0.0001
                 11        2.3014          0.1698      13.5522  <0.0001
                 12       -0.1835          0.0463      -3.9604  <0.0001
                 13        0.1733          0.1268       1.3673   0.1718
                 14       -0.0894          0.1248      -0.7165   0.4738
                 15       -0.2138          0.0767      -2.7867   0.0054
```

Figure 7: Selected components of the GUI for the client-side SCS software. *Top:* GUI to specify secure regression protocol, location and field separators for data file, location for file containing results, and model. *Bottom:* GUI showing output from secure regression.

	Attribute Blocks Present			
Pattern Number	1	2	3	4
1	X	X		
2	X	X		X
3	X			X
4	X	X	X	X
5	X	X	X	

Table 2: Patterns of "data missingness" corresponding to the database in Figure 8.

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 $\sum y_j$.

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 $\sum y_j y_k$ 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 $\sum y_j y_k$ one time for each (j,k). The table has many structural zeros, 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 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 Schafer (1997). Further inference from the data, for example, 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 (although see §4.3) but also to confidentiality risks associated with sharing of 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 co-linear.

A possibly deeper difficulty is that EM algorithms are based on the assumption that the incomplete data are missing at random. Figure 8, however, makes clear that the missingness may be "structural," with attributes missing in blocks. Further research is necessary to address this issue.

It is clear, in any case, that the secure EM protocol does work when the data are partitioned horizontally (§1.2) and the missing values truly are missing at random.

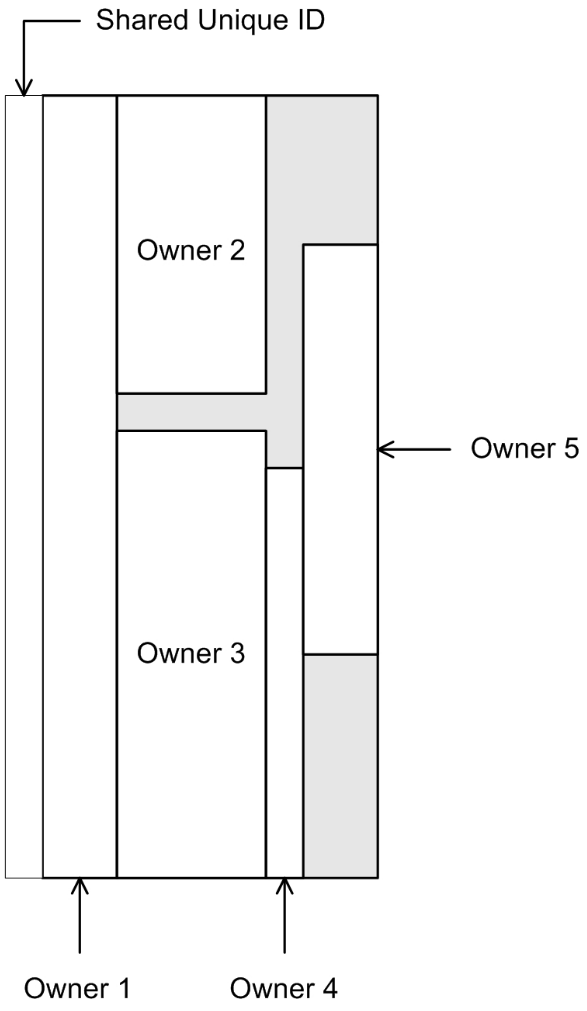


Figure 8: Example of complex data partitioning with five database owners.

4.3 Partially Trusted Third Parties

Here we consider more carefully the semi-honesty assumption introduced in §2.1.

In the secure regression setting on §3.2, let us consider two problematic scenarios. First, suppose that all owners other than j are semi-honest, but that instead of contributing $[X^j\ y^j]^T[X^j\ y^j]$ to the summation in (4), 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

$$\big([X\ y]^T[X\ y]\big)_{-j} = \sum_{k\neq j}[X^k\ y^k]^T[X^k\ y^k]. \tag{8}$$

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

But, active lying is even more effective. If instead of $[X^j\ y^j]^T[X^j\ j^j]$ or 0, owner j adds *false* values $[X^j_{\text{false}}\ y^j_{\text{false}}]^T[X^j_{\text{false}}\ y^j_{\text{false}}]$, then what each other owner thinks is $[X\ y]^T[X\ y]$ is now

$$\sum_{k=1,k\neq j}^{K}[X^k\ y^k]^T[X^k\ y^k]+[X^j_{\text{false}}\ y^j_{\text{false}}]^T[X^j_{\text{false}}\ y^j_{\text{false}}].$$

Owner j obtains the correct value of $[X\ y]^T[X\ y]$ by subtracting $[X^j_{\text{false}}\ y^j_{\text{false}}]^T[X^j_{\text{false}}\ y^j_{\text{false}}]$ and adding $[X^j\ y^j]^T[X^j\ y^j]$, and then can calculate the correct regression. Unless $[X^j_{\text{false}}\ y^j_{\text{false}}]^T[X^j_{\text{false}}\ y^j_{\text{false}}]$ is egregiously false, the other owners will be unaware that they have a completely bogus regression.

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

The concept of *partially trusted third parties (PTTPs)*, currently under development at NISS, may be able to reduce unilateral incentives to cheat in situations where the results of interest are functions of two or more other quantities. For secure regression, PTTPs work because $\hat{\beta}$ is a function of X^TX and X^Ty. They would work for secure averages such as (1), but not secure summation, nor would they work for secure likelihood (§3.4), although they could work for secure Bayes.

A PTTP is, in effect, a dataless owner that initializes secure summations, receives the result, calculates quantities of interest, and shares (only) the result of the computation with the real 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 whose entries is a very large random numbers.

Iteration. Using either an ordinary secure summation protocol (§2.2) or the SCS of §4.1, the database owners each add their $[X^j\ y^j]^T[X^j\ y^j]$.

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 (3).

Dissemination. The PTTP disseminates $\hat{\beta}$ to the owners. *It does not disseminate* $[X\ y]^T[X\ y]$.

So does PTTP work? There is, of course, one significant drawback: the PTTP may end up knowing more than the database owners. For secure regression, the PTTP knows $[X\ y]^T[X\ y]$, while the owners know only $\hat{\beta}$. This may be acceptable in some settings, but will not be in others. There are also ways around it. For instance, for secure regression, the owners could each replace their X^j by $Z^j = X^jB$, where B is an invertible $p\times p$ matrix not known to the PTTP, and the regression could be performed using the PTTP and the Z^j. Finally, each owner would multiply the $\hat{\beta}$ received from the PTTP by B^{-1} to recover the true estimated coefficients.

The major advantage of the PTTP protocol is that it is much harder to undo the effects of cheating. Let $\hat{\beta}_{-j}$ denote the (true) coefficients for the regression involving all owners other than j. If instead of adding $[X^j\ y^j]^T[X^j\ y^j]$, owner j adds $[X^j_{\text{false}}\ y^j_{\text{false}}]^T[X^j_{\text{false}}\ y^j_{\text{false}}]$, and then receives false coefficients $\hat{\beta}_{\text{false}}$, in order to recover true coefficients, j must somehow remove the effects of $[X^j_{\text{false}}\ y^j_{\text{false}}]^T[X^j_{\text{false}}\ y^j_{\text{false}}]$ from $\hat{\beta}_{\text{false}}$, and then "add" the effects of $[X^j\ y^j]^T[X^j\ y^j]$.

Even adding zero instead of $[X^j\ y^j]^T[X^j\ y^j]$ is problematic, although in this case owner j would know that the $\hat{\beta}$ it receives is $\hat{\beta}_{-j}$. But, combining this with X^j and y^j to obtain the global regression coefficients is not straightforward. One possible approach, if j knew that the regression without it were a good one—which it doesn't—would be to simulate data from the other owners, combine those with its own real data, and perform a regression. However, $\hat{\beta}_{-j}$ does not contain sufficient information to do the simulation. Whether Bayesian methods will work is a topic of ongoing research.

5 Conclusions and Discussion

In this paper 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 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. A prototype software system was described, together with initial approaches to complex data partitioning and reducing incentives to "cheat."

Many research challenges remain, several of which are noted in the paper. One of the most central of these is to link our approach to traditional concerns in statistical disclosure limitation (SDL) (Willenborg and de Waal, 1996, 2001), which focus on protecting the identifies of individual data records and sensitive attributes within them.

Acknowledgements

This research was supported by NSF grant EIA–0131884 to the National Institute of Statistical Sciences (NISS). 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.

References

Benaloh, J. (1987). Secret sharing homomorphisms: Keeping shares of a secret sharing. In Odlyzko, A. M., editor, *CRYPTO86*, pages 251–260. Springer–Verlag. Lecture Notes in Computer Science No. 263.

Bishop, Y. M. M., Fienberg, S. E., and Holland, P. W. (1975). *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, Cambridge, MA.

Dempster, A., Laird, N., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc., Series B*, 39(1):1–38.

Du, W. and Zhan, Z. (2002). A practical approach to solve secure multi-party computation problems. In *New Security Paradigms Workshop*, pages 127–135, New York. ACM Press.

Goldwasser, S. (1997). Multi-party computations: Past and present. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, pages 1–6, New York. ACM Press.

Karr, A. F., Feng, J., Lin, X., Reiter, J. P., Sanil, A. P., and Young, S. S. (2005a). Secure analysis of distributed chemical databases without data integration. *J. Computer-Aided Molecular Design*, November, 2005:1–9. Available on-line at www.niss.org/dgii/technicalreports.html.

Karr, A. F., Lin, X., Reiter, J. P., and Sanil, A. P. (2004). Analysis of integrated data without data integration. *Chance*, 17(3):26–29.

Karr, A. F., Lin, X., Reiter, J. P., and Sanil, A. P. (2005b). Secure analysis of distributed databases. In Olwell, D. and Wilson, A. G., editors, *Statistical Methods in Counterterrorism*, ASA/SIAM Series on Statistics and Applied Probability, pages 199–220. SIAM, Philadelphia. To appear. Available on-line at www.niss.org/dgii/technicalreports.html.

Karr, A. F., Lin, X., Reiter, J. P., and Sanil, A. P. (2005c). Secure regression on distributed databases. *J. Computational and Graphical Statist.*, 14(2):263–279.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. Wiley, New York. 2nd Edition.

Reiter, J. P. (2005). Releasing multiply-imputed, synthetic public use microdata: An illustration and empirical study. *J. Royal Statist. Soc., Series A*, 168:185–205.

Sanil, A. P., Karr, A. F., Lin, X., and Reiter, J. P. (2004a). Privacy preserving analysis of vertically partitioned data using secure matrix products. *J. Official Statist.* Submitted for publication. Available on-line at www.niss.org/dgii/technicalreports.html.

Sanil, A. P., Karr, A. F., Lin, X., and Reiter, J. P. (2004b). Privacy preserving regression modelling via distributed computation. In *Proc. Tenth ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining*, pages 677–682. Available on-line at www.niss.org/dgii/technicalreports.html.

Schafer, J. L. (1997). *Analysis of Incomplete Multivariate Data*. Chapman & Hall, London.

Vera, F., Fulp, W. J., and Karr, A. F. (2005). Software for secure analysis of distributed databases. In preparation.

Willenborg, L. C. R. J. and de Waal, T. (1996). *Statistical Disclosure Control in Practice*. Springer–Verlag, New York.

Willenborg, L. C. R. J. and de Waal, T. (2001). *Elements of Statistical Disclosure Control*. Springer–Verlag, New York.

Yao, A. C. (1982). Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, New York. ACM Press.