The Annals of Applied Statistics
2018, Vol. 12, No. 2, 877–914
https://doi.org/10.1214/18-AOAS1171

PROVIDING ACCURATE MODELS ACROSS PRIVATE PARTITIONED DATA: SECURE MAXIMUM LIKELIHOOD ESTIMATION

BY JOSHUA SNOKE[*,1], TIMOTHY R. BRICK[*], ALEKSANDRA SLAVKOVIĆ[*,1] AND MICHAEL D. HUNTER[†,2]

Pennsylvania State University[*] and University of Oklahoma Health Sciences Center[†]

This paper focuses on the privacy paradigm of providing access to researchers to remotely carry out analyses on sensitive data stored behind separate firewalls. We address the situation where the analysis demands data from multiple physically separate databases which cannot be combined. Motivating this work is a real model based on research data on kinship foster placement that came from multiple sources and could only be combined through a lengthy process with a trusted research network. We develop and demonstrate a method for accurate calculation of the multivariate normal likelihood, for a set of parameters given the partitioned data, which can then be maximized to obtain estimates. These estimates are achieved without sharing any data or any true intermediate statistics of the data across firewalls. We show that under a certain set of assumptions our method for estimation across these partitions achieves identical results as estimation with the full data. Privacy is maintained by adding noise at each partition. This ensures each party receives noisy statistics, such that the noise cannot be removed until the last step to obtain a single value, the true total log likelihood. Potential applications include all methods utilizing parameter estimation through maximizing the multivariate normal likelihood. We give detailed algorithms, along with available software, and present simulations and analyze the kinship foster placement data estimating structural equation models (SEMs) with partitioned data.

1. Introduction. In many real-life settings, researchers wish to utilize data from separate databases but are unable to physically combine the data due to restrictions such as privacy concerns, proprietary issues, sheer size of the data, or massively distributed data such as proposed by Boker et al. (2015). Consider situations such as with health data where a researcher wishes to carry out a longitudinal study on PTSD for veterans who utilize different hospitals and primary care physicians, with education data where students switch between schools, or with

Received November 2017; revised April 2018.

[1]Supported in part by NSF Grants BCS-0941553 and SES-1534433 to the Department of Statistics, Pennsylvania State University. Supported also in part by the U.S. Census Bureau.

[2]Supported in part by Grant HHS-2012-ACF-ACYF-CF-0510 awarded to NorthCare by the Children's Bureau.

Key words and phrases. Partitioned data, privacy, secure multiparty computation, structural equation models, distributed maximum likelihood estimation.

behavioral psychology in a twin study where the twins do not wish to share their data with each other. In all of these cases, standard practice is to go through user agreements or to develop trusted data centers which are allowed to pool data. Both of these options take lengthy periods of time, and sometimes it is not even possible to establish such a trusted party. Alternatively, estimates may be obtained through algorithms that pool information across the databases to produce aggregrate statistics or estimate models without sharing the data. When privacy is a concern, we want to do this pooling of information in a "secure" manner. This implies two elements: first that the private inputs, the data, from each database are not shared or revealed, and second that the estimates produced are accurate to what would be produced if all the data were combined. While guaranteeing the data are not shared, different algorithms can provide different levels of security based on the potential for data leakage through the sharing of intermediate values.

Motivating this work is a real model based on research data that came from multiple sources and could only be combined through a lengthy process with a trusted research network. Specifically, we utilize data collected between 2012 and 2015 on kinship foster placements from a collaboration between the University of Oklahoma Health Sciences Center (OUHSC), the Oklahoma Department of Human Services (OKDHS), and the North Oklahoma County Mental Health Center (NorthCare). Because of the partnership between OUHSC, OKDHS, and NorthCare all data could be gathered and merged into a single data table for analyses, but establishing trust and data sharing agreements between these kinds of agencies in many states is often difficult and tenuous at best. Though the government and university institutions involved in the data collection have been collaborating for over twenty years, the data sharing agreement still took months to establish.

Similar agreements often never come to fruition because sufficient trust and legal protection cannot be forged to release potentially sensitive and identifying information to outside institutions. Moreover, laws (e.g., the Health Insurance Portability and Accountability Act and the Family Educational Rights and Privacy Act) sometimes preclude data transfers without special permissions from the individuals themselves. Even when data sharing is mandated by funding agencies or journals, the compliance rate is typically near zero percent, for example, as Savage and Vickers (2009) showed using the PLoS journals. However, without gathering all the data into a single location, only aggregated summaries would typically be possible and no relationships across variables stored in separate places could be investigated. In particular, no statistical model could be built from data stored at separate locations without at some point bringing all the data to the same location. Our work shows that for a wide class of models results can be reproduced accurately without needing to physically combine the data, providing a mechanism to allow statistical model building across multiple separate data sources, obviating the need for many data sharing agreements, and dramatically reducing the amount of trust required for institutions to collaborate.

A key feature of this paper is that we present our algorithm with real applicability in mind. We imagine this method working inside a framework that includes a research network of data holders who have agreed to be part of a research network in which they allow their data to be queried using the secure algorithm, but they do not share their data. In addition to the parties holding data, our algorithm includes a "central" party who is responsible for facilitating researchers' model requests and initiating the algorithm. Past work has focused on hypothetical situations where two or more groups *who all hold data* have an interest in jointly computing some statistic. While our framework allows for the possibility that a data holder also has research questions, we believe it is at least as useful if not more so to assume that the researcher is an outside entity who wishes to query a model across various data sets which they are unable to physically access. In that case, the data nodes play no role in choosing the model but have agreed, as part of being in the research network, to go along with the algorithm.

Our method relies on the framework known as secure multiparty computation (SMPC), which was introduced and originally termed such by Yao (1982) [see also Goldwasser (1997), Lindell and Pinkas (2009)]. Various algorithms using SMPC have been proposed in the statistical disclosure control (SDC) literature for the purpose of statistical inference. While much of the SDC work has focused on releasing noisy microdata sets through perturbation, suppression, or generation of synthetic data [see Fienberg and Slavković (2011), Hundepool et al. (2012), Raghunathan, Reiter and Rubin (2003), Willenborg and de Waal (2001) for more on these methods], the work based on SMPC has focused on providing accurate model estimates without sharing the data and minimizing sharing of intermediate statistics. This is essentially a secure query on multiple physically separate data sets.

Partitioned data can be classified as vertically, horizontally, or complexly partitioned; see Figure 1. Vertical partitions imply each database holds the same set

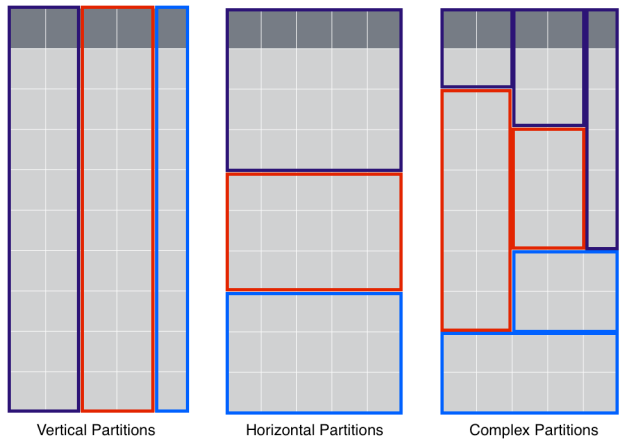

FIG. 1. *Data partition types, with rows for individual entries and columns for attributes.*

of individuals but different variables. The converse, horizontal partitions, implies common variables across databases but different sets of individuals. Finally, complex partitions imply some combination of vertical and horizontal. Another case, which we do not address in this paper, is the case of overlapping partitions. Reiter et al. (2004) considered this problem, but it fundamentally changes the security question. In this paper, we will assume the data sets are not overlapping, but that they can be correctly linked using a unique identifier that is shared across all databases. This unique ID is necessary in the vertical partitions case because each database will contain different variables measured on the same individuals, and we need to correctly match these individuals across databases.

SMPC techniques have been used for parameter estimation in various papers previously in the SDC literature. For example, Karr et al. (2007) and Lin and Karr (2010) proposed distributed likelihood estimation (DLE) for horizontally and vertically partitioned data using secure sums and oblivious transfer. Karr et al. (2009) proposed a secure matrix multiplication algorithm to estimate the sample data covariance matrix without combining data, and Ghosh, Reiter and Karr (2007) proposed adaptive regression splines for horizontal data. Sanil et al. (2004), Karr et al. (2005), Fienberg et al. (2006), Fienberg, Nardi and Slavković (2009), and Nardi, Fienberg and Hall (2012) proposed various methods for secure logistic regression and log-linear models for categorical data. There is also a large body of literature on achieving data mining or machine learning results through SMPC [e.g., see Vaidya and Clifton (2004) or Vaidya et al. (2008)], but this literature is typically concerned only with prediction and not parameter estimation and interpretation.

When dealing with horizontally partitioned data, a secure routine can often be devised easily, but it is more tricky with vertically or complexly partitioned data. Previous work related to ours focused on two strategies for vertical partitions, either devising secure methods to compute the sample covariance matrix [e.g., Karr et al. (2009)] or finding maximum likelihood estimates for specific models such as logistic regression [e.g., Nardi, Fienberg and Hall (2012)], including performing Lasso [e.g., Samizo (2016)]. In this paper, we will not compute the sample covariance in closed form but achieve it by maximum likelihood. By doing this for the multivariate normal parameters, rather than specific parameters for models such as logistic regression, we allow the flexibility to fit a wider class of models.

We rely on a couple of common assumptions used in SMPC algorithms, which are also included in all of the previous related work. We assume semi-honesty, which means that all parties will follow the algorithm as stated, and we assume that they will not collude to try to uncover a third party's information. These two assumptions have been termed "honest but curious," coined by Kissner and Song (2005), and it allows that the parties may use available information to try to uncover other parties' information. Another assumption is that the final output is itself not risky, such as an estimated covariance matrix in the case of vertically

partitioned regression; see Karr et al. (2009). We also make that assumption here, while noting that this may very well not always be the case. Further work should consider methods to combine our algorithm with efforts to minimize any risk of releasing model estimates, but here we focus on getting accurate estimates without sharing data or intermediate statistics, minimizing data leakage.

Ultimately, this paper presents a secure algorithm to calculate the correct multivariate normal log likelihood for a set of parameters (mean and covariance matrix) given the partitioned data, which can be used with standard optimization techniques to produce maximum likelihood estimates (MLEs) of the parameters. This extends our previous algorithm given in Snoke, Brick and Slavković (2016), where we showed it was possible to accurately compute multivariate normal maximum likelihood estimates without passing any data or statistics that allow for immediate reconstruction of the data. In this paper, we go a step further and show that by using secure multiparty computation techniques we can get accurate estimates without sharing any true information between partitions, minimizing any possible data leakage. From a privacy standpoint, this greatly reduces disclosure risk due to the algorithm, leaving only the risk from disclosure due to releasing final model estimates.

For inference, the MLEs of the multivariate normal parameters are commonly used to estimate a variety of models, such as linear models, factor analysis, principal components analysis (PCA), or structural equation models (SEMs). This approach provides a more general modeling framework from previous work, which focused on specific model classes. Under our algorithm, any analysis relying on the multivariate normal MLE can be performed across partitioned data sources. Additionally, our algorithm naturally extends to complex partitions (see Figure 1), which we demonstrate, while previous methods have focused solely on horizontal or vertical partitions (offering only hints of solutions for complex partitions). Finally, our method reduces the remaining privacy risks in the vertical partition setting over previous methods of computing the sample covariance matrix directly through methods such as secure matrix multiplication, and the risk under our algorithm does not increase with the number of partitions, as is the case for some previous methods which we will discuss further in Section 2.

The rest of this paper is organized as follows. Section 2 reviews related prior methods and discusses potential security or computational improvements. Section 3 covers the mechanics of distributed likelihood estimation. Section 4 presents a detailed walk-through of our secure algorithm. Section 5 presents simulations to test the computational complexity and accuracy of the algorithm. Section 6 addresses the motivating real data problem, discussing various models, and showing accurate results without combining the data. Section 7 gives final remarks and discussion.

2. Previous secure methods. Here, we detail some of the relevant prior methods used for secure MLE and covariance modeling, and we cover some of the ways

in which we improve on these methods. We focus on previous algorithms that allow for general covariance modeling to show how we improve on them with respect to data leakage. We do not cover specific secure methods such as linear or logistic regression, since our method generalizes to a larger class of models. We also focus on the vertical partition case, since secure MLE in the purely horizontal case has been shown to be easily generalizable [see, e.g., Karr et al. (2007)].

2.1. *Sample covariance estimation by secure matrix multiplication.* One of the straight-forward applications of secure protocols with statistical quantities is an algorithm for calculating the sample covariance by secure matrix multiplication. The computation of the sample covariance in this way allows for various model estimation, most notably linear and logistic regression as shown in Karr et al. (2009), Fienberg, Nardi and Slavković (2009), and Slavkovic, Nardi and Tibbits (2007). This is a general estimation algorithm, similar to our method; see Algorithm 10 in Appendix D for details.

The strength of this algorithm is that it is fairly simple and easy to implement. It can also be implemented when only two parties exist, which we will see in Section 2.2 is not always possible. The downside to this algorithm is that there exists some data leakage, so it is possible the parties involved can learn about the other parties' data. Karr et al. (2009) gives a preliminary explanation of the data leakage, and Samizo (2016) provides an in-depth analysis. Simply put, each party will learn a certain number of linearly independent constraints on the other party's data matrix (contingent on the choice of a in step 1 of Algorithm 10). Additionally, if there are more than two parties, the off-diagonal elements will need to be computed for each pair of databases, increasing both the computational burden [$O(n^2)$] and more importantly the data leakage. This is because the routine will need to be run for every pairwise combination of parties and every party will learn a certain number of linearly independent constraints on every other parties' data, implying an increase in data leakage for an increase in the number of data parties.

2.2. *Maximum likelihood estimation by secure summation and oblivious transfer.* In the case of horizontally partitioned data, Karr et al. (2007) and Lin and Karr (2010) describe a method to get MLE estimates for general exponential family models using secure summation. This is a fairly straightforward method, since each partition relies on the same parameters and can compute the likelihoods individually.

Suppose each party computes the log likelihood for a set of parameters given only their data. In the case of K parties, we can refer to these values as $LL_1, LL_2, \ldots, LL_K$, respectively. We can get the total log likelihood sum in a secure manner by adding random noise at the beginning and removing this noise only when all intermediate summations have been completed. This routine is known as secure summation, and this guarantees that as long as the noise is adequately large

Algorithm 1 Secure Summation (three or more parties)

Input: $LL_1, LL_2, \ldots, LL_K$ held by parties 1, 2, ..., K, respectively
Output: $LL_{1+2+\cdots+K}$

1: Party 1 computes $\tilde{LL}_1 = LL_1 + R$ where R is a large random value
2: Party 2 receives $\tilde{LL}_1$ and computes $\tilde{LL}_{1+2} = \tilde{LL}_1 + LL_2$
3: **for** i in $3, \ldots, K$ **do**
4: Party i receives $\tilde{LL}_{1+2+\cdots+i-1}$ and computes $\tilde{LL}_{1+2+\cdots+i} = \tilde{LL}_{1+2+\cdots+i-1} + LL_i$
5: **end for**
6: Party 1 receives $\tilde{LL}_{1+2+\cdots+K}$ and computes $LL_{1+2+\cdots+K} = \tilde{LL}_{1+2+\cdots+K} - R$
7: Party 1 shares $LL_{1+2+\cdots+K}$ with all other parties

and random, only the final sum will be learnable to all parties. The secure summation steps are shown in Algorithm 1.

This routine also allows for a clear understanding of collusion. If party 1 were to collude with party 3, they could send $\tilde{LL}_1$, their noisy log likelihood, to party 3. This would allow party 3 to learn LL_2, disclosing party's 2 information at no cost to parties 1 or 3. Thus in this horizontal setting with secure summation, we must assume no collusion to ensure security. We must also assume at least three parties, since the other party's value could be learned from the total sum in the case of only two parties.

For vertically partitioned data, Lin and Karr (2010) offer a method to get MLE estimates for the general exponential family; see Algorithm 11 in Appendix D for details. This method is nice because it generalizes to the exponential family, but it is not as strong from a security standpoint. In the oblivious transfer protocol, the computational burden increases significantly with the ability to obscure the data, so in some cases satisfactory privacy is (computationally) difficult to achieve. Also each party learns certain sufficient statistics, which in the case of the multivariate normal log likelihood function is the sum of the other party's data. This could be highly disclosive in certain situations. A final drawback, as with the secure matrix multiplication, is that this routine will need to be run for each pairwise combination of parties if there are more than two total parties, again increasing the risk as the number of parties grows.

Disclosure from sharing of aggregated statistics has long been researched at places such at the Census Bureau, where the concern primarily stemmed from outliers; see Sullivan (1992). More recently, many works stemming from the computer science literature have shown there are significant risks, particularly in the case of high dimensional data and in the presence of other external sources, e.g., see Dinur and Nissim (2003), Homer et al. (2008), and Calandrino, Kilzer, Narayanan, Felten and Shmatikov (2011). Such cases, in part, have motivated the rise of *Differential Privacy (DP)*, a formal framework for defining the worst-case risk. For further re-

view of this theory, see Dwork (2008). DP is beyond the scope of this paper, but in Section 7 we address a potential interplay of DP with our proposed method.

3. Distributed likelihood estimation. Distributed likelihood estimation (DLE) refers generally to the concept of calculating a likelihood for estimation purposes, that is, using maximum likelihood, in separate pieces and combining the resulting likelihoods to get the total. We discuss it here specifically in the context of partitioned data, where likelihoods must be calculated at separate databases and combined to get estimates because the data cannot be physically combined. Furthermore, with partitioned data, maximum likelihood must be achieved through an optimization routine, such as gradient descent, because the closed-form function requires a nonpartitioned data matrix.

To facilitate this optimization, we assume a central party exists who controls no data but chooses the initial parameters (based upon a requested or agreed model), the intermediate parameters, and ultimately the convergence criterion. The central node does not control any data, but it is possible they are the most interested in the estimates. This can be imagined as a research node where users can request models and receive estimates. Note that this does *not* mean the central party is a trusted party, as has been implemented by de Montjoye et al. (2014) or Gaye et al. (2014). The central node, along with the data nodes, does *not* receive any data or intermediate statistics from the other nodes which is a typical case with a trusted third party.

A strength of our setup is that having a central party allows us to utilize secure summation techniques with only two data-holding parties, since there are still three total parties. As discussed in the Introduction, the central party also enables the practicality of our algorithm. Previous methods have either utilized a trusted central party to facilitate an implementation of their algorithm, or they have only addressed a practical implementation in general terms. For example, the secure matrix multiplication or secure summation methods given in Section 2 do not specifically address the question of the origin of the research question(s) or model choice(s).

3.1. *Modeling by multivariate normal MLE.* We focus on the multivariate normal MLE, a well-known and flexible modeling framework which allows for the estimation of a variety of models such as linear models, factor models, PCA, and SEMs. Assuming a multivariate normal distribution, the goal of MLE is to maximize the likelihood equation:

$$
\begin{aligned}
L(\mu, \Sigma | Z) &= \prod_{i=1}^{n} L_i(\mu_i, \Sigma_i | z_i) \\
&= \prod_{i=1}^{n} (2\pi)^{-\frac{1}{2}p_i} |\Sigma_i|^{-\frac{1}{2}} e^{(z_i-\mu_i)\Sigma_i^{-1}(z_i-\mu_i)^T}
\end{aligned} \tag{1}
$$

for mean and covariance parameters $\mu_i \in \mathbb{R}^{p_i}$ and $\Sigma_i \in \mathbb{R}^{p_i \times p_i}$ given a data matrix $Z \in \mathbb{R}^{n \times p}$. We will work with the log likelihood:

$$\begin{aligned}
&\ell(\mu, \Sigma|Z) \\
&\quad = \sum_{i=1}^{n} \ell_i(\mu_i, \Sigma_i|z_i) \\
&\quad = \sum_{i=1}^{n} -\frac{1}{2}\left[p_i * \log(2 * \pi) + \log(|\Sigma_i|) + (z_i - \mu_i)\Sigma_i^{-1}(z_i - \mu_i)^T\right]
\end{aligned} \tag{2}$$

rather than the likelihood, since it is easier to work with sums than products across each data row. We use the formulation that allows the parameters to vary for each data row, that is, the full-information likelihood [e.g., see Arbuckle, Marcoulides and Schumacker (1996)]. This is often used to deal with missing data in order to compute a likelihood for rows with missing elements rather than performing listwise deletion. We use it because the log likelihood will be calculated in separate partitions, and in the vertical case the parameters are partitioned to only pertain to certain variables in the data matrix (see Section 3.3 for more details). Maximizing over equation (2) gives the MLE estimates $\hat{\mu}$ and $\hat{\Sigma}$. With the full data matrix, the maximum can be found in closed form, but in our case because the data are partitioned we can find the maximum by using any standard optimization routine.

3.2. *Horizontally distributed likelihood.* Horizontally partitioned data are separated by observations, such that each database has different sets of individuals all measured on the same variables. The parameters are the same for each partition, so DLE in this case requires only a secure summation of total log likelihoods from each partition as described in Section 2.2. If we consider the log likelihood given in equation (3), we can separate this into different elements based on the rows in each database. All variables are the same across databases, so the parameters relating to the data are the same in each. This is shown in equation (4) for a theoretical combined database Z with partitions $\begin{pmatrix} X_1 & X_2 & X_3 \end{pmatrix}^T$, where X_k denotes the data matrix of the kth partition. Note that these partitions do not need to be of equal size:

$$\ell(\mu, \Sigma|Z) = \sum_{i=1}^{n} \ell_i(\mu_i, \Sigma_i|z_i), \tag{3}$$

$$\ell(\mu, \Sigma|Z) = \sum_{i=1}^{n_{X_1}} \ell_i(\mu, \Sigma|x_{1i}) + \sum_{i=1}^{n_{X_2}} \ell_i(\mu, \Sigma|x_{2i}) + \sum_{i=1}^{n_{X_3}} \ell_i(\mu, \Sigma|x_{3i}). \tag{4}$$

In this case, each node receives the full set of model-defined parameters from the central optimizer and calculates their portion of the total log likelihood. The results are added using a secure summation and new parameter estimates are chosen for

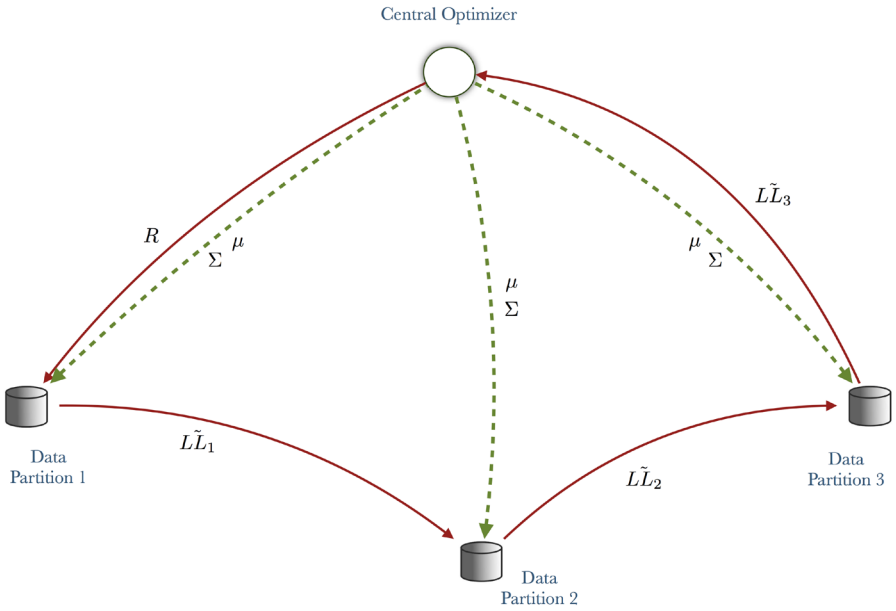

FIG. 2. *Example of a three node system using secure summation for horizontally partitioned data. Red solid lines denote path of noisy log likelihood, green dashed lines denote path of model-implied parameters.*

the next step of the optimization; see Figure 2 for a visual depiction. In the horizontal case, as noted by Karr et al. (2007), the form of the likelihood function does not matter, so applications are not constrained to the multivariate normal case.

3.3. *Vertically distributed likelihood.* Vertically partitioned data are separated by variables, with each database containing different measurements on the same set of individuals. This implies that the set of parameters used in calculating the log likelihood differs at each partition. As shown in Snoke, Brick and Slavković (2016), DLE must combine log likelihoods calculated using marginal and conditional parameters to get the correct total log likelihood. Note that complex partitions can be formulated as a combination of horizontal and vertical DLE, which we discuss further in Section 4.3.

Equation (5) shows the decomposition of the log likelihood function in the vertical case, where $Z = \begin{pmatrix} X_1 & X_2 & X_3 \end{pmatrix}$ and X_k is the data matrix of the kth partition. Note that these partitions do not need to be of equal size. This adds complexity because some intermediate parameters and statistics must be calculated and shared in order to accurately calculate the total log likelihood. These must be shared securely, and a simple secure summation of the log likelihoods is not possible because each partition needs different parameters. We address this further in

Section 4.3:

$$\ell(\mu, \Sigma|Z) = \sum_{i=1}^{n} \ell_i(\mu_{X_1}, \Sigma_{X_1}|x_{1i}) + \sum_{i=1}^{n} \ell_i(\hat{\mu}_{X_2|X_1}, \Sigma_{X_2|X_1}|x_{2i}) + \sum_{i=1}^{n} \ell_i(\hat{\mu}_{X_3|X_2,X_1}, \Sigma_{X_3|X_2,X_1}|x_{3i}). \tag{5}$$

3.3.1. *Marginal and conditional parameters for vertical partitions.* To calculate the partitioned log likelihoods, we need to calculate marginal and conditional parameters from μ and Σ. Subsetting the parameters as shown in equations (6) and (7) for the combined data set, we can write the conditional parameters as shown in equations (8), (9), (10), and (11). These relationships come from the well-known properties of the multivariate normal distribution and are also known as the Schur complement in matrix theory [e.g., see Schur (1905) and Haynsworth (1968)]:

$$\Sigma = \begin{pmatrix} \Sigma_{X_1X_1} & \Sigma_{X_2X_1} & \Sigma_{X_3X_1} \\ \Sigma_{X_1X_2} & \Sigma_{X_2X_2} & \Sigma_{X_3X_2} \\ \Sigma_{X_1X_3} & \Sigma_{X_2X_3} & \Sigma_{X_3X_3} \end{pmatrix}, \tag{6}$$

$$\mu = \begin{pmatrix} \mu_{X_1} & \mu_{X_2} & \mu_{X_3} \end{pmatrix}. \tag{7}$$

$\Sigma_{X_kX_k}$ and $\Sigma_{X_kX_l}$ are the model-implied marginal covariance elements for the variables in X_k and the model-implied covariance elements between the variables in X_k and X_l, respectively. μ_{X_k} are the mean parameters for the variables in X_k.

It is important to note here that these mean and covariance parameters are *not* the observed sample estimates from the data, since it would be impossible to calculate $\Sigma_{X_kX_l}$ across partitioned data without sharing data. They are parameters defined by our model with values chosen at each step according to the optimization routine. The exception, though, are the conditional mean parameters that are estimated using real data [see equations (9) and (11)], and thus are denoted with the hat notation. Based on these equations, we can estimate a distributed log likelihood for vertical partitions that is equivalent to the joint log likelihood.

Before proceeding, a note on notation: let X^+ denote all variables following X in the node sequence, that is, $X_1^+ = (X_2, X_3)$ (the variables in node 2 and 3 for a 3-node system) and $X_2^+ = (X_3)$. In the same way, let X^- denote all prior variables.

Condition X_1^+ on X_1:

$$\Sigma_{X_1^+|X_1} = \Sigma_{X_1^+X_1^+} - \Sigma_{X_1X_1^+}\Sigma_{X_1X_1}^{-1}\Sigma_{X_1^+X_1} = \begin{pmatrix} \Sigma_{X_2X_2|x} & \Sigma_{X_2^+X_2|X_1} \\ \Sigma_{X_2X_2^+|X_1} & \Sigma_{X_2^+X_2^+|X_1} \end{pmatrix} \tag{8}$$

and

$$\hat{\mu}_{X_1^+|X_1} = \mu_{X_1^+} + \Sigma_{X_1X_1^+}\Sigma_{X_1X_1}^{-1}(X_1 - \mu_{X_1}) = \begin{pmatrix} \hat{\mu}_{X_2|X_1} & \hat{\mu}_{X_2^+|X_1} \end{pmatrix}. \tag{9}$$

Condition X_2^+ on X_1, X_2:

$$\begin{aligned}\Sigma_{X_2^+|X_3^-} &= \Sigma_{X_2^+X_2^+|X_1} - \Sigma_{X_2X_2^+|X_1}\Sigma^{-1}_{X_2X_2|X_1}\Sigma_{X_2^+X_2|X_1} \\ &= \begin{pmatrix} \Sigma_{X_3X_3|X_3^-} & \Sigma_{X_3^+X_3|X_3^-} \\ \Sigma_{X_3X_3^+|X_3^-} & \Sigma_{X_3^+X_3^+|X_3^-} \end{pmatrix}\end{aligned} \tag{10}$$

and

$$\begin{aligned}\hat{\mu}_{X_2^+|X_3^-} &= \hat{\mu}_{X_2^+|X_1} + \Sigma_{X_2X_2^+|X_1}\Sigma^{-1}_{X_2X_2|X_1}(X_2 - \hat{\mu}_{X_2|X_1}) \\ &= \begin{pmatrix} \hat{\mu}_{X_3|X_3^-} & \hat{\mu}_{X_3^+|X_3^-} \end{pmatrix}.\end{aligned} \tag{11}$$

This conditioning process can be repeated for as many partitions as necessary. Next, we combine these with techniques from SMPC to produce a secure algorithm.

4. Secure algorithm for vertically partitioned data. Here, we describe in more detail our proposed algorithm for secure multiparty log likelihood estimation for vertically partitioned data with K partitions (or nodes). This also extends to complex partitions as discussed in Section 4.3.

4.1. *Notation*. The following terms are used in the algorithm and their corresponding equations. For nodes $k \in 1, \ldots, K$ with data $\begin{pmatrix} X_1 & X_2 & \ldots & X_K \end{pmatrix}$ with dimensions $(n \times p_1), (n \times p_2), \ldots, (n \times p_K)$:

- P_k, R_k, Q_k, M_k random noise matrices with dimensions $(n \times p_k), (n \times p_k), (p_k \times n), (n \times p_k)$, respectively.
- $A_k^1 = \Sigma^{-1}_{X_kX_k|X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} + R_k)$.
- $A_k^2 = \Sigma^{-1}_{X_kX_k|X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} - R_k) + Q_k$.
- $B_k = \tilde{\mu}_{X_k^+|X_k^-} + \Sigma_{X_kX_k^+|X_k^-}A_k^1$.
- $C_k = \Sigma_{X_kX_k^+|X_k^-}(\Sigma_{X_kX_k|X_k^-})^{-1}$.
- $\tilde{\mu}_{X_k^+|X_{k+1}^-} = \begin{pmatrix} \tilde{\mu}_{X_{k+1}|X_{k+1}^-} & \tilde{\mu}_{X_{k+2}|X_{k+1}^-} & \cdots & \tilde{\mu}_{X_K|X_{k+1}^-} \end{pmatrix} = B_k - M_k - C_k(R_k - P_k)$.

As noted before, the $+$ and $-$ notation denote all variables in partitions following or preceding a node, for example, $\mu_{X_1^+|X_2^-}$ denotes the conditional mean parameters for all variables in node 2 and following, conditional on all variables in node 1. The $\sim$ and $*$ notation denote noisy versions of the true statistics. There are two noise notation, since the same statistics will have multiple stages of noise addition or removal.

4.2. *Adaptation from the nonsecure algorithm.* From Snoke, Brick and Slavković (2016), Algorithm 12, shown in Appendix D, gives a method for estimating the joint log likelihood without sharing any data, but it does not provide that nothing can be learned from the statistics which are passed. There are four elements computed and shared in this original algorithm in order to obtain correct parameter estimates that we consider to be risky. Two are statistics of the data and two converge to statistics (recall that we use optimization to find maximum likelihood estimates of Σ and μ). For each node k they are:

1. $\Sigma_{X_k X_k | X_k^-}$.
2. $\hat{\mu}_{X_k | X_k^-}$.
3. $\Sigma_{X_k X_k^+ | X_k^-}$.
4. LL_k.

We handle these elements as follows. First, rather than the central node sending Σ and μ to the first data node and each node calculating the following conditional covariance parameters, the central node will calculate $\Sigma_{X_k X_k | X_k^-}$ for each node k. These will be distributed only to each node with the corresponding variables, so nodes will not learn the covariance parameters for other nodes' variables.

Second, the central node no longer sends μ to the first node for each node to do the conditioning. The true conditional means are risky even for data nodes containing the corresponding variables to see because they are statistics that depend on all the previous nodes' data. To protect this information, we add noise to μ. We also must calculate them in parts and combine the pieces in such a way that true values are not recoverable, since the conditional means are calculated using unshareable information both from the data nodes and the central node.

Next, we protect the off-diagonal covariance elements by not passing them unless combined with other terms which cannot be differenced out. Though the off-diagonal elements are not statistics, it is possible through the iteration of the optimization routine that a partition could learn how another partition's data covary with their own data. Since they know their own data, they may disclose information about the other partition's data.

Lastly, we share noisy log likelihood totals instead of the true values, since these are statistics of data. There are two noise components to the log likelihood. First, by adding noise to the mean parameters this introduces noise into the log likelihoods. Second, we use an altered log likelihood form which allows us to record the noise introduced and remove it at the final step after safely passing it through all the nodes.

We can think of the secure algorithm as having two primary functions: one being a secure summation of log likelihoods across partitions and the second being a multiparty computation of the conditional mean parameters for each node. Vitally, the solution to these goals work together both in obscuring the true values and in removing all the noise by the end in order to obtain the correct joint log likelihood

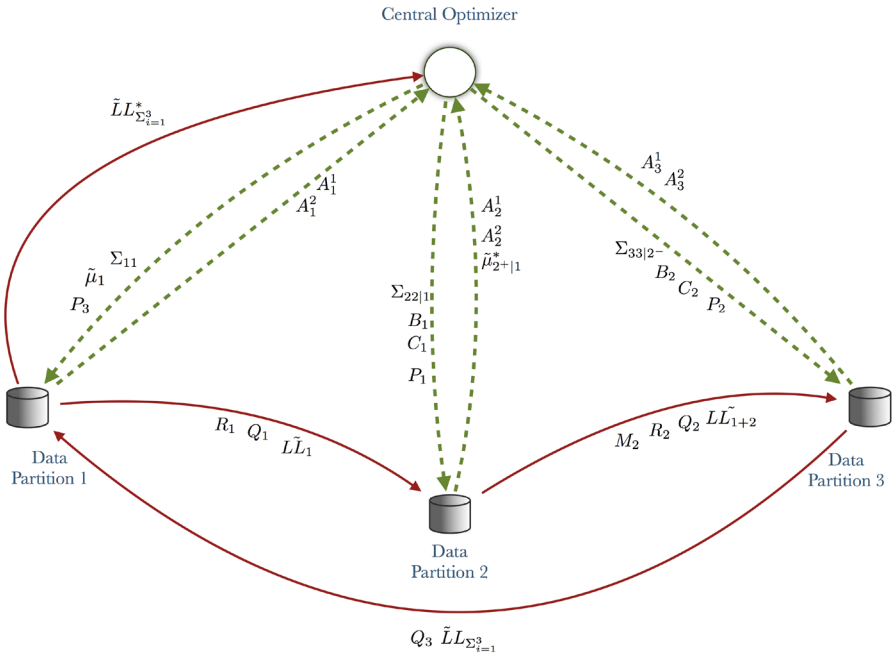

FIG. 3. *Example three node secure algorithm visual following Algorithm 2. Red solid lines denote path of noisy log likelihood, green dashed lines denote path of conditional parameters.*

value. In Figure 3, we give a visual representation of the algorithm for a three node system, and we highlight these two complimentary functions. The solid red lines denote the flow of the noisy log likelihood, and the dashed green lines denote the steps in computing the conditional mean parameters.

4.3. *Algorithm walk-through.* Algorithm 2 gives the complete K data partition secure process for calculating the total log likelihood across vertical partitions. There are a number of internal algorithms referenced, which are given in Appendix C and named as shown. We will refer to them by number in the following description of the algorithm. To help with understanding the details, Appendix B gives a small numerical example showing each step of the algorithm explicitly.

This algorithm represents one step in an optimization routine, returning the total log likelihood value for a given set of parameters which will then be used to choose new values until convergence. The inputs are parameters, μ and Σ, according to an assumed multivariate normal distribution. The output is a single number, the sum of log likelihoods across the entire set of partitioned databases.

In the initial step of Algorithm 2, shown in Algorithm 3, the central node, which runs the optimization and has no data, partitions the model-defined parameters corresponding to the variables present at each data node. It then produces the marginal

Algorithm 2 Secure Multiparty Log Likelihood Estimation for Vertical Partitions

Input: $\mu \in \mathbb{R}^p$, $\Sigma \in \mathbb{R}^{p \times p}$ (central node), $X_k \in \mathbb{R}^{n \times p_k}$ $\forall k \in K$ (data nodes)
Output: $LL_{\sum_{j=1}^{K} j}$

1: CN compute: $CN\_Initiate(\mu, \Sigma)$
2: CN $\rightarrow DN_1$: $\Sigma_{X_1 X_1}$, $\tilde{\mu}_{X_1}$, P_K
3: DN_1 compute: $EN\_Compute(\tilde{\mu}_{X_1}, \Sigma_{X_1 X_1}, X_1, \varnothing)$
4: $DN_1 \rightarrow$ CN: A_1^1, A_1^2
5: $DN_1 \rightarrow DN_2$: $\tilde{L}L_1$, R_1, Q_1
6: **for** k in $2, \ldots, K$ **do**
7: CN compute: $CN\_Adjust(A_{k-1}^1, \tilde{\mu}_{X_{k-1}^+ | X_{k-1}^-}, \Sigma_{X_{k-1} X_{k-1} | X_{k-1}^-})$
8: CN $\rightarrow DN_k$: $\Sigma_{X_k X_k | X_k^-}$, B_{k-1}, C_{k-1}, P_{k-1}
9: DN_k compute: $EN\_Adjust(B_{k-1}, C_{k-1}, R_{k-1}, P_{k-1}, Q_{k-1}, \tilde{L}L_{\Sigma_{j=1}^{k-1} j}, M_{k-1})$
10: DN_k compute: $EN\_Compute(\tilde{\mu}_{X_k | X_k^-}, \Sigma_{X_k X_k | X_k^-}, X_k, \tilde{L}L^*_{\Sigma_{j=1}^{k-1} j})$
11: $DN_k \rightarrow$ CN: A_k^1, A_k^2
12: **if** $k! = K$ **then**
13: $DN_k \rightarrow$ CN: $\tilde{\mu}^*_{X_k^+ | X_k^-}$
14: $DN_k \rightarrow DN_{k+1}$: $\tilde{L}L_{\Sigma_{j=1}^{k} j}$, R_k, Q_k, M_k
15: **end if**
16: **end for**
17: $DN_K \rightarrow DN_1$: $\tilde{L}L_{\Sigma_{j=1}^{k} j}$, Q_k
18: DN_1 compute: $FN\_Adjust(\tilde{L}L_{\Sigma_{j=1}^{K} j}, P_k, Q_k)$
19: $DN_1 \rightarrow CN$: $\tilde{L}L^*_{\Sigma_{j=1}^{K} j}$
20: CN compute: $CN\_Final(\tilde{L}L^*_{\Sigma_{j=1}^{K} j}, P_k, A_k^1, A_k^2, \Sigma_{X_k X_k | X_k^-}\ \forall k \in K)$

and conditional covariance matrices for each partition, which it can do without any information from the data nodes. It also calculates the marginal mean vectors for each node, adding noise to them,

$$\tilde{\mu} = \begin{pmatrix} \tilde{\mu}_{X_1} & \tilde{\mu}_{X_2} & \ldots & \tilde{\mu}_{X_K} \end{pmatrix} = \begin{pmatrix} \mu_{X_1} + P_1 & \mu_{X_2} + P_2 & \ldots & \mu_{X_K} + P_K \end{pmatrix} \tag{12}$$

and saving the noise vectors $\begin{pmatrix} P_1 & P_2 & \ldots & P_K \end{pmatrix}$. It passes on to the first data node the marginal parameters for that node as well as the noise vector which was added to the part of the mean vector pertaining to the Kth node's variables (P_K). This will be used for de-noising in a later step.

The first data node uses the marginal covariance and noisy mean parameters to calculate a noisy version of the total of the log likelihood for its data using the

following formula:

$$\begin{aligned}\tilde{LL}_1 = \sum_{i=1}^{n} \big[& p_1 \log(2\pi) + \log\big(|\Sigma_{X_1X_1}|\big) \\ & + (X_{1i} - \tilde{\mu}_{X_1} + R_1)\Sigma_{X_1X_1}^{-1}(X_{1i} - \tilde{\mu}_{X_1} - R_1)^T + R_1^T \Sigma_{X_1X_1}^{-1} R_1 \big]\end{aligned} \tag{13}$$

with the R_1 random noise vector generated at the node level; see Algorithms 4 and 5 for more details. This different formula allows the log likelihood to be calculated with noise that will enable it to be passed securely to other data nodes without revealing the true value, but it is also noise which can be recorded and removed partially at each node and partially at the final step. We can rewrite it as

$$\begin{aligned}\tilde{LL}_1 = \underbrace{LL_1}_{\text{true value}} & - \underbrace{A_1^1 P_1^T - P_1 \big(A_1^2\big)^T - P_1 \Sigma_{X_1X_1}^{-1} P_1^T}_{\text{noise to be removed at the end by central node}} \\ & + \underbrace{P_1 Q_1}_{\text{noise to be removed by next data node}},\end{aligned} \tag{14}$$

where there are two noise elements that keep the true values from being revealed. One term will be removed by the following data node, since the Q_k vectors cannot be known by the central node. The other terms are removed only at the final step by the central node and serve to keep the following data nodes from knowing the true value. Since the running total is only passed back to the central node at the final step, when it removes all the noise it only learns a single number, the true total log likelihood across all the data.

This noisy formulation serves a second purpose of calculating the conditional mean parameter, namely through A_k^1. As mentioned previously, the conditional mean parameters are statistics, depending on information at both the data node and central node levels, so they must be calculated securely in multiple stages. The data nodes can safely pass A_k^1 and A_k^2 back to the central node because they contain noise generated at the node level (R_k and Q_k). This facilitates both the computation of the conditional mean parameter and the eventual de-noising of the log likelihood.

After performing these computations, the first data node passes the objects discussed back to the central node, and they also pass along the noisy log likelihood and noise vectors they generated to the next data node. The noisy log likelihood needs to be passed along to each data node with each noisy total being added before returning to the central node for final de-noising, akin to the process of secure summation. The noise vectors passed to the second data node, R_1 and Q_1, are used both for intermediate de-noising of the log likelihood, removing noise which cannot be computed by the central node, and in finishing the calculation of the conditional mean parameter; see Algorithm 7 for more details.

Before moving on to the second data node, the central node needs to make the next step in computing the conditional mean parameter. This combines information

received from the previous data node with information known only the central node, computing

$$B_1 = \tilde{\mu}_{X_1^+} + \Sigma_{X_1 X_1^+} A_1^1; \tag{15}$$

see Algorithm 6 for more details. This is then passed on to the next data node, along with the conditional covariance parameters ($\Sigma_{X_2 X_2|X_1}$), the noise vector (P_1), and the other element needing for calculating the conditional mean parameter (C_1).

The second data node finalizes the computation of the conditional mean parameter, producing a parameter which is noisy in the same way the original marginal mean parameter ($\tilde{\mu}_{X_1}$) was noisy, with only a noise vector unknown to the data node:

$$\tilde{\mu}_{X_1^+|X_1} = B_1 - C_1(R_1 - P_1) = \hat{\mu}_{X_1^+|X_1} + P_{1^+}. \tag{16}$$

From here, the steps start to repeat and look identical at each node. There is one additional wrinkle, as can be seen in Algorithm 7. From data node 2 onwards, the conditional mean parameters corresponding to the nodes in the following partitions must be passed back to the central node, so that the conditioning stacks. Because the central node knows the noise added to this parameter (P_k), it would be disclosive to return it as is to the central node. Additional noise M_k is generated by the data node and added to prevent this. This noise matrix is also passed along to the next node, so it can be removed in the final step of computing the next conditional mean parameter:

$$\tilde{\mu}_{X_2^+|X_2} = B_2 - M_2 - C_2(R_2 - P_2) = \hat{\mu}_{X_2^+|X_2} + P_{2^+}. \tag{17}$$

After all the nodes have completed the process of obtaining parameters and calculating noisy log likelihoods, the total sum is passed back to the first node, not the central node yet, where it undergoes one final de-noising; see Algorithm 8 for details. This is the same intermediate de-noising that occurs at each data node. It must occur at the first node before returning to the final node, since there is no node following the last node. Finally, the central node receives the noisy value, and knowing all the noise that has accumulated throughout the process is able to remove it and obtain the correct total value; see Algorithm 9 for details.

A simple extension exists for complex partitions by subsetting the routines by horizontally distinct vertical routines and then summing all values at the end. Figure 4 gives a visual depiction of this process. As mentioned earlier, this requires an assumption that the central node knows how the partitions are complexly divided and can appropriately divide the algorithm into subroutines. Each subroutine yields a total log likelihood value as if it were a pure vertical partition, and each total from all subroutines are added to get the total across all partitions. In the next section, we present a simulation study to further evaluate the accuracy and computational efficiency of the proposed algorithm.

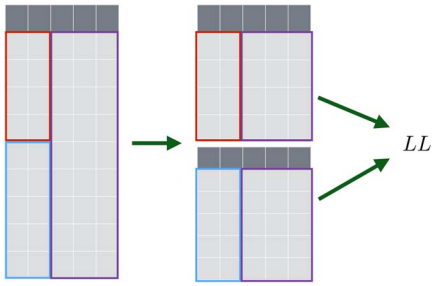

FIG. 4. *Splitting complex partitions into vertical subroutines.*

5. Simulations for accuracy and computational complexity. We ran simulations (available along with code for the algorithm at https://github.com/jsnoke/Firewall) to test the computational complexity of the algorithm as the number of observations (n), variables (p), and partitions (k) increase. For each value of n, p, and k, we generated twenty replicates of data from a multivariate normal distribution with parameters defined by a true latent growth model (LGM) with intercept and slope latent variables, variance and covariance parameters on the latent variables, and a fixed variance parameter across the manifest variables. See Section 6 for a more detailed description of this class of models. We then estimated these true model parameters using our partitioned algorithm and standard SEM software from the *OpenMx* **R** package. Results are plotted in Figure 5 for all replicates. Experiments were run on an Intel Xeon E5-2680 with 20 processors at 2.80 GHz.

We expected the algorithm to scale normally with respect to variables and observations, and we see this in the top two left panels of Figure 5. When we vary the number of observations, each line for a different number of nodes shows the same slope, but shifted from one another. In the case of varying number of variables, again each line shows the same curve, but the optimal point is in a different place due to the relationship between p and k. This implies that our algorithm does not change the computational complexity with respect to n or p. Increasing the number of partitions will not alter the fundamental computational complexity as n or p increase, but it shifts the location of the curve.

We are also interested to see the results as the number of nodes increase. In one respect, the run time should slow because the number of total computations is increasing, but as each node holds fewer variables the covariance inversions will shrink, reducing run time. As we see from the bottom left plot in Figure 5, this tradeoff does exist. When we look at fixed p and n, we again see an optimal number of nodes, $k = 10$, for $p = 100$ and then an uptick after that. Generally though, there is not a significant difference in times for the different number of partitions (apart from 1). It is possible as k gets much larger that this could change, but in applications we envision 100 data parties each holding separate variables is already many more than we would expect.

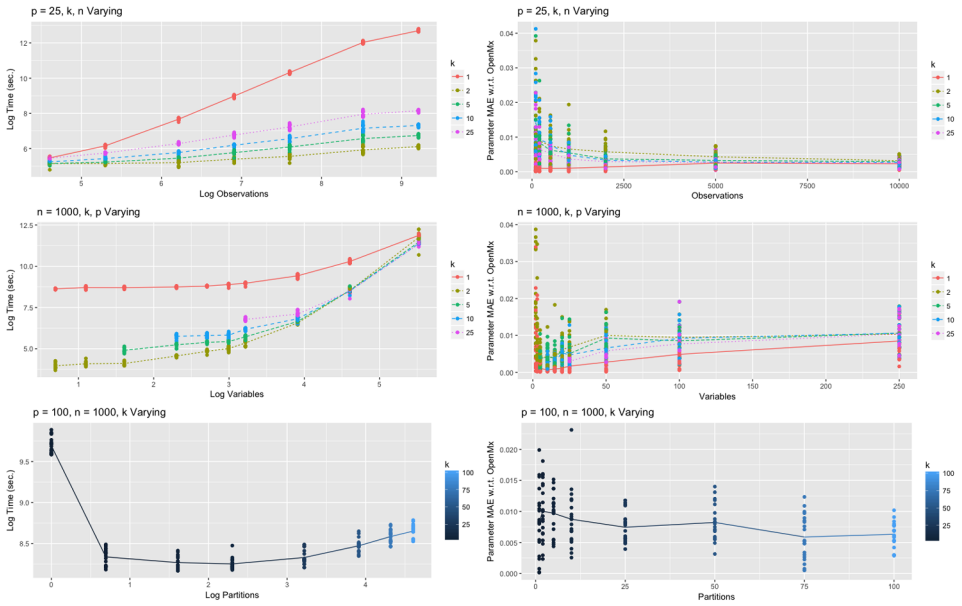

FIG. 5. *Simulations for run time* (*left column*) *and parameter accuracy* (*right column*) *for varying numbers of variables* (p), *observations* (n), *and nodes* (k).

The three right panels show the accuracy of the estimates compared to standard software for nonpartitioned estimates. Theoretically, the estimates should be identical, but we were curious to investigate possible numerical differences in the actual computation. On average, there is less than 0.01 error in value, which is less than 1% error for the scale of the parameters we used. Recall that theoretically there is no difference between the maximum likelihood estimates from the partitioned or nonpartitioned algorithms, so these differences occur due to numerical precision and the software used to find the MLE. For practical purposes, this shows our secure partitioned algorithm produces equivalent results. Next, we address the motivating problem, showing how accurate results can be obtained across a real complex data partition without needing to combine data.

6. Kinship foster placement study. We consider a data set collected over four years (2012–2015) from multiple sources gathered by a trusted research network between the University of Oklahoma Health Sciences Center (OUHSC), the Oklahoma Department of Human Services (OKDHS), and the North Oklahoma County Mental Health Center (NorthCare). Specifically, we examine a study aimed at improving the stability of foster care placements when the foster caregiver was a relative or person known to the family of the child. These are so-called kinship foster placements. In the collaboration, families with kinship foster placements were randomized to receive either (a) services as usual through OKDHS or (b)

OKDHS services and additional services termed Family KINnections from a community resource specialist (CRS) associated with NorthCare. OUHSC conducted the randomization and OUHSC data collectors gathered baseline data on families randomized to services as usual. However, to aid in subsequent service administration the CRS gathered baseline data for families receiving Family KINnections. Research is ongoing, but initial work has shown the program reduced the time for a foster family to become certified and increased the stability of a placement by roughly a factor of two; see Hecht, Hunter and Beasley (2016).

As initially collected, the baseline data (wave 1) were horizontally partitioned into two databases. Waves 2 through 7 of data collection were conducted by an OUHSC data collector for both wave 1 groups, but these later waves were stored in a separate database from the baseline data, giving us a total of three databases complexly partitioned (both horizontally and vertically). For the application here, we model only the Family Needs Scale (FNS) [e.g., see Dunst, Trivette and Deal (1988)]. The FNS measures a family's needs for different resources and support with 41 items such as the extent to which the family has a need for "Having money to pay the bills" or "Getting clothes." The items are scored from 1 ("Almost Never") to 5 ("Almost Always") with higher scores indicating more need.

To investigate whether a family's needs change over time, we use a Latent Growth model (LGM) which is a common way to model a longitudinal trajectory within the SEM framework, see Meredith and Tisak (1990). Algebraically, we write the model as

$$FNS_{ij} = intercept_i + \lambda_j * slope_i + e_i, \tag{18}$$

where i denotes the observation and j denotes the wave. The latent factors are denoted by *intercept* and *slope*, and the λ_j's are fixed at $\{0, 1, \ldots, j-1\}$. We are interested in estimating the covariance matrix for the latent factors (3 parameters), the residual error for the observed variables (1 parameter), and the latent factor means (2 parameters). We assume here the residual error term is fixed across waves and the observed data has mean zero.

Table 1 shows the estimates for the six parameters in the model. We fit the models in three ways to compare estimates and standard errors. The first is with our secure algorithm across the three partitions. For optimization, we used the *optimx* package [Nash and Varadhan (2011)] in **R** [R Core Team (2017)]. The data was partitioned as follows: the first partition held roughly two thirds (158 out of 244) of the observations for the first wave of FNS, the second partition held the other subset of the first wave, and the third partition held all observations for waves 2–7. This gave the data a complex partition, similar to that depicted in Figure 4. The second estimation method was with all data combined also using the *optimx* package, and the third method again used a combined data set but used the *OpenMx* package [Neale et al. (2016)] which utilizes a different optimizer. Recall that the combined data was only possible due to a research data sharing agreement between the data

TABLE 1
Simple Latent Growth Curve Model with three estimation methods and three imputation methods. Standard errors given in parenthesis (NA when model did not return an estimate)

	Parameter	Partitioned	Nonpartitioned	OpenMx
Marginal Imputation	$\hat{\sigma}_{\text{intercept}}$	0.4468 (0.0236)	0.4470 (0.0236)	0.4470 (0.0256)
	$\hat{\sigma}_{\text{intr\_slp}}$	0.0000 (NA)	0.0000 (NA)	0.0000 (0.0062)
	$\hat{\sigma}_{\text{slope}}$	0.0460 (0.0063)	0.0462 (0.0063)	0.0462 (0.0065)
	$\hat{\sigma}_e$	0.3657 (0.0074)	0.3646 (0.0074)	0.3645 (0.0077)
	$\hat{\mu}_{\text{intercept}}$	2.0230 (0.0318)	2.0722 (0.0327)	2.0721 (0.0327)
	$\hat{\mu}_{\text{slope}}$	−0.0809 (0.0059)	−0.0750 (0.0053)	−0.0750 (0.0053)
Joint Imputation	$\hat{\sigma}_{\text{intercept}}$	0.5173 (0.0285)	0.5183 (0.0285)	0.5183 (0.0294)
	$\hat{\sigma}_{\text{intr\_slp}}$	0.0000 (NA)	0.0000 (NA)	0.0000 (0.0104)
	$\hat{\sigma}_{\text{slope}}$	0.0701 (0.0064)	0.0704 (0.0064)	0.0704 (0.0082)
	$\hat{\sigma}_e$	0.3608 (0.0079)	0.3594 (0.0078)	0.3594 (0.0079)
	$\hat{\mu}_{\text{intercept}}$	2.0963 (0.0359)	2.1585 (0.0367)	2.1584 (0.0367)
	$\hat{\mu}_{\text{slope}}$	−0.1050 (0.0069)	−0.0978 (0.0063)	−0.0978 (0.0063)
Full Information Maximum Likelihood	$\hat{\sigma}_{\text{intercept}}$			0.5941 (0.0421)
	$\hat{\sigma}_{\text{intr\_slp}}$			0.0000 (0.0191)
	$\hat{\sigma}_{\text{slope}}$			0.0456 (0.0270)
	$\hat{\sigma}_e$			0.4365 (0.0179)
	$\hat{\mu}_{\text{intercept}}$			2.1097 (0.0483)
	$\hat{\mu}_{\text{slope}}$			−0.0697 (0.0129)

collection agencies. The purpose of having both these nonpartitioned methods was to parse out the two potential causes for difference in estimates: the partitioning and the software.

In addition to the different methods of estimation, we employed three different approaches to handle missing data. As with most real data problems, missingness is an issue in this data, and currently our algorithm only works for complete data. In order to run the secure algorithm without imputing, some parties would need to know the missing data pattern of other parties, which we believe constitutes too much of a privacy risk. Future work should consider ways, if any, to get around this issue, but for this analysis we impute all missing values before estimating parameters.

To handle the missingness in the real data, we employed imputation as is commonly done in behavioral and social science research; see Schafer (1997). The first method, marginal imputation, reflects the type of imputation that would need to be done in a real partitioned setting, since joint imputation requires combining all the data. For comparison purposes, we also employed joint imputation and full-information maximum likelihood (FIML), a method that subsets the mean and covariance when calculating the log likelihood in order to calculate the total log likelihood without imputing or performing listwise deletion. Note that only the *OpenMx* software is capable of performing the FIML approach since this keeps

the missing data. Future work could also consider a type of EM approach that combines our secure estimation with a joint imputation in order to perform joint imputation while preserving the data partitions, potentially akin to what was proposed in Reiter et al. (2004). This extension might improve estimates over the marginal imputation approach.

We see in Table 1 that for a given imputation method, the parameter estimates and standard errors for the different estimation methods are very similar, generally identical up to two decimal places, and for inferential purposes they are equivalent. This follows from the theory that the estimates should be the same for the partitioned and nonpartitioned methods. We do see larger differences across the different imputation methods, particularly FIML versus the imputation methods, which is understandable given the real data has a decent amount of missingness to impute. Sixty-three percent of all data were missing, varying from 28% at the initial wave to 84% at the final wave.

Substantively, we see that family's needs decrease over time, as evidenced by the negative latent slope mean parameter. While this effect is small compared to the inherent baseline need (intercept mean), it does appear significantly negative. From the first wave to the seventh wave, we would expect an average decrease of roughly 0.5 in responses on the FNS.

For further analysis, we considered two hypotheses and tested them using χ^2 goodness-of-fit tests. The first question we tested was whether the slope parameter in the LGM was indeed nonzero. This helps us further investigate whether the decrease we see in the first model is a real change. To test this, we estimated the same model but with the slope parameter fixed to zero. We then compared the reduced model's log likelihood with the larger model to get a χ^2 test statistic. The full parameter results for the reduced model are given in Table 3 in Appendix A. Table 2 gives the hypothesis test results for each of the imputation and estimation approaches. As we can see, there is no substantive difference in the outcomes; all of them significantly reject the hypothesis that the slope parameter is zero. This further confirms the evidence that family's needs decreased over time in this study.

TABLE 2

Results from χ^2 goodness-of-fit test. (1) *Test of slope parameter versus reduced model with slope fixed at zero.* (2) *Test of LGM model with slope versus saturated model. Test statistics shown with p-values in parenthesis*

	Null Hypothesis	**DF**	**Partitioned**	**Nonpartitioned**	**OpenMx**
Marg. Imputation	$\hat{\mu}_{\text{slope}} = 0$	1	727.12 ($\leq$1e$-$6)	735.49 ($\leq$1e$-$6)	151.48 ($\leq$1e$-$6)
Joint Imputation	$\hat{\mu}_{\text{slope}} = 0$	1	673.36 ($\leq$1e$-$6)	683.39 ($\leq$1e$-$6)	169.71 ($\leq$1e$-$6)
FIML	$\hat{\mu}_{\text{slope}} = 0$	1			24.54 ($\leq$1e$-$6)
Marg. Imputation	LGM is true	29	799.57 ($\leq$1e$-$6)	799.57 ($\leq$1e$-$6)	799.95 ($\leq$1e$-$6)
Joint Imputation	LGM is true	29	876.00 ($\leq$1e$-$6)	877.53 ($\leq$1e$-$6)	877.53 ($\leq$1e$-$6)
FIML	LGM is true	29			103.96 ($\leq$1e$-$6)

Next, we considered the overall fit of the models in order to test whether a LGM is an appropriate model for this data. While in the partitioned setting, one must submit a model without seeing the data; we can test our hypothesized model against a saturated model in order to test the overall fit. This type of goodness-of-fit test is common in the SEM framework (among others), and it is a strength of our approach that a researcher can still perform these tests using our secure algorithm.

The saturated model in this case estimates a parameter for every element of the mean and covariance matrices for the observed variables, which results in $n + \frac{n(n+1)}{2} = 35$ parameters. We are less interested in the actual parameter estimates for the saturated model and more interested in the χ^2 test statistic given in Table 2. We see that the saturated model shows significant improvement over our proposed LGM model, and that the test results agree regardless of the estimation or imputation method. While this does not directly contradict the previous evidence we found for a decreasing family need over time, it does make us consider whether a more complex model would be better. It is possible that introducing other covariates or higher order latent growth factors (such as quadratic) would improve the fit. A better fit model would likely give us more accurate parameter estimates and a better understanding of the change in FNS over time.

In terms of the accuracy of the secure algorithm, we find that the differences from the imputation method far outweigh any differences from using a partitioned versus non-partitioned estimation procedure. Interestingly, the goodness-of-fit statistics are quite a bit smaller using the FIML approach for missingness, suggesting that the test is sensitive to the missing data method. For a given imputation method, our method produces roughly equivalent values using partitioned estimation versus nonpartitioned estimation. In practice, our inference will be limited to the estimation based on complete data models, and currently only marginal imputation is possible in the partitioned setting. If the method for handling missing data will have a large impact of the analysis, such as the goodness-of-fit test, that is a potential limitation of the partitioned setup.

7. Discussion. In this paper, we address the growing need of doing estimation over partitioned databases, which cannot be combined due to privacy constraints. By focusing on maximum likelihood, we enable a wide range of models that rely on covariance modeling, allowing researchers to answer questions for numerous applications. This can greatly aid research fields where the current alternatives require building research networks or developing data sharing agreements which can take months if not years. Through the kinship study presented in this paper and further simulations, we showed our theoretical methods translate into accurate inference for real data problems.

We extend previous methods for covariance modeling by strengthening the privacy guarantees, particularly for increasing numbers of partitions, and our method allows estimation in the case of horizontal, vertical, or complex partitions. The

algorithm presented here provides a solution for partitioned estimation when researchers are interested in covariance modeling under a multivariate normal assumption, and it guarantees no sharing of data or intermediate statistics for two or more data parties. Some practical issues still exist, such as the need to impute missingness marginally and the general complexity of the algorithm, but these are details which can and should be further tuned as part of future work. The algorithm contains a fair level of complexity, with numerous steps needed to obtain the correct values without sharing any true information. It may be possible in future work to simplify the algorithm, potentially making it easier to understand or save on computation time. By giving detailed algorithms and making the code easily accessible, we hope to facilitate any future implementations or improvements of this methodology.

For a practical implementation, decisions need to be made concerning the random noise distributions used in the algorithm and the optimization method. Fortunately, these do not fundamentally affect the guarantees, either of security or accuracy, that we present in this paper. Assuming the random noise distributions used by the central and data nodes are nontrivial and not disclosed to other parties, there is reasonable surety that no one would be able to guess the noise added. Generally, the larger the noise added, the more certain this guarantee, and the only constraint would be running up against the machines' numerical precision. The noise is all removed at the final step, so the distributions used will not affect the final result's accuracy. Similarly, our algorithm does not alter the theoretical optimality of any optimization routines. If an optimization method if susceptible to local modes or poor convergence, it will be the case regardless of using the secure algorithm or a nonsecure one.

We note that a general drawback of secure multiparty methods such as ours is that models are somewhat blindly chosen. Though our method provides standard errors and goodness-of-fit statistics in addition to parameter estimates, researchers are still unable to do the typical exploratory data analysis (EDA) or model diagnostics which a thorough statistical analysis demands. For example, in the case of regression the inability to examine residual plots or visualize the underlying variable relationships is a significant drawback. In some cases, privacy restrictions will be such that this is the best we can do, but we caution any potential implementers or users of these methods in this regard.

As a final remark, there has been a move in the privacy literature toward formal privacy methods, such as differential privacy. While this work guarantees security under a different definition, we believe future work should look for ways to combine the methods here with formal protections such as offered by perturbations of the output statistics. In some applications, it may be the case that revealing the true estimated model (or mean and covariance matrix) constitutes a violation of privacy. For problems where that is not the case, the algorithm presented here provides a strong guarantee of security that multiple parties can engage in partitioned estimation with only the final model values being shared.

APPENDIX A: ADDITIONAL MODEL TABLE

Table giving reduced model with slope parameter fixed at zero.

TABLE 3

Latent Growth Curve Model with slope parameter fixed to zero, with three estimation methods and three imputation methods. Standard errors given in parenthesis (NA when model did not return an estimate)

	Parameter	Partitioned	Nonpartitioned	OpenMx
Marginal	$\hat{\sigma}_{\text{intercept}}$	2.1336 (0.0982)	2.1670 (0.1000)	0.4567 (0.0266)
Imputation	$\hat{\sigma}_{\text{intr\_slp}}$	0.0000 (NA)	0.0000 (NA)	0.0000 (0.0118)
	$\hat{\sigma}_{\text{slope}}$	0.0567 (0.0065)	0.0569 (0.0064)	0.0842 (0.0065)
	$\hat{\sigma}_e$	0.3608 (0.0073)	0.3596 (0.0073)	0.3643 (0.0077)
	$\hat{\mu}_{\text{intercept}}$	−0.0715 (0.0063)	−0.0653 (0.0057)	1.9819 (0.0490)
Joint	$\hat{\sigma}_{\text{intercept}}$	2.2548 (0.1035)	2.3021 (0.1062)	0.5662 (0.0234)
Imputation	$\hat{\sigma}_{\text{intr\_slp}}$	0.0000 (NA)	0.0000 (NA)	0.0000 (NA)
	$\hat{\sigma}_{\text{slope}}$	0.0908 (0.0064)	0.0909 (0.0063)	0.1208 (NA)
	$\hat{\sigma}_e$	0.3491 (0.0071)	0.3481 (0.0071)	0.3533 (0.0074)
	$\hat{\mu}_{\text{intercept}}$	−0.0946 (0.0078)	−0.0871 (0.0072)	2.0898 (NA)
Full Information	$\hat{\sigma}_{\text{intercept}}$			0.6078 (0.0434)
Maximum Likelihood	$\hat{\sigma}_{\text{intr\_slp}}$			0.0000 (0.0245)
	$\hat{\sigma}_{\text{slope}}$			0.0851 (0.0195)
	$\hat{\sigma}_e$			0.4292 (0.0181)
	$\hat{\mu}_{\text{intercept}}$			2.0325 (0.0512)

APPENDIX B: NUMERICAL EXAMPLE

This follows step for step Algorithm 2 given in Section 4.3 and visualized in Figure 3 with a simple 3×3 data set.

Input. Central node chooses, based on research model of interest, Σ and μ, model defined parameters, where

$$\Sigma = \begin{bmatrix} \Sigma_{X_1X_1} & \Sigma_{X_2X_1} & \Sigma_{X_3X_1} \\ \Sigma_{X_1X_2} & \Sigma_{X_2X_2} & \Sigma_{X_3X_2} \\ \Sigma_{X_1X_3} & \Sigma_{X_2X_3} & \Sigma_{X_3X_3} \end{bmatrix} = \begin{bmatrix} 1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{bmatrix}, \tag{19}$$

$$\mu = \begin{bmatrix} \mu_{X_1} & \mu_{X_2} & \mu_{X_3} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}. \tag{20}$$

Data node 1 holds X_1, data node 2 holds X_2, data node 3 holds X_3, where

$$Data = Z = \begin{bmatrix} X_1 & X_2 & X_3 \end{bmatrix} = \begin{bmatrix} -0.36 & 1.31 & -0.23 \\ -0.09 & 0.75 & 2.82 \\ -0.92 & 0.43 & -0.64 \end{bmatrix}. \tag{21}$$

B.1. Central node (CN) initiates. The follow objects are generated by CN:

1. $\Sigma_{X_1X_1} = [1]$, $\Sigma_{X_2X_2|X_1} = [0.99]$, $\Sigma_{X_3X_3|X_2,X_1} = [0.9818182]$.
2. $C_1 = \Sigma_{X_1X_1^+}\Sigma^{-1}_{X_1X_1} = \left[\begin{smallmatrix}0.1\\0.1\end{smallmatrix}\right]$, $C_2 = \Sigma_{X_2X_3|X_1}(\Sigma_{X_2X_2|X_1})^{-1} = [0.09090909]$.
3. Generated from random distribution:

$$P_1 = \begin{bmatrix} 65.18644 \\ -20.08849 \\ 135.41011 \end{bmatrix}, \qquad P_2 = \begin{bmatrix} -181.81430 \\ 280.12343 \\ -26.61653 \end{bmatrix}, \qquad P_3 = \begin{bmatrix} -196.07673 \\ 89.11074 \\ -44.19684 \end{bmatrix}.$$

4.

$$\tilde{\mu}_{X_1} = \mu_{X_1} + P_1 = \begin{bmatrix} 65.28644 \\ -19.98849 \\ 135.51011 \end{bmatrix}, \qquad \tilde{\mu}_{X_2} = \mu_{X_2} + P_2 = \begin{bmatrix} -181.71430 \\ 280.22343 \\ -26.51653 \end{bmatrix},$$

$$\tilde{\mu}_{X_3} = \mu_{X_3} + P_3 = \begin{bmatrix} -195.97673 \\ 89.21074 \\ -44.09684 \end{bmatrix}.$$

B.2. Central node passes to data node 1.

1. $\Sigma_{X_1X_1}$.
2. $\tilde{\mu}_{X_1}$.
3. P_3.

B.3. Data node 1 computes. The first data node generates the following matrices:

1. Generated from random distribution:

$$R_1 = \begin{bmatrix} 1494.8524 \\ 1930.3440 \\ 161.8065 \end{bmatrix}, \qquad Q_1 = \begin{bmatrix} 4113.309 & 557.0139 & 964.1046 \end{bmatrix}.$$

2. $A_1^1 = \Sigma^{-1}_{X_1X_1}(X_1 - \tilde{\mu}_{X_1} + R_1)^T = \begin{bmatrix} 1429.206 & 1950.242 & 25.37639 \end{bmatrix}$.
3. $A_1^2 = \Sigma^{-1}_{X_1X_1}(X_1 - \tilde{\mu}_{X_1} - R_1)^T + Q_1 = \begin{bmatrix} 2552.81 & -1353.432 & 665.868 \end{bmatrix}$.
   And then the noisy log likelihood based on its data:
4. $L\tilde{L}_1 = \Sigma_{i=1}^3[1 * \log(2 * \pi) + \log(|\Sigma_{X_1X_1}|) + (X_1 - \tilde{\mu}_{X_1} + R_1)_i\Sigma^{-1}_{X_1X_1}(X_1 - \tilde{\mu}_{X_1} - R_1)_i^T + R_1^T\Sigma^{-1}_{X_1X_1}R_1] = 23{,}324.09$.

B.4. Data node 1 passes to central node.

1. A_1^1.
2. A_1^2.

B.5. Data node 1 passes data node 2.

1. R_1.
2. Q_1.
3. $L\tilde{L}_1$.

B.6. Central node computes. First half of noisy conditional mean parameter:

1. $B_1 = \tilde{\mu}_{X_1^+} + (\Sigma_{X_1 X_1^+} A_1^1)^T = \begin{bmatrix} -38.79370 & -53.05613 \\ 475.24768 & 284.23498 \\ -23.97889 & -41.55920 \end{bmatrix}$.

B.7. Central node passes to data node 2.

1. P_1.
2. B_1.
3. C_1.
4. $\Sigma_{X_2 X_2 | X_1}$.

B.8. Data node 2 computes. Second half of noisy conditional mean parameter and partially de-noised log likelihood:

1. $\tilde{\mu}_{X_1^+ | X_1} = B_1 - C_1(R_1 - P_1) = \begin{bmatrix} -181.76030 & -196.02273 \\ 280.20443 & 89.19174 \\ -26.61853 & -44.19884 \end{bmatrix}$.
2. $L\tilde{L}_1^* = L\tilde{L}_1 - Q_1 P_1 = -364{,}167.8$.

B.9. Data node 2 computes. The second data node generates the following matrices:

1. Generated from random distribution:

$$R_2 = \begin{bmatrix} 214.6229 \\ 860.1230 \\ 1393.1503 \end{bmatrix}, \qquad Q_2 = \begin{bmatrix} 781.3601 & 530.806 & 227.6579 \end{bmatrix}.$$

2.

$$\begin{aligned} A_2^1 &= \Sigma_{X_2 X_2 | X_1}^{-1} (X_2 - \tilde{\mu}_{X_2 | X_1} + R_2)^T \\ &= \begin{bmatrix} 401.7103 & 586.5339 & 1434.544 \end{bmatrix}. \end{aligned}$$

3.

$$\begin{aligned} A_2^2 &= \Sigma_{X_2 X_2 | X_1}^{-1} (X_2 - \tilde{\mu}_{X_2 | X_1} - R_2)^T + Q_2 \\ &= \begin{bmatrix} 749.4888 & -620.2823 & -1152.243 \end{bmatrix}. \end{aligned}$$

4. Generated from random distribution: $M_2 = \begin{bmatrix} 1437.0787 \\ 323.9371 \\ 301.7027 \end{bmatrix}$.

5. $\tilde{\mu}^*_{X_3|X_1} = \tilde{\mu}_{X_3|X_1} + M_2 = \begin{bmatrix} 1241.0559 \\ 413.1288 \\ 257.5039 \end{bmatrix}$.

And then the noisy log likelihood based on its data:

6. $\tilde{LL}_2 = \sum_{i=1}^{3}[1 * \log(2 * \pi) + \log(|\Sigma_{X_2X_2|X_1}|) + (X_2 - \tilde{\mu}_{X_2|X_1} + R_2)_i \times \Sigma^{-1}_{X_2X_2|X_1}(X_2 - \tilde{\mu}_{X_2|X_1} - R_2)_i^T + R_2^T \Sigma^{-1}_{X_2X_2|X_1} R_2] = 387{,}491.9$.

B.10. Data node 2 passes to central node.

1. A_2^1.
2. A_2^2.
3. $\tilde{\mu}^*_{X_3|X_1}$.

B.11. Data node 2 passes to data node 3.

1. R_2.
2. Q_2.
3. $\tilde{LL}_{1+2} = \tilde{LL}_1^* + \tilde{LL}_2 = -250{,}686.5$.
4. M_2.

B.12. Central node computes. First half of noisy conditional mean parameter:

1. $B_2 = \tilde{\mu}^*_{X_3|X_1} + (\Sigma_{X_2X_2^+|X_1} A_2^1)^T = \begin{bmatrix} 1277.2099 \\ 465.9168 \\ 386.6128 \end{bmatrix}$.

B.13. Central node passes to data node 3.

1. P_2.
2. B_2.
3. C_2.
4. $\Sigma_{X_3X_3|X_1,X_2}$.

B.14. Data node 3 computes. Second half of noisy conditional mean parameter and partially de-noised log likelihood:

1. $\tilde{\mu}_{X_3|X_2,X_1} = B_2 - M_2 - C_2(R_2 - P_2) = \begin{bmatrix} -195.90854 \\ 89.25255 \\ -44.15956 \end{bmatrix}$.
2. $\tilde{LL}^*_{1+2} = \tilde{LL}_{1+2} - Q_2 P_2 = -251{,}255.8$.

B.15. Data node 3 computes. The third data node generates the following matrices:

1. Generated from random distribution:

$$R_3 = \begin{bmatrix} 363.1359 \\ 310.8918 \\ 1739.9768 \end{bmatrix}, \qquad Q_3 = \begin{bmatrix} 1848.916 & 1849.285 & 309.7504 \end{bmatrix}.$$

2.

$$A_3^1 = \Sigma_{X_3X_3|X_2,X_1}^{-1}(X_3 - \tilde{\mu}_{X_3|X_2,X_1} + R_3)^T$$
$$= \begin{bmatrix}569.1629 & 228.6159 & 1816.524\end{bmatrix}.$$

3.

$$A_3^2 = \Sigma_{X_3X_3|X_2,X_1}^{-1}(X_3 - \tilde{\mu}_{X_3|X_2,X_1} - R_3)^T + Q_3$$
$$= \begin{bmatrix}1678.358 & 1444.602 & -1418.123\end{bmatrix}.$$

And then the noisy log likelihood based on its data:

4. $\tilde{LL}_3 = \sum_{i=1}^3 [1 * \log(2 * \pi) + \log(|\Sigma_{X_3X_3|X_2,X_1}|) + (X_3 - \tilde{\mu}_{X_3|X_2,X_1} + R_3)_i \times \Sigma_{X_3X_3|X_2,X_1}^{-1}(X_3 - \tilde{\mu}_{X_3|X_2,X_1} - R_3)_i^T + R_3^T \Sigma_{X_3X_3|X_2,X_1}^{-1} R_3] = 48{,}542.58.$

B.16. Data node 3 passes to central node.

1. A_3^1.
2. A_3^2.

B.17. Data node 3 passes to data node 1.

1. Q_3.
2. $\tilde{LL}_{1+2+3} = \tilde{LL}_{1+2}^* + \tilde{LL}_3 = -202{,}713.2.$

B.18. Data node 1 computes. Partially de-noised log likelihood:

1. $\tilde{LL}_{1+2+3}^* = \tilde{LL}_{1+2+3} - Q_3 P_3 = 8715.143.$

B.19. Data node 1 passes to central node.

1. $\tilde{LL}_{1+2+3}^*$.

B.20. Node final de-noising. Let $LL_{\text{Noise}i} = \sum_{i=1}^n [P_i A_i^1 + P_i A_i^2 + P_i \times \Sigma_{X_iX_i}^{-1} P_i]$. The central node computes the following:

1. $LL = \tilde{LL}_{1+2+3}^* + LL_{\text{Noise1}} + LL_{\text{Noise2}} + LL_{\text{Noise3}} = 8715.143 + 364{,}174.6 + -112{,}904.4 + -259{,}957.5 = 27.91202.$

Compare with:

1. $LL = \sum_{i=1}^3 [3 * \log(2 * \pi) + \log(|\Sigma|) + (Z - \mu)_i \Sigma^{-1} (Z - \mu)_i^T = 27.91202.$

This concludes one complete secure calculation of the true log likelihood of a set of parameters given the data. Parameter estimates are obtained by maximizing over this value using an optimizer of choice.

Algorithm 3 Central Node Initiate (*CN_Initiate*)

Input: μ, Σ

Output: $\Sigma_{X_k X_k|X_k^-}$, P_k, $\tilde{\mu}_{X_k}$, C_k $\forall k \in K$

1: **for** k in $1, \ldots, K$ **do**
2: Compute $\Sigma_{X_k X_k|X_k^-}$
3: Generate random $P_k \in \mathbb{R}^{n \times p_k}$
4: Compute $\tilde{\mu}_{X_k} = \mu_{X_k} + P_k$
5: Compute $C_k = \Sigma_{X_k X_k^+|X_k^-}(\Sigma_{X_k X_k|X_k^-})^{-1}$
6: **end for**

APPENDIX C: INTERNAL ALGORITHMS

Algorithms 3 - 9 give the internal functions for Algorithm 2. Reference notation:

- P_k, R_k, Q_k, M_k random noise matrices with dimensions $\mathbb{R}^{n \times p_k}$, $\mathbb{R}^{n \times p_k}$, $\mathbb{R}^{p_k \times n}$, $\mathbb{R}^{n \times p_k}$, respectively.
- $A_k^1 = \Sigma^{-1}_{X_k X_k|X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} + R_k)$.
- $A_k^2 = \Sigma^{-1}_{X_k X_k|X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} - R_k) + Q_k$.
- $B_k = \tilde{\mu}_{X_k^+|X_k^-} + \Sigma_{X_k X_k^+|X_k^-} A_k^1$.
- $C_k = \Sigma_{X_k X_k^+|X_k^-}(\Sigma_{X_k X_k|X_k^-})^{-1}$.
- $\tilde{\mu}_{X_k^+|X_{k+1}^-} = B_k - M_k - C_k(R_k - P_k)$.

Algorithm 4 External Node Computation (*EN_Compute*)

Input: $\tilde{\mu}_{X_k|X_k^-}$, $\Sigma_{X_k X_k|X_k^-}$, X_k, $\tilde{L}L^*_{\sum_{j=1}^{k-1} j}$

Output: $\tilde{L}L_{\sum_{j=1}^{k} j}$, A_k^1, A_k^2, R_k, Q_k

1: Generate random $R_k \in \mathbb{R}^{n \times p_k}$
2: Generate random $Q_k \in \mathbb{R}^{p_k \times n}$
3: Compute $A_k^1 = \Sigma^{-1}_{X_k X_k|X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} + R_k)$
4: Compute $A_k^2 = \Sigma^{-1}_{X_k X_k|X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} - R_k) + Q_k$
5: Compute $\tilde{L}L_k = computeNoisyLL(\tilde{\mu}_{X_k|X_k^-}, \Sigma_{X_k X_k|X_k^-}, X_k, R_k)$
6: **if** $\tilde{L}L^*_{\sum_{j=1}^{k-1} j} != \varnothing$ **then**
7: $\tilde{L}L_{\sum_{j=1}^{k} j} = \tilde{L}L^*_{\sum_{j=1}^{k-1} j} + \tilde{L}L_k$
8: **else**
9: $\tilde{L}L_{\sum_{j=1}^{k} j} = \tilde{L}L_k$
10: **end if**

Algorithm 5 Compute Noisy LL (*computeNoisyLL*)

Input: $\tilde{\mu}_{X_k|X_k^-}, \Sigma_{X_kX_k|X_k^-}, X_k, R_k$

Output: $\tilde{L}L_k$

1: Compute $\tilde{L}L_k = \sum_{i=1}^{n}[p_k \log(2\pi) + \log(|\Sigma_{X_kX_k|X_k^-}|) + (X_{ki} - \tilde{\mu}_{X_k|X_k^-} + R_k)\Sigma^{-1}_{X_kX_k}(X_{ki} - \tilde{\mu}_{X_k|X_k^-} - R_k)^T + R_k^T\Sigma^{-1}_{X_kX_k|X_k^-}R_k] = LL_k - A_k^1P_k^T - P_k(A_k^2)^T - P_k\Sigma^{-1}_{X_kX_k}P_k^T + P_kQ_k$

Algorithm 6 Central Node Adjustment (*CN_Adjust*)

Input: $A_k^1, \tilde{\mu}_{X_k^+|X_k^-}, \Sigma_{X_kX_k|X_k^-}$

Output: B_k

1: Compute $B_k = \tilde{\mu}_{X_k^+|X_k^-} + (\Sigma_{X_kX_k^+|X_k^-}A_k^1)^T$

Algorithm 7 External Node Adjustment (*EN_Adjust*)

Input: $B_k, C_k, R_k, P_k, Q_k, \tilde{L}L_{\sum_{j=1}^{k} j}, M_k$

Output: $\tilde{L}L^*_{\sum_{j=1}^{k} j}, \tilde{\mu}_{X_k^+|X_{k+1}^-}, \tilde{\mu}^*_{X_{k+1}^+|X_{k+1}^-}, M_{k+1}$

1: Compute $\tilde{L}L^*_{\sum_{j=1}^{k} j} = \tilde{L}L_{\sum_{j=1}^{k} j} - P_kQ_k^T$
2: **if** $M_k != \varnothing$ **then**
3: Compute $\tilde{\mu}_{X_k^+|X_{k+1}^-} = B_k - M_k - C_k(R_k - P_k)$
4: **else**
5: Compute $\tilde{\mu}_{X_k^+|X_{k+1}^-} = B_k - C_k(R_k - P_k)$
6: **end if**
7: Generate $M_{k+1} \in \mathbb{R}^{n\times p_k}$
8: Compute $\tilde{\mu}^*_{X_{k+1}^+|X_{k+1}^-} = \tilde{\mu}_{X_{k+1}^+|X_{k+1}^-} + M_{k+1}$

Algorithm 8 First Node Adjustment (*FN_Adjust*)

Input: $\tilde{L}L_{\sum_{j=1}^{K} j}, P_K, Q_K$

Output: $\tilde{L}L^*_{\sum_{j=1}^{K} j}$

1: Compute $\tilde{L}L^*_{\sum_{j=1}^{K} j} = \tilde{L}L_{\sum_{j=1}^{K} j} - P_KQ_K^T$

Algorithm 9 Central Node Final De-noising (*CN_Final*)

Input: $\tilde{L}L^*_{\sum_{j=1}^K j}$, P_k, A_k^1, A_k^2, $\Sigma_{X_k X_k | X_k^-}$ $\forall k \in K$

Output: $LL_{\sum_{j=1}^K j}$

1: **for** k in $1, \ldots, K$ **do**

2: Compute $LL_{\text{Noise}k} = \sum_{i=1}^n [A_k^1 P_k^T + P_k (A_k^2)^T + P_k \Sigma^{-1}_{X_k X_k | X_k^-} P_k^T]$

3: **end for**

4: Compute $LL_{\sum_{j=1}^K j} = \tilde{L}L^*_{\sum_{j=1}^K j} + \sum_{j=1}^K LL_{\text{Noise}j}$

APPENDIX D: ALGORITHMS FROM PREVIOUS WORK

Algorithms 10 - 12 detail previous methods. For Algorithm 10, suppose we have two parties, with data $X_1 \in \mathbb{R}^{n \times p_1}$ and $X_2 \in \mathbb{R}^{n \times p_2}$, who wish to securely compute the off-diagonal elements of the covariance matrix $X_1^T X_2$.

Algorithm 10 Secure Matrix Multiplication [Karr et al. (2009)]

Input: $X_1 \in \mathbb{R}^{n \times p_1}$ and $X_2 \in \mathbb{R}^{n \times p_2}$ held by party 1 and 2 respectively

Output: $X_1^T X_2$

1: Party 1 generates an orthonormal matrix $Z \in \mathbb{R}^{n \times a}$, such that $Z_i^T X_{1j} = 0 \; \forall i, j$ (columns)

2: Party 1 sends Z to party 2

3: Party 2 computes $W = (I - ZZ^T) X_2$ where $I \in \mathbb{R}^{n \times n}$ is the identity matrix

4: Party 2 sends W to party 1

5: Party 1 computes $X_1^T W = X_1^T (I - ZZ^T) X_2 = X_1^T X_2$

6: (Optional) Party 1 sends $X_1^T X_2$ to party 2

Algorithm 11 Exponential family likelihood by oblivious transfer [Lin and Karr (2010)]

Input: Data matrices X_1, X_2 held by parties 1 and 2 respectively

Output: $\hat{\theta} = \arg\max_\theta a(\theta)^T \sum_{i=1}^n t(X_i) - nc(\theta)$.

1: Party 1 generates a vector W of length s, one component of which is X_{1i}, and the other $s - 1$ of which are random, and sends it to party 2

2: Party 2 computes $t(W_1, X_{2i}) \ldots t(W_s, X_{2i})$, generates a random value ϵ_i, and calculates $t(W_1, X_{2i}) - \epsilon_i \ldots t(W_s, X_{2i}) - \epsilon_i$

3: Party 1 obtains $t(X_{1i}, X_{2i}) - \epsilon_i$ from these using 1 out of s oblivious transfer [Di Crescenzo, Malkin and Ostrovsky (2000)]

4: Party 1 holds $\Sigma_i [t(X_{1i}, X_{2i}) - \epsilon_i]$ and party 2 holds $\Sigma_i \epsilon_i$, which add to $\sum_{i=1}^n t(X_{1i}, X_{2i})$

Algorithm 12 Nonsecure Passing Algorithm [Snoke, Brick and Slavković (2016)]

Input: μ, Σ (central node), $X_k \in \mathbb{R}^{n \times p_k}$ $\forall k \in K$ (data nodes)
Output: $LL_{\sum_{j=1}^{K} j}$

CN $\to$ DN_1: Σ, μ
DN_1 compute: $\Sigma_{X_1 X_1}$, μ_{X_1}, $\Sigma_{X_1^+ X_1^+ | X_1}$, $\hat{\mu}_{X_1^+ | X_1}$
DN_1 compute: LL_1
$DN_1 \to DN_2$: LL_1, $\Sigma_{X_1^+ X_1^+ | X_1}$, $\hat{\mu}_{X_1^+ | X_1}$
for k in $2, \ldots, (K-1)$ **do**
    DN_k compute: $\Sigma_{X_k X_k | X_k^-}$, $\hat{\mu}_{X_k | X_k^-}$, $\Sigma_{X_k^+ X_k^+ | X_{k+1}^-}$, $\hat{\mu}_{X_k^+ | X_{k+1}^-}$
    DN_k compute: LL_k
    $DN_k \to DN_{k+1}$: $LL_{\Sigma_{j=1}^{k} j}$, $\Sigma_{X_k^+ X_k^+ | X_{k+1}^-}$, $\hat{\mu}_{X_k^+ | X_{k+1}^-}$
end for
DN_K compute: $\Sigma_{X_K X_K | X_K^-}$, $\hat{\mu}_{X_K | X_K^-}$
DN_K compute: LL_K
$DN_K \to CN$: $LL_{\sum_{j=1}^{K} j}$

APPENDIX E: DATA LEAKAGE EVALUATION

We acknowledge an asymmetry among the objects received by the different nodes. By showing that none of the nodes receive disclosive statistics, we make this asymmetry irrelevant.

E.1. Node C.

E.1.1. *Starting objects*. Node C holds Σ and μ (model defined parameters not statistics).

E.1.2. *Received objects*.

- A_k^1 for $k \in 1, \ldots, K$.
- A_k^2 for $k \in 1, \ldots, K$.
- $\tilde{\mu}^*_{X_k^+ | X_k^-}$ for $k \in 2, \ldots, K-1$.
- $\tilde{LL}^*_{\sum_{j=1}^{K} j}$.

E.1.3. *Analysis*. The central node receives A_k^1 and A_k^2 for $i \in 1, \ldots, K$, such that

$$
\begin{aligned}
A_k^1 &= \Sigma^{-1}_{X_k X_k | X_k^-}(X_k - \tilde{\mu}_{X_k | X_k^-} + R_k) \\
&= \Sigma^{-1}_{X_k X_k | X_k^-}(X_k - \mu_{X_k | X_k^-} - P_k + R_k), \qquad (22)
\end{aligned}
$$

$$A_k^2 = \Sigma^{-1}_{X_k X_k | X_k^-}(X_k - \tilde{\mu}_{X_k|X_k^-} - R_k) + Q_k$$
$$= \Sigma^{-1}_{X_k X_k | X_k^-}(X_k - \mu_{X_k|X_k^-} - P_k - R_k) + Q_k.$$

Clearly, it is important that X_k should not be recovered, and for $k > 1$, $\tilde{\mu}_{X_k|X_k^-}$ is also risky because it is a statistic. Importantly, the central node knows $\Sigma^{-1}_{X_k X_k|X_k^-}$ and P_k, so these can be differenced out. To protect disclosure, R_k and Q_k are random noise added which the central node does not know. R_k protects the values in A_k^1 and Q_k protects from learning R_k by differencing the two equations (both A_1 and A_2 are needed to calculate the correct log likelihood).

Next, the central node receives $\tilde{\mu}^*_{X_k^+|X_{k-1}}$ for $k \in 2, \ldots, K-1$ such that

$$\tilde{\mu}^*_{X_k^+|X_k^-} = \tilde{\mu}_{X_k^+|X_k^-} + M_k \tag{23}$$
$$= \mu_{X_k^+|X_k^-} + P_k + M_k.$$

The central node knows P_k, so M_k is essential here to protect the true value of the conditional mean statistic. With that, there is no way for the central node to recover it.

Lastly, the central node receives $\tilde{LL}^*_{\sum_{j=1}^K j}$. They can remove all the noise (as we want) to get the true total log likelihood, $LL_{\sum_{j=1}^K j}$, since this is one value composed across the entire set of partitioned databases. As assumed, this total is not risky and is necessary to obtain accurate estimates.

E.2. Node 1.

E.2.1. *Starting objects.* Node 1 holds X_1.

E.2.2. *Received objects.*

- $\Sigma_{X_1 X_1}$.
- $\tilde{\mu}_{X_1}$.
- P_K.
- $\tilde{LL}_{\sum_{j=1}^K j}$.
- Q_K.

E.2.3. *Analysis.* The only object received by Node 1 that is conditioned on other nodes' data and potentially disclosive is $\tilde{LL}_{\sum_{j=1}^K j}$.

Fortunately, this value is heavily perturbed with a variety of noise that Node 1 cannot access. Another note is that Node 1 could potentially estimate the value of P_1, since $\tilde{\mu}_{X_1} = \mu_{X_1} + P_1$ and the sample mean of Node 1's data should converge to μ_{X_1}. Again though, this value is nondisclosive, since there is not anything Node 1 can learn from having P_1.

E.3. Nodes 2 through K.

E.3.1. *Starting objects.* Node k holds X_k:

- $\Sigma_{X_k X_k|k^-}$.
- B_{k-1}.
- C_{k-1}.
- P_{k-1}.
- $\tilde{LL}_{\Sigma_{j=1}^{k-1} j}$.
- R_{k-1}.
- Q_{k-1}.
- M_{k-1} ($M_1 = 0$).

E.3.2. *Analysis.* There is an asymmetry among nodes, since Nodes 2 and following receive more information than Node 1. That being said, if none of it is actually disclosive the asymmetry is acceptable.

The first objects received that are potentially disclosive are B_{k-1} and C_{k-1} such that

$$\begin{aligned} B_{k-1} &= \tilde{\mu}_{X_{k-1}^+|x_{k-1}^-} + \Sigma_{X_{k-1}X_{k-1}^+|x_{k-1}^-} A_{k-1}^1 \\ &= \mu_{X_{k-1}^+|x_{k-1}^-} + \begin{pmatrix} P_k & P_{k+1} & \cdots & P_K \end{pmatrix} \\ &\quad + \Sigma_{X_{k-1}X_{k-1}^+|x_{k-1}^-} (\Sigma_{X_{k-1}X_{k-1}|x_{k-1}^-})^{-1} \\ &\quad \times (X_{k-1} - \tilde{\mu}_{X_{k-1}|x_{k-1}^-} - P_{k-1} + R_{k-1}), \end{aligned} \tag{24}$$

$$C_{k-1} = \Sigma_{X_{k-1}X_{k-1}^+|x_{k-1}^-} (\Sigma_{X_{k-1}X_{k-1}|x_{k-1}^-})^{-1}. \tag{25}$$

Node k is able to remove some of the noise, since

$$\begin{aligned} &B_{k-1} - C_{k-1}(R_{k-1} - P_{k-1}) \\ &\quad = \tilde{\mu}_{X_{k-1}^+} + \Sigma_{X_{k-1}X_{k-1}^+|x_{k-1}^-} \Sigma^{-1}_{X_{k-1}X_{k-1}|x_{k-1}^-} (X_{k-1} - \tilde{\mu}_{X_{k-1}|x_{k-1}^-}). \end{aligned}$$

The key is that the noise $\begin{pmatrix} P_k & P_{k+1} & \cdots & P_K \end{pmatrix}$ is still present to protect the true value of the data and parameters. More importantly, since Node k does not know any of the other individual components (apart from P_{k-1} and R_{k-1}), it cannot decompose B_{k-1} or C_{k-1} further.

C_{k-1} is unperturbed, but two things convince us it is okay to share these. First, it is a product of two matrices, neither of which Node k knows. Second, these are low risk objects. It is possible Node k learns the scale of Node $(k-1)$'s data, but for now we consider that acceptable.

Lastly, Node k receives $\tilde{LL}_{\Sigma_{j=1}^{k-1} j}$, a noisy version of the running total log likelihood. Node k can remove some noise based on P_{k-1} and Q_{k-1} (which we want), but does not know A_{k-1}^1 or A_{k-1}^2, and thus cannot recover the true LL value.

REFERENCES

ARBUCKLE, J. L., MARCOULIDES, G. A. and SCHUMACKER, R. E. (1996). Full information estimation in the presence of incomplete data. *Adv. Struct. Equ. Model. Issues Techn.* **243** 277.

BOKER, S. M., BRICK, T. R., PRITIKIN, J. N., WANG, Y., OERTZEN, T. V., BROWN, D., LACH, J., ESTABROOK, R., HUNTER, M. D., MAES, H. H. and NEALE, M. C. (2015). Maintained Individual Data Distributed Likelihood Estimation (MIDDLE). *Multivar. Behav. Res.* **50** 706–720.

CALANDRINO, J. A., KILZER, A., NARAYANAN, A., FELTEN, E. W. and SHMATIKOV, V. (2011). "You might also like:" privacy risks of collaborative filtering. In *Security and Privacy* (*SP*), 2011 *IEEE Symposium on* 231–246. IEEE.

DE MONTJOYE, Y.-A., SHMUELI, E., WANG, S. S. and PENTLAND, A. S. (2014). Openpds: Protecting the privacy of metadata through safeanswers. *PLoS ONE* **9** e98790.

DI CRESCENZO, G., MALKIN, T. and OSTROVSKY, R. (2000). Single database private information retrieval implies oblivious transfer. In *Advances in Cryptology—EUROCRYPT* 2000 (*Bruges*). *Lecture Notes in Computer Science* **1807** 122–138. Springer, Berlin. MR1772023

DINUR, I. and NISSIM, K. (2003). Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* 202–210. ACM.

DUNST, C. J., TRIVETTE, C. M. and DEAL, A. G. (1988). *Enabling and Empowering Families*: *Principles and Guidelines for Practice*. Brookline Books, Cambridge, MA.

DWORK, C. (2008). Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*. *Lecture Notes in Computer Science* **4978** 1–19. Springer, Berlin. MR2472670

FIENBERG, S. E., NARDI, Y. and SLAVKOVIĆ, A. B. (2009). Valid statistical analysis for logistic regression with multiple sources. In *Protecting Persons While Protecting the People* 82–94. Springer.

FIENBERG, S. E. and SLAVKOVIĆ, A. B. (2011). Data privacy and confidentiality. In *International Encyclopedia of Statistical Science* 342–345. Springer.

FIENBERG, S. E., FULP, W. J., SLAVKOVIC, A. B. and WROBEL, T. A. (2006). "Secure" log-linear and logistic regression analysis of distributed databases. In *Privacy in Statistical Databases* 277–290. Springer.

GAYE, A., MARCON, Y., ISAEVA, J., LAFLAMME, P., TURNER, A., JONES, E. M., MINION, J., BOYD, A. W., NEWBY, C. J., NUOTIO, M.-L., WILSON, R., BUTTERS, O., MURTAGH, B., DEMIR, I., DOIRON, D., GIEPMANS, L., WALLACE, S. E., BUDIN-LJØSNE, I., SCHMIDT, C. O., BOFFETTA, P., BONIOL, M., BOTA, M., CARTER, K. W., DEKLERK, N., DIBBEN, C., FRANCIS, R. W., HIEKKALINNA, T., HVEEM, K., KVALØY, K., MILLAR, S., PERRY, I. J., PETERS, A., PHILLIPS, C. M., POPHAM, F., RAAB, G., REISCHL, E., SHEEHAN, N., WALDENBERGER, M., PEROLA, M., VAN DEN HEUVEL, E., MACLEOD, J., KNOPPERS, B. M., STOLK, R. P., FORTIER, I., HARRIS, J. R., WOFFENBUTTEL, B. H. R., MURTAGH, M. J., FERRETTI, V. and BURTON, P. R. (2014). DataSHIELD: Taking the analysis to the data, not the data to the analysis. *Int. J. Epidemiol.* **43** 1929–1944.

GHOSH, J., REITER, J. P. and KARR, A. F. (2007). Secure computation with horizontally partitioned data using adaptive regression splines. *Comput. Statist. Data Anal.* **51** 5813–5820. MR2407679

GOLDWASSER, S. (1997). Multi party computations: Past and present. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing* 1–6. ACM.

HAYNSWORTH, E. V. (1968). On the Schur complement. Technical Report, DTIC Document.

HECHT, D. B., HUNTER, M. D. and BEASLEY, L. O. (2016). Family KINnections: A Kinship Navigation Program. Presented to the University of Oklahoma Health Sciences Center Department of Pediatrics Section of Developmental and Behavioral Pediatrics at the Section Research Meeting.

HOMER, N., SZELINGER, S., REDMAN, M., DUGGAN, D., TEMBE, W., MUEHLING, J., PEARSON, J. V., STEPHAN, D. A., NELSON, S. F. and CRAIG, D. W. (2008). Resolving individuals

contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genet*. **4** e1000167.

HUNDEPOOL, A., DOMINGO-FERRER, J., FRANCONI, L., GIESSING, S., NORDHOLT, E. S., SPICER, K. and DE WOLF, P.-P. (2012). *Statistical Disclosure Control*. *Wiley Series in Survey Methodology*. Wiley, Chichester. MR3026260

KARR, A. F., LIN, X., SANIL, A. P. and REITER, J. P. (2005). Secure regression on distributed databases. *J. Comput. Graph. Statist*. **14** 263–279. MR2160813

KARR, A. F., FULP, W. J., VERA, F., YOUNG, S. S., LIN, X. and REITER, J. P. (2007). Secure, privacy-preserving analysis of distributed databases. *Technometrics* **49** 335–345. MR2408637

KARR, A. F., LIN, X., SANIL, A. P. and REITER, J. P. (2009). Privacy-preserving analysis of vertically partitioned data using secure matrix products. *J. Off. Stat*. **25** 125.

KISSNER, L. and SONG, D. (2005). Privacy-preserving set operations. In *Advances in Cryptology—CRYPTO* 2005. *Lecture Notes in Computer Science* **3621** 241–257. Springer, Berlin. MR2237310

LIN, X. and KARR, A. F. (2010). Privacy-preserving maximum likelihood estimation for distributed data. *J. Priv. Confid*. **1** 6.

LINDELL, Y. and PINKAS, B. (2009). Secure multiparty computation for privacy-preserving data mining. *J. Priv. Confid*. **1** 5.

MEREDITH, W. and TISAK, J. (1990). Latent curve analysis. *Psychometrika* **55** 107–122.

NARDI, Y., FIENBERG, S. E. and HALL, R. J. (2012). Achieving both valid and secure logistic regression analysis on aggregated data from different private sources. *J. Priv. Confid*. **4** 9.

NASH, J. C. and VARADHAN, R. (2011). Unifying optimization algorithms to aid software system users: optimx for R. *J. Stat. Softw*. **43** 1–14.

NEALE, M. C., HUNTER, M. D., PRITIKIN, J. N., ZAHERY, M., BRICK, T. R., KIRKPATRICK, R. M., ESTABROOK, R., BATES, T. C., MAES, H. H. and BOKER, S. M. (2016). OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika* **81** 535–549. MR3505378

R CORE TEAM (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

RAGHUNATHAN, T. E., REITER, J. P. and RUBIN, D. B. (2003). Multiple imputation for statistical disclosure limitation. *J. Off. Stat*. **19** 1–17.

REITER, J. P., KOHNEN, C. N., KARR, A. F., LIN, X. and SANIL, A. P. (2004). Partitioned, Vertically and Data, Partially Overlapping. Technical Report, NISS. Available at https://www.niss.org/sites/default/files/technicalreports/tr146.pdf.

SAMIZO, Y. (2016). Secure statistical analyses on vertically distributed databases. Master's thesis, The Pennsylvania State Univ.

SANIL, A. P., KARR, A. F., LIN, X. and REITER, J. P. (2004). Privacy preserving regression modelling via distributed computation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 677–682. ACM.

SAVAGE, C. J. and VICKERS, A. J. (2009). Empirical study of data sharing by authors publishing in PLoS journals. *PLoS ONE* **4** e7078. DOI:10.1371/journal.pone.0007078.

SCHAFER, J. L. (1997). *Analysis of Incomplete Multivariate Data*. *Monographs on Statistics and Applied Probability* **72**. Chapman & Hall, London. MR1692799

SCHUR, I. (1905). Neue Begründung der Theorie der Gruppencharaktere. *Sitzungsberichte Königl. Preuss. Akad. Wiss*. 406–432.

SLAVKOVIC, A. B., NARDI, Y. and TIBBITS, M. M. (2007). "Secure" logistic regression of horizontally and vertically partitioned distributed databases. In *Data Mining Workshop*, 2007, *ICDM Workshops* 2007, *Seventh IEEE International Conference on Data Mining* 723–728.

SNOKE, J., BRICK, T. and SLAVKOVIĆ, A. (2016). Accurate estimation of structural equation models with remote partitioned data. In *International Conference on Privacy in Statistical Databases* 190–209. Springer.

SULLIVAN, C. M. (1992). *An Overview of Disclosure Principles*. Bureau of the Census.

VAIDYA, J. and CLIFTON, C. (2004). Privacy preserving naïve Bayes classifier for vertically partitioned data. In *Proceedings of the Fourth SIAM International Conference on Data Mining* 522–526. SIAM, Philadelphia, PA. MR2388481

VAIDYA, J., CLIFTON, C., KANTARCIOGLU, M. and PATTERSON, A. S. (2008). Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data* **2** 14.

WILLENBORG, L. and DE WAAL, T. (2001). *Elements of Statistical Disclosure Control*. *Lecture Notes in Statistics* **155**. Springer, New York. MR1866909

YAO, A. C. (1982). Protocols for secure computations. In 23*rd Annual Symposium on Foundations of Computer Science* (*Chicago*, *IL*, 1982) 160–164. IEEE, New York. MR0780394

J. SNOKE
A. SLAVKOVIĆ
STATISTICS DEPARTMENT
PENNSYLVANIA STATE UNIVERSITY
UNIVERSITY PARK, PENNSYLVANIA 16802
USA
E-MAIL: snoke@psu.edu
sesa@psu.edu

T. R. BRICK
HUMAN DEVELOPMENT
AND FAMILY STUDIES DEPARTMENT
PENNSYLVANIA STATE UNIVERSITY
UNIVERSITY PARK, PENNSYLVANIA 16802
USA
E-MAIL: tbrick@psu.edu

M. D. HUNTER
DEPARTMENT OF PEDIATRICS
UNIVERSITY OF OKLAHOMA HEALTH SCIENCES CENTER
OKLAHOMA CITY, OKLAHOMA 73104
USA
E-MAIL: mhunter1@ouhsc.edu