DATA SCIENCE JOURNAL

Alter, G, et al. 2018. Computing Statistics from Private Data. *Data Science Journal*, 17: 31, pp. 1–16. DOI: https://doi.org/10.5334/dsj-2018-031

RESEARCH PAPER

Computing Statistics from Private Data

George Alter[1], Brett Hemenway Falk[2], Steve Lu[3] and Rafail Ostrovsky[4]

[1] Inter-University Consortium for Political Science Research (ICPSR), University of Michigan, US

[2] Computer and Information Science Department, University of Pennsylvania, US

[3] Stealth Software Technologies Inc., US

[4] UCLA, work partially done while consulting for Stealth Software Technologies Inc., US

Corresponding author: Brett Hemenway Falk (fbrett@cis.upenn.edu)

In several domains, privacy presents a significant obstacle to scientific and analytic research, and limits the economic, social, health and scholastic benefits that could be derived from such research. These concerns stem from the need for privacy about personally identifiable information (PII), commercial intellectual property, and other types of information. For example, businesses, researchers, and policymakers may benefit by analyzing aggregate information about markets, but individual companies may not be willing to reveal information about risks, strategies, and weaknesses that could be exploited by competitors. Extracting valuable utility from the new "big data" economy demands new privacy technologies to overcome barriers that impede sensitive data from being aggregated and analyzed.

Secure multiparty computation (MPC) is a collection of cryptographic technologies that can be used to effectively cope with some of these obstacles, and provide a new means of allowing researchers to coordinate and analyze sensitive data collections, obviating the need for data-owners to share the underlying data sets with other researchers or with each other. This paper outlines the findings that were made during interdisciplinary workshops that examined potential applications of MPC to data in the social and health sciences.

The primary goals of this work are to describe the computational needs of these disciplines and to develop a specific roadmap for selecting efficient algorithms and protocols that can be used as a starting point for interdisciplinary projects between cryptographers and data scientists.

Keywords: data; cryptography; privacy; federated data sets; distributed computing; encryption

I. Introduction: The need for statistics from private data

Tension between research and privacy: In many arenas privacy concerns pose a significant barrier to scientific research and any ensuing economic, social and health benefits. Many privacy concerns stem from personally identifiable information (PII). Detailed health records can help doctors and researchers improve diagnosis and treatment, but at an individual level, these records contain extremely sensitive personal information. Similarly, data-sets concerning student performance can help researchers and teachers identify and implement successful pedagogical techniques, but the underlying data (which may contain detailed information about children) must remain private. In other scenarios, privacy concerns may be economic rather than personal. Businesses may stand to gain by analyzing aggregate information about their markets, customers, risks etc., but the underlying data may contain information about each company's costs, strategies and weaknesses that could be exploited by competitors. For example, satellite operators often experience "anomalies" which indicate a malfunction or complete loss of function of the satellite. Diagnosing these anomalies is time and resource intensive, and the diagnosis could be streamlined if information were shared between operators. Operators are unwilling to share anomaly information, however, as it often indicates a business weakness that could be exploited by competitors (Galvan et al. 2014). In all these situations privacy considerations pose obstacles that increase the cost, slow the pace, or completely prevent using the data to their fullest.

These privacy concerns are only intensified as improved technology allows us to collect, store, transmit and process data at an ever-increasing scale. But these technological improvements have done little to address privacy concerns, and providing researchers with a means of securely and efficiently analyzing sensitive data remains a problem. Extracting maximum utility from the new "big data" economy requires new techniques to overcome privacy barriers that prevent sensitive data from being aggregated and analyzed.

Secure multiparty computation (MPC) is a cryptographic tool that can be used to overcome some of these obstacles, and provides a means of allowing researchers to securely merge and analyze sensitive data sets, without requiring data-owners to share the underlying data sets with researchers or each other.

To illustrate the power of MPC, we consider a simple, idealized example. Suppose a researcher wishes to perform statistical tests on patient data held by multiple healthcare providers. In some situations, the researcher may be able to negotiate a data-use agreement, allowing her access to each of the data sets in question, but when privacy considerations prevent the data owners from sharing their data, there may be no way to aggregate all the data into a single data set accessible to the researcher. Using MPC, the problem can be resolved as follows: the researcher and the data owners will engage in a cryptographic protocol, exchanging enciphered messages, with the guarantee that at the end of the protocol, the researcher will learn the result of her statistical query (e.g. regression coefficients) as applied to the aggregation of the data held by all the data owners, *and nothing more*. In this scenario, the data owners never need to share the underlying data sets with anyone, instead cryptography provides a means of *computing* on private data without requiring data-sharing.

In this report, we describe how existing MPC protocols work, and examine how these general purpose cryptographic protocols can be applied to concrete problems of computing statistics on private data.

The need for real-time analysis: In addition to its strong privacy guarantees, MPC has other benefits that improve upon existing methods for securely analyzing sensitive data. One example is the need for real-time analysis. Currently, it can be difficult to use sensitive data to provide real-time analytics to support decision making. A traditional workflow might involve a researcher requesting sensitive data from a data owner, the data owner would then check the credentials of the researcher, anonymize and sanitize the data, ship the sanitized data to the researcher, possibly review any summary statistics produced by the researcher to determine whether they pose a disclosure risk. Each of these steps may require human interaction, and the entire process can take weeks or months from the time when the researcher makes a request to the time the sanitized data are received. This lag time eliminates the possibility of using these data for real-time decision making. MPC can streamline this entire process. Because using MPC for data analytics does not require sharing the underlying data sets, data owners will not have to anonymize or de-identify their data before engaging in an MPC protocol. (Data owners will, however, still be required to engage in non-privacy related data cleaning procedures, e.g. outlier removal). Because MPC protocols are completely automatic, there is no need for human interaction in the process, other than to agree on the statistics being computed.

Reproducibility and transparency: Open science requires that researchers make their methods and data public so that results and conclusions can be independently verified and validated. This notion of open science is at odds with privacy considerations, since a researcher, who has been granted access to a sensitive data set, cannot simply publish the underlying data sets for others to see. MPC provides an alternative means of transparency. Through the interface of MPC, data owners can provide other researchers with a secure means of performing statistics on the sensitive data without sharing the underlying data themselves. This type of secure interface could make it easier for researchers to reproduce, validate and extend each other's work.

Understanding MPC – a cryptographic replacement for a "trusted broker": In many scenarios, privacy obstacles can be overcome if an external broker, trusted by all stakeholders, can be found. The trusted broker will aggregate the private data, perform computations, and report conclusions. This type of "trusted broker" solution requires all data providers to identify and employ a jointly trusted entity. For example, in order to prevent collisions between on-orbit satellites, some satellite operators employ Analytical Graphics Incorporated (AGI) as a trusted broker. AGI runs the Space Data Association (Anon n.d.), where members share their confidential orbital information with AGI, and AGI issues warnings and reports, but is contractually bound not to share the underlying data with any outside party. Similarly, the SCAlable National Network for Effectiveness Research (SCANNER) (Anon n.d.) developed distributed statistical analysis infrastructure to support hospitals build logistic regression model using locally aggregated statistics without exchanging sensitive patient level data. In many situations, however, legal and trust issues may make finding a trusted broker difficult or impossible. Even in situations where a mutually trusted broker can be found, employing their services can be costly and exposes sensitive data to an additional threat if the trusted broker is hacked and all sensitive data is stolen.

The trusted broker model provides a simple means of understanding the power of MPC. MPC can be viewed as a way of replacing a trusted broker with cryptographic algorithms, eliminating the need for mutual trust and the need to aggregate all the sensitive data in one place, making a single point of vulnerability. The trusted broker analogy also provides a useful means of identifying use-cases where MPC might be applicable. If the existence of a trusted broker would provide a means for securely computing statistics on sensitive data, then MPC could do the same without requiring the existence of a trusted broker. On the other hand, in situations where a trusted broker is insufficient to provide a completely secure solution, then MPC will also be insufficient. For example, a trusted broker who securely computes statistics and simply provides the results does not immediately solve the problem of inferential disclosure. This is discussed in detail in the next section.

Efficiency considerations: MPC protocols incur a significant overhead in both communication and computational costs compared to computing the same function insecurely. Different MPC architectures have different performance characteristics, but historically, the efficiency concerns have hindered deployment of MPC protocols. The cryptographic community has been working for decades to improve the performance of MPC protocols, and today, although efficiency concerns are an issue, knowledge and usability of the software are often bigger impediments to adoption.

II. MPC as a solution

A. What MPC is

A simple example — securely computing averages: Secure multiparty computation provides a means for securely computing *any* computable function on private data without the data owners ever sharing the data with each other or anyone else. At first glance, this appears to be an impossible guarantee. To illustrate the mathematics behind MPC, we describe a simple MPC protocol that allows three data owners to compute the mean of their private data.[1] Call the data owners A, B, C and their (private) data D_A, D_B, D_C. At a high level, the protocol works as follows. Each participant A, B, C will generate completely random numbers, sampled uniformly from a sufficiently large range (R_{ij}). They will then distribute the random values to the other participants, who will each add random numbers to their private values (D_A, D_B, D_C) to get enciphered values (Y_A, Y_B, Y_C). Finally, participants will combine the outputs of all local computations to obtain the final global computation result. Because each participant only obtains randomly generated values from the other participants, no information about each participant's private value is leaked.

The protocol to compute the mean consists of the following steps.

- **Randomness generation**
  - A generates random values R_{AB} and R_{AC} and sets $R_{AA} = D_A - R_{AB} - R_{AC}$
  - B generates random values R_{BA} and R_{BC} and sets $R_{BB} = D_B - R_{BA} - R_{BC}$
  - C generates random values R_{CA} and R_{CB} and sets $R_{CC} = D_C - R_{CA} - R_{CB}$
- **Distribution**
  - A sends R_{AB} to B and R_{AC} to C
  - B sends R_{BA} to A and R_{BC} to C
  - C sends R_{CA} to A and R_{CB} to B
- **Computation**
  - A computes $Y_A = R_{AA} + R_{BA} + R_{CA}$
  - B computes $Y_B = R_{AB} + R_{BB} + R_{CB}$
  - C computes $Y_C = R_{AC} + R_{BC} + R_{CC}$
- **Reconstruction**
  - The players share Y_A, Y_B, Y_C with each other and (publicly) compute $S = Y_A + Y_B + Y_C$

Now,

$$S = Y_A + Y_B + Y_C = \left(R_{AA} + R_{BA} + R_{CA}\right) + \left(R_{AB} + R_{BB} + R_{CB}\right) + \left(R_{AC} + R_{BC} + R_{CC}\right)$$

Rearranging, this becomes

$$S = \left(R_{AA} + R_{AB} + R_{AC}\right) + \left(R_{BA} + R_{BB} + R_{BC}\right) + \left(R_{CA} + R_{CB} + R_{CC}\right) = D_A + D_B + D_C$$

[1] We are grateful to Daniel Goroff for this example.

Then each participant can compute the mean which is simply $S/3$. This simple protocol can easily be adjusted to allow any number of participants to compute any *linear* function of their private data. Multiplying private values (e.g. computing a variance of private values) can also be done, but is significantly more complex, and requires different techniques (Ben-Or et al. 1988).

This example highlights an important feature of the security model of MPC: it does not protect against inferential disclosure. The above computation provably reveals no more information than revealing the mean alone, but the mean itself reveals information. A mean of three values is clearly highly dependent on the individual values, and thus the mean itself may reveal too much. This is described in more detail below. In cryptographic parlance, the random values R_{AA}, R_{AB}, R_{AC} are "secret shares" of the private value D_A and the distribution phase is called "secret sharing" (Blakley 1979; Shamir 1979). Secret sharing is a fundamental part of many MPC protocols, and despite its name, secret sharing does not reveal secrets, instead it provides a means of distributing secret data among participants such that no individual participant retains any information about the secret, but collectively they can reconstruct the secret.

Open algorithms and provable security: MPC provides a means of replacing a trusted broker with a cryptographic algorithm. Because MPC is technically complex, it is sometimes viewed as a sort of cryptographic "black-box", but if data owners are asked to replace their mutual trust in a broker with a mutually trusted algorithm, has the need for trust really gone down? In fact, using MPC for secure computation does not require blind trust in the algorithms and implementation. Existing MPC protocols are all based on public, published algorithms. These algorithms have mathematical proofs of security, and have been widely reviewed and analyzed by the cryptographic community. Because the algorithms underlying MPC are open, data owners are not forced to use software provided by a single vendor. Just as researchers can use standardized email protocols to send emails back and forth while using different mail clients and even operating systems, data owners can use public MPC protocols to perform secure computations on private data using software clients provided by different vendors. Security conscious data owners could, in principle, write their own MPC software clients, thus completely eliminating the need to trust any external software vendors or cryptographers. Alternatively, data owners could use open-source implementations of MPC protocols thus allowing each participant to independently verify the specific algorithms and implementations being used. It is precisely this openness that makes MPC a useful tool in reducing the need for mutual trust and eliminating trusted broker as an additional source of vulnerability. MPC does not provide "security through obscurity"; it provides security through mathematics.

B. What MPC is not

Differential privacy: MPC provides a means for a group of data owners to perform computations on their private data in such a way that performing the computation reveals no more information than the output alone would reveal. There are some situations, however, where the output alone reveals too much. In the example above, if three data owners securely compute the mean of their private values, the mean itself reveals a significant amount of information even when the sample size is larger. For example, calculating the mean wealth of residents in Medina, WA would reveal significant information about Bill Gates's fortune (Jones 2005). Even statistics that were deemed to be "anonymized" may reveal too much information about the underlying data sets (Heffetz & Ligett 2013). Protecting against problems of inferential disclosure requires alternative methods known as Differential Privacy, which usually involve adding noise to the data (Dwork 2008). Because MPC does not attempt to solve the problem of inferential disclosure, securely computing statistics using MPC yields exact results, not approximations. These are inherently complementary problems in privacy: MPC solves the problem of *computing* without leaking information, and differential privacy solves the problem of *disclosing results* without leaking information. Because MPC protocols can be used to securely compute any function, MPC protocols could be used to implement any rule-based mechanism to protect against inferential disclosure. For example, MPC could be used to compute cross-tabulations on private data, but only for those cells that have at least (say) ten observations. Similarly, MPC could be used to securely implement any differentially private mechanism (see e.g. (Pettai & Laud 2015)), and this combination would protect against information leakage from both the computation and the final result.

Record linkage: MPC provides a means of securely computing statistics on separate, private data sets. Computing statistics across multiple databases (even without privacy) requires a method for linking records in one database to another. For example, suppose a researcher wanted to examine connections between income and scholastic achievement, but education records are in a database held by the department of education, whereas income records are held by the IRS. If the education database has columns "social security

number" and "test score," and the IRS database has columns "social security number" and "household income," then linking the data sets (without privacy) is straightforward. In many real-world scenarios, however, linking records may be much more complex. The problem of identifying the correct way to link records across databases is not a question of privacy, and is not addressed by MPC technology. If researchers have a method for linking records across databases without privacy, MPC can perform the linkage (and subsequent statistical analysis) with privacy, but MPC cannot identify the "correct" way to link records across databases. Throughout the discussion below, we will always assume that vertically partitioned databases include a single column that provides a simple and natural way of linking records (e.g. a social security number or other unique identifier).

Clean data: Statistical research requires clean, consistent data sets, and most research projects devote significant time and resources to clean data before analysis begins. Statistics involving multiple data sets require additional care to ensure that the different databases are coded consistently, having uniform variable (column) names and variable types. Preparing data sets for analysis can be time consuming, but it is not a problem addressed by MPC. In the MPC use-cases outlined below, we assume the data owners have worked to ensure that their data sets are uniform and consistent before embarking on any secure statistical analyses. Thus data owners must mutually agree on things like column names and variable types before invoking MPC.

C. Previous applications of MPC

MPC has been proposed as a method for achieving financial oversight (Abbe et al. 2012; Flood et al. 2013), and general scientific calculations (Du et al. 2004; Lindell & Pinkas 2009; Wenliang Du et al. n.d.). Although cryptographers have touted the potential benefits of MPC for decades, early MPC protocols imposed such a computational burden on the participants as to be impractical for most real-world applications. The steady increase in computing power coupled with considerable algorithmic improvements have led to many real-world tests of MPC over the past few years. See (Archer et al. 2016) for a survey of the computational efficiency of modern MPC protocols.

The first major deployment of MPC was in an auction for Danish sugar beet production contracts (Bogetoft et al. 2009). In this application, each farmer had a (private) bid, and the MPC protocol securely computed the market clearing price, and allocated production rights to farmers. In this case, the MPC eliminated the need for a trusted auctioneer. Since then, MPC has been used for genome-wide association studies (Kamm et al. 2013; Jagadeesh et al. 2017; Cho et al. 2018), financial analytics (Bogdanov et al. 2012; Lapets et al. 2018), tax-fraud detection (Bogdanov, Jõemets, et al. 2016), identifying sexual offenders (Rajan et al. 2018) and preventing satellite collisions (Kamm & Willemson 2014; Hemenway et al. 2016). The Obliv-C framework (Zahur & Evans 2015) has been used for linear regression (Gascón, Schoppmann, Borja Balle, et al. 2017) and matching algorithms (Doerner et al. 2016). Nevertheless, MPC has not yet achieved its full potential, and the technology is not widely known outside of the cryptographic community.

III. Use cases and scenarios

One of the primary technical barriers to the widespread adoption of MPC technology has been the computational complexity of the MPC protocols themselves. Using MPC for a secure computation, without appropriate engineering considerations, might be thousands of times slower than performing the same computation insecurely. Thus MPC may be a viable solution for "simple" calculations, while it is currently of little value for securely computing extremely complex functionalities. In most situations, waiting milliseconds instead of nanoseconds to compute a set of regression coefficients is likely an acceptable tradeoff, whereas waiting years instead of days for the result of a more complex calculations is almost certainly unacceptable. As we build towards implementing a full suite of secure statistical tests, we will work in order of increasing complexity, starting from statistics that are almost certainly simple enough to admit efficient MPC implementations and gradually building towards more complex statistics that are closer to the limits of what can be computed efficiently using MPC.

A. Descriptions of data

We propose using MPC to allow researchers to perform statistical analyses across multiple data sets, where the data owners are unwilling or unable to share the underlying data sets with researchers or each other. Throughout this work, we imagine data sets as tables, where rows correspond to observations, and columns correspond to variables. **Figure 1** shows different ways data may be partitioned across multiple databases.

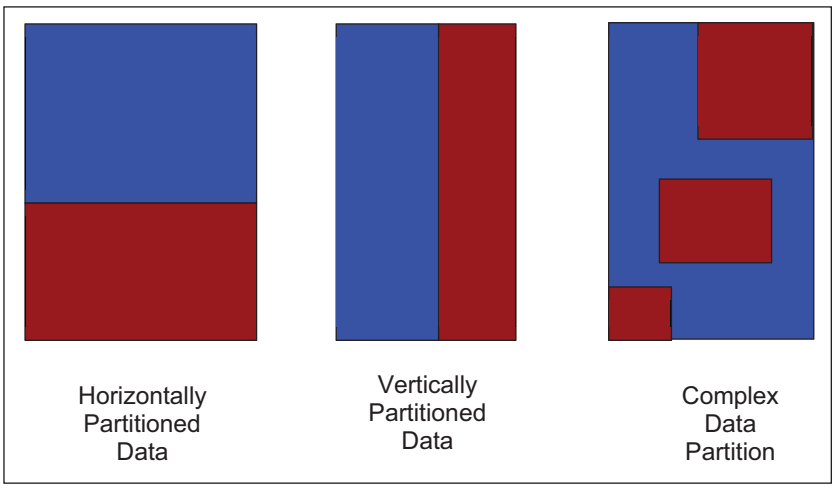

Figure 1: Different partitions of data ownership within a database.

Horizontally partitioned data: We use the term *horizontally partitioned* to denote the situation where two (or more) data owners have collected the same data from different subjects. Thus the data sets could be viewed as one large data set that was split horizontally. An example of a horizontal partitioning could be two different school districts, each of which collected information like age, classroom and test scores for students in their district.

Vertically partitioned data: We use the term *vertically partitioned* to denote the situation where data owners have different information about the same subjects. An example of a vertical partitioning could be joining education records (held by the school district) with health records (held by the medical provider). In this example, the school district may have information about an individual's scholastic aptitude, while the medical provider has information about the same individual's health.

Complex data partitions: There are also scenarios where the data are not simply vertically or horizontally partitioned, but instead are joined by a more complex relationship.

B. Examples

Health: In the U.S. it is common to change health insurers when changing schools, jobs or moving out of state. Although each insurer may keep detailed records about its clients, the data as a whole will be *horizontally* partitioned among the different insurers. Longitudinal studies are often stymied by a variety of regulations or privacy concerns.

Business & Finance: In the financial setting, each financial institution may have only a local view of its investments and risks. This inability to see the "big picture" can hinder oversight and lead to systemic risks that contribute to overall instability in the financial network or a supply chain within an industry (e.g. the effects of General Motors on the automotive industry and suppliers downstream). A data set where each row corresponds to the state of the financial network at a given point in time would then be *vertically* partitioned among financial institutions or regulatory agencies.

Social welfare: Researchers and policy-makers are routinely faced with decisions that require analyzing data from multiple domains. For example, examining how changes in the penal code impact truancy might require combining data from the judicial system and the schools. Or, examining how changes to the food stamp program impact obesity rates might require combining SNAP data with data from local health clinics. In most of these situations, the data will be *vertically* partitioned, with different entities holding different data about each individual in the population.

Education: Designing an efficient and effective education system is crucial for the long-term health, happiness and productivity of the population. On the other hand, the sensitive nature of data relating to children make fine-grained education data difficult to obtain and distribute. Comparing how policy-changes (e.g. changes to the tax code, penal code, welfare programs or Medicaid) impact student learning might require analyzing vertically partitioned data sets, whereas detailed statistics about the population of students (encompassing public schools, private schools and charter schools) might require analyzing *horizontally* partitioned data. MPC has already been used to link student and tax records in order to examine the effects of employment on academic performance (Bogdanov, Kamm, et al. 2016).

C. Statistics

Descriptive statistics: Some of the most useful and widely used statistics are also the simplest, and the first task for designing an MPC-based software suite will be to implement secure versions of basic descriptive statistics. These include means, variances, covariances and cross-tabulations (e.g. means by category). These statistics can all be expressed as polynomials of degree at most two, and thus are extremely amenable to secure computation. Slightly more complicated model-based statistics, like least squares, generalized least squares, and k-means clustering can also be implemented within an MPC protocol without too much extra computational overhead.

Multiple regression: Multiple (linear) regression is only slightly more computationally complex than descriptive statistics. The regression coefficients can be expressed as $(X^TX)^{-1}X^TY$, where X is the matrix of independent variables, and Y is the matrix of dependent variables. No matter, how the data (X and Y) are partitioned, the regression coefficients could be computed by first secret-sharing X and Y, and then computing the above statistic using a generic MPC framework.[2] The most computationally difficult step in this procedure is the inversion of the matrix X^TX. In most real-world scenarios, however, the matrix X has only a small number of columns (i.e., the regression uses only a small number of predictors), thus the matrix X^TX will be small even if the total number of observations is large. Thus, the secure computation of a matrix inversion needs to only calculate the inverse of a small matrix. Because of this fact, the bottleneck in performance of the MPC solution will most likely be in the communication since the matrix X may have many rows (observations).

Logistic regression: Logistic regression is more computationally complex than linear regression, and unlike linear regression there is no completely general closed-form solution for performing a logistic regression. In practice, logistic regression algorithms are iterative, which will impact the design of MPC protocols. Although there are many methods for computing a logistic regression, the method of iteratively reweighted least squares (IRLS) seems like it will be the most amenable to secure computation. Using IRLS to compute a logistic regression should require no more than ten iterations, where each iteration is essentially a (weighted) linear regression calculation. Thus using IRLS allows us to leverage the (secure) linear regression routines outlined above and compute logistic regressions with about ten-fold slowdown over (secure) linear regression.

Survival analysis: Moving up a level in complexity, we come to survival analysis algorithms, like the Cox Proportional Hazards Model (Cox 1972). These statistics are widely used, and are of specific interest in the health-care use cases outlined above. Calculating a Cox model requires first sorting the data, and then performing an iterative optimization algorithm. Secure sorting is itself a well-studied task (Jónsson et al. 2011; Hamada et al. 2013), and most secure sorting is based on sorting networks (Batcher 1968). Even after (securely) sorting the data, optimizing the objective function requires a relatively complex secure calculation, e.g. Newton's Method. It is an interesting empirical question whether MPC protocols can be made to implement survival analysis algorithms like the Cox model efficiently enough for practical applications.

More complex statistics: Creating an MPC platform that could securely compute arbitrarily complex statistics (e.g. full Bayesian inference on completely general models) would be of great benefit to the community. Given the wide variety of statistics that researchers would like to compute, in order to address this problem, better tools for compiling general statistical algorithms into MPC protocols need to be developed. In recent years, however, significant progress has been made in MPC compilers (Aly et al. n.d.; Zhang et al. 2013; Wang et al. 2017). Although these compilers can be used to run arbitrary computations, the cryptographic requirements of the MPC protocols mean that the most efficient (insecure) algorithms for calculating a statistic of interest, may not be the most efficient algorithms to run within a secure computation (Esperança et al. 2017). Finding the most "MPC-friendly" algorithm for computing statistics of interest will require collaborations between statisticians and cryptographers.

D. Privacy assumptions

Collusion: When secure computations involve more than two participants, there is a risk that a subset of participants will collude in order to violate the privacy of other participants. For example, if a group of hospitals wanted to compute joint statistics about the overall rate of hospital acquired infections, could a subset of the hospitals collude, sharing information between themselves in order to gain more information

[2] For specific data partitions (e.g. one participant holds X and the other holds Y, or a simple vertical or horizontal partitioning) further optimizations can be made.

about one of their competitors? The problem of collusion is directly addressed by MPC protocols and there exist MPC protocols that are fully collusion resistant — even if all of the other participants collude against you, your private data will remain secure (Ishai et al. 2008; Keller et al. 2013). Collusion-resistance has a price, however, and protocols that are fully collusion resistant are more computationally intensive (and hence slower) than protocols that are not collusion resistant. A good rule of thumb, however, is that when the majority of participants are assumed to be non-colluding, MPC protocols can be much faster. Thus if there are five participants in a computation, providing privacy in the face of three colluding participants is much more difficult than only providing privacy against at most two colluding participants.[3] In the use-cases outlined below, collusion is usually not a serious concern. It seems unlikely, say, that two school districts will collude together to steal sensitive information from a third.

Even when all the participants are honest, collusion-resistance can be important, as it protects individual participants against the negligence of others. For example, in a fully collusion-resistant protocol, if an adversary broke into the systems of all the other participants, stole their data, and monitored all their communications, the adversary would still learn nothing about your private data.

Leakage: General-purpose MPC protocols are designed to leak no information that could not be inferred from the output alone. This is an extremely strong security guarantee, and in some situations by "leaking" additional information the protocols can be made significantly more efficient. As a simple example, suppose that two data owners wish to compute the mean of their combined data. If the owners are willing to first share the total number of records in their data sets, the secure computation will be much faster than if the number of observations must also remain private. If the number of records is shared, then the stakeholders can securely compute the *sum* of their private data and obtain the mean by dividing by the (public) number of records. If the number of records must remain private as well, then the division must be computed securely as well, which increases the computational complexity of the protocol. For a slightly more complex example, consider a statistical computation that is computed iteratively, e.g. a Cox proportional hazards model. In calculations like these, if the intermediate iterates can be made public, the secure computation can be made dramatically more efficient. In fact, in some cases, sharing intermediate values in a computation can eliminate the need for MPC entirely (Lu et al. 2015). Determining what portions of the computation (if any) can be made public requires domain expertise, and identifying what leakage will yield performance gains in MPC requires cryptographic expertise, thus questions about leakage will require cross-disciplinary collaboration.

Threat models (passive vs active): Designing secure protocols requires carefully specifying the threat model, and MPC protocols can be divided into two categories depending on whether they provide passive or the stronger notion of active security.[4] Passive security models a world where the participants in an MPC protocol follow the specifications of the protocol, but may try to glean information from any messages they receive. The passive security model effectively captures the scenario where a hacker or rogue employee gains access to a machine, monitors its communication and exfiltrates data, but (for fear of detection) does not otherwise interfere with the running of the system. A passively secure MPC protocol would ensure that even if the other participants were compromised in this way, your private data would remain safe. Actively secure protocols provide strong security guarantees even when participants actively try to subvert the protocol, possibly forging data or sending mal-formed messages in an attempt to gather information about other participants' private data. Actively secure protocols ensure that your data remain safe *no matter what the other participants do.* Like everything else, this stronger security guarantee comes at a price, and actively secure protocols require more computation and communication than their passively secure counterparts. It is common to first develop and implement passively secure protocols, and once these protocols are in place and validated, then move to the stronger notion of active security.

IV. MPC architectures

There are many different ways MPC could be used to provide a means of securely computing statistics on jointly private data sets. In this section, we review three different MPC architectures and their implications for the data owners. In the following, a "cloud server" refers to a commodity cloud computing provider like Microsoft Azure, Amazon EC2 or Google's Compute Engine. In all the solutions outlined below, there is no need to "trust" these cloud service providers, and they will never have access to the underlying data. This is an important point: even an attacker who had complete power to read all data stored by the cloud, and could

[3] The amount of collusion is measured by the largest single block of colluding participants. Thus in a protocol with five participants, if two groups of two collude, that is still only a collusion of size two.

[4] In the cryptographic literature, passive security is often called security against semi-honest adversaries, or security in the honest-but-curious model. Active security is often called security against malicious adversaries.

view all incoming and outgoing traffic to the cloud, would provably learn nothing about any of the sensitive data. Using a cloud provider may be beneficial because MPC protocols are extremely computationally and bandwidth intensive, and offloading this burden to the cloud may alleviate some of the pressures on the data owners. The third solution outlined below does not make use of cloud providers, and will likely be the most desirable solution when data owners do not want an untrusted cloud provider to handle even encrypted data.

All of the scenarios outlined below could support an arbitrary number of data owners. **Table 1** provides a brief summary of some of the characteristics of the three different architectures.

A. Single cloud

The data owners could employ a single cloud server to assist in the computation **Figure 2**. In this scenario, the data owners will make use of the power of fully homomorphic encryption (FHE) to allow the cloud to perform computations on their encrypted data (Gentry 2009). First, the data owners engage in a short MPC protocol to generate a public-key/private-key pair for a fully homomorphic cryptosystem. This protocol will leave each data owner with the resulting FHE public-key and a secret-sharing of the private-key. The public-key for the FHE scheme will allow each of the data owners to encrypt, but since none of the owners has the private-key, the only way an FHE ciphertext can be decrypted is if the data owners engage in a further MPC protocol to decrypt it. Using the public-key, each data owner will encrypt their private data set, cell-by-cell, and upload the encrypted data to the cloud. The security of the FHE scheme guarantees that the cloud provider will learn the number of rows provided by each of the data owners and *nothing* else. Once the cloud provider is in possession of the encrypted data sets, a researcher can make a statistical query to the cloud (e.g. compute the mean of a certain column), and using the power of the FHE scheme, the cloud can compute an *encryption* of the answer. Because the data are never decrypted, even the cloud itself cannot learn the answer to the researcher's query. Finally, the cloud will provide the encrypted result to the data owners, and the data owners will engage in a short MPC protocol (using their shares of the decryption key) to decrypt the ciphertext containing the result of the researcher's query. The PALISADE FHE library recognizes the utility of this model and provides wrapper-functions for multiparty key generation and decryption, that interface with all of its different FHE back-end implementations (Ryan & Rohloff n.d.).

This model has been suggested for privately tallying votes (Damgård et al. 2003). In the voting context, a set of authorities would engage in a distributed key generation protocol to generate the public key for an additively homomorphic cryptosystem, and the stakeholders would keep shares of the private key. Voters could encrypt their votes (using the public key), and these votes could be privately tallied (using the additive

Table 1: MPC architectures.

Model	Trust Requirements	Performance	Involvement of data owners	Scalability with number of data owners
Single cloud	Low	Low	Low	Good
Multiple cloud providers	Medium	High	Low	Good
Private servers	Low	Medium	High	Bad

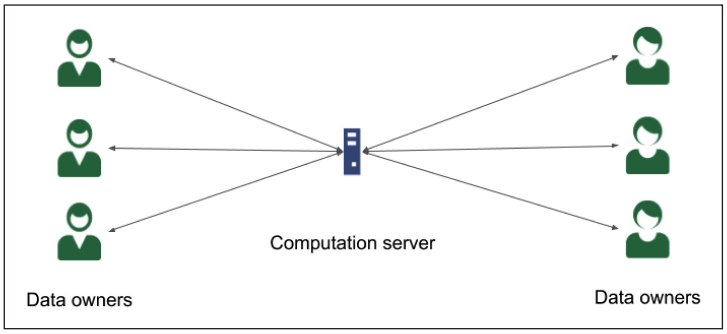

Figure 2: Data owners send (encrypted) data to the computation server, the server performs the computation, and returns (encrypted) results. In this protocol, the server learns nothing about the underlying data. The computational burden on the data owners does not increase as the complexity of the computation increases.

homomorphism). Finally, the authorities could engage in an MPC to decrypt the ciphertext containing the cumulative result. In the voting context, because the function is linear (summing the votes) this protocol can be made very efficient.[5]

Because the MPC protocol is only used for key generation and decryption, the cost of the MPC protocol does not increase with the data size or the complexity of the underlying statistics of interest. This means that the efficiency of the underlying MPC protocol will most likely not be the determining factor in the efficiency of the overall protocol.

Security guarantees: In the single-cloud model, researchers can make statistical queries to the cloud server without involving the data owners, but because the results are decrypted, the researcher (and the cloud) can never learn the answer to any of these queries without involving the data owners. Thus the data owners maintain complete control over what queries are answered, and who sees the answers. Because the underlying data are never decrypted, there is no risk that the cloud could learn any information about the underlying data sets or leak this information through its negligence.

Efficiency considerations: Currently, FHE technology is less efficient than other MPC solutions, so this single-cloud scenario, will be more computationally intensive than the other solutions outlined below. Using state-of-the-art FHE schemes, this type of architecture should be able to provide a method for efficiently computing descriptive statistics (e.g. means, variances, covariances, crosstabs), but the computational burden may be too great to compute more complicated statistics like multiple regressions on large data sets. For example, using FHE to compute a linear regression on a small data set (200 observations and 20 variables) took about three hours of computation time (Lu et al. 2017). Although this architecture places an extremely high computational burden on the cloud provider, it has benefits for the data owners. The computational burden on the data owners is minimal and once the encrypted data sets are sent to the cloud, researchers can make any number of statistical queries with minimal communication costs. Of the three architectures proposed in this section, this one has the highest computational cost, but the lowest communication cost.

B. Multiple cloud providers

Instead of employing a single cloud provider, the data owners could employ multiple cloud servers to assist them in their secure computations **Figure 3**. Employing two cloud servers allows them to use Yao's garbled circuits (Yao 1982; Yao 1986), or the GMW protocol (Goldreich et al. 1987). Employing three (or more) cloud servers allows them to use the GMW or BGW protocols (Ben-Or et al. 1988). In this scenario, the data owners would use secret sharing to distribute their data among the cloud providers. The security of the secret sharing protocol, ensures that each cloud server learns no information about any of the underlying data sets. Once the sensitive data sets are secret shared to the cloud servers, a researcher could engage in an MPC protocol

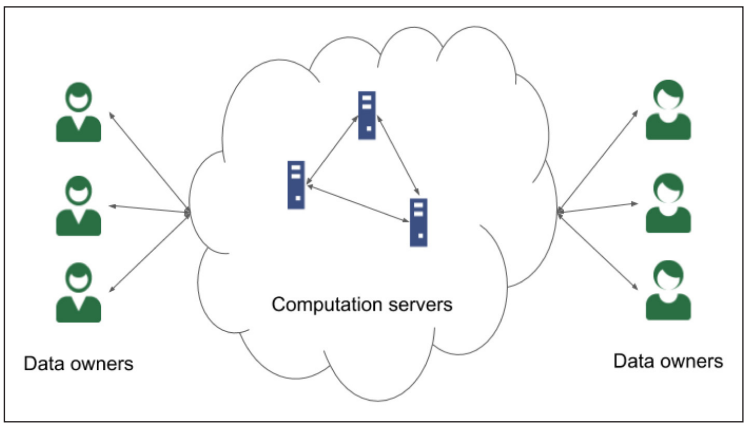

Figure 3: The data owners "secret-share" their data among a small number of computation servers. The servers execute an MPC protocol and return the result (or encryptions of the result) to the data owners (or an analyst). In this model, the data owners must trust the computation servers not to collude. If the servers do not collude, then the servers learn nothing about the data (or nothing beyond what is revealed by the output of the computation alone if the result is returned in the clear). The computational burden on the data owners does not increase as the complexity of the computation increases.

[5] The naïve protocol described here does not address many of the primary key challenges in the voting context which revolve around authenticating voters, preventing double-voting, and proving to a voter that her vote was indeed included in the final tally. These problems can be addressed using cryptography, and many different special-purpose protocols have been devised. See e.g. (Groth 2005).

to calculate statistics on the aggregate data sets. Each cloud server would have its own MPC software client, and performing the MPC calculation would require computation and communication between the cloud servers. Finally, the result would be returned to the researcher. This architecture has the flexibility to allow the result of the computation to be returned directly to the researcher, or to the data owners. In the former case, the data owners might specify a certain set of "legal" statistical tests, and then they would not need to interact directly with researchers who wanted to perform tests from this category. In the latter case, the data owners would have to explicitly "ok" each result before it was returned to the researcher. For efficiency reasons, the three cloud scenario is extremely popular in implementations.[6]

Security guarantees: In the multi-cloud scenario, there is a risk that all the cloud providers will collude together to learn information about the private data. If all of the cloud servers collude against the data owners, all privacy is lost. This type of scenario can be mitigated by choosing cloud servers from different vendors, under the assumption that it is extremely unlikely that Microsoft, Amazon and Google (or their rogue employees) will join together to collude against the data owners. If at least one cloud server refuses to collude against the data owners, then complete privacy can be maintained. As discussed above, the higher the threshold for collusion-tolerance, the more inefficient the MPC protocol becomes. Protocols in which strictly fewer than half of the cloud servers collude against the data owners are generally the most efficient.

Efficiency considerations: By secret sharing their data to multiple cloud providers, the data owners can offload essentially all of the computational burden of MPC to the cloud providers. Unlike the single-cloud scenario above, in the multi-cloud scenario every statistical query will require large amounts of communication between the cloud servers. In most situations, cloud providers have extremely fast network connections, and so the communication burden between the clouds may not be a bottleneck. As a performance baseline, a three-server implementation of linear regression using the Sharemind platform was used to compute a linear regression with 10k observations and 10 variables in 12 seconds (Bogdanov et al. 2018). Two-server, garbled-circuit implementations of regression have run computations with millions of observations (Nikolaenko et al. 2013; Gascón, Schoppmann, Balle, et al. 2017).

C. Private servers

In order to maintain complete control over their data sets, data owners can eliminate the cloud servers and perform the MPC protocol themselves. Eliminating the cloud servers reduces the attack surface, but places much larger computational burden on the data owners themselves. In this model, each data owner (and possibly the researcher) will maintain an in-house networked server (**Figure 4**). This server will contain the data owner's private data set, and will run the MPC client software. As discussed above, because the MPC algorithms are public, the data owners could, in principle, purchase their MPC client software from different vendors or each write their own MPC software client. When a researcher wants to calculate a statistic, the data owners' servers will engage in the MPC protocol, and obtain the result, which can then be relayed to the researcher.

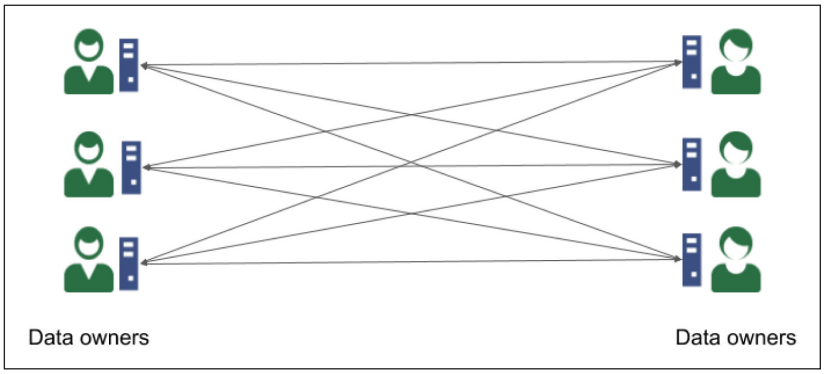

Figure 4: The data owners play the role of computation servers, and each data owner installs and runs the MPC client software locally. In this model, the data owners no longer have to trust the computation servers not to collude. On the other hand, the this increases the computational (and communication) burden of the data owners who must now execute the MPC protocol themselves. Since all data owners must now communicate with all other data owners, the communication cost of this architecture does not scale to support a large number of data owners.

[6] The three-cloud environment is the basis of the Sharemind platform, and all of their MPC use-cases (https://sharemind.cyber.ee/).

Security guarantees: In the private-server model, none of the data owners (or their MPC software clients) ever accesses anyone else's private data, the only threat to data security is collusion among the data owners (i.e., some subset of owners works together to steal data from another participant). As discussed above, MPC protocols exist which provide privacy even if all other data owners collude against one target owner. If such a protocol is used, then no assumptions about collusion are necessary to ensure data privacy. For efficiency reasons, however, it may be necessary to design protocols that only provide privacy when the majority of data owners are non-colluding.

Efficiency considerations: This architecture imposes the largest computational burden on the data owners themselves, and in this scenario the data owners will most likely each have to assign a dedicated server to perform the MPC computation. Because the MPC protocols are extremely bandwidth intensive, for this solution to be practical, the data owners' servers must be connected with high-bandwidth, low-latency network connections. In this architecture, all data owners (or their software clients) must send messages to all others, the overall communication cost of the protocol grows quadratically with the number of data owners. By contrast, in the two protocols discussed above ("single server" and "multiple cloud providers") the total communication cost only grows linearly with the number of data owners. Thus, this model may be impractical when the number of data owners is large. For reference, the largest MPC computation to date had 128 participants (Wang et al. 2017).

V. Possibilities and Frontiers

A. Integration with differential privacy

MPC does not attempt to address the problem of inferential disclosure. The outputs of an MPC protocol are exact, and the security guarantee is that all the messages exchanged in the MPC protocol during computation of the output reveal no more information than what is revealed by the output alone. On the other hand, MPC protocols are extremely general, and in principle MPC can be used to securely compute *any* functionality — including a differentially private one. Moving forward, we imagine building systems that integrate MPC with differential privacy to ensure that neither the computation, nor the final output violate individual privacy. Integrating MPC and differential privacy is an extremely natural goal, and this integration has been proposed (Flood et al. 2013) and prototyped (Narayan et al. 2014) in the context of overseeing financial markets.

B. Certificates of correctness of computation

As discussed above, transparency and reproducibility are core tenets of the open science movement, as these provide a means for the scientific community to verify and validate each other's work. Unfortunately, privacy concerns often prevent publicly sharing data sets, and this presents a barrier to transparency and reproducibility. A well-designed MPC solution provides a means for achieving reproducibility without sharing any of the underlying sensitive data. Data owners running MPC clients could essentially provide researchers with a virtual interface to their data sets, thus allowing future researchers to reproduce statistical calculations without the need for publicly posting the data sets.

Cryptography provides many additional tools to verify the correctness of computations without violating privacy. Using digital signatures, data owners could "sign" the output of a computation. Digital signatures could be easily integrated into a secure computation protocol, so that when computing (say) a linear regression, the secure computation yields both the regression coefficients as well as a digital signature from the data owners that attests that these regression coefficients were obtained from a specific computation on their data sets. More powerful tools like Succinct Non-interactive zero-knowledge ARguments of Knowledge (SNARKs) (Parno et al. 2013; Ben-Sasson et al. 2013) provide a means of "proving" that an entire calculation was carried out correctly. Using SNARKs, researchers would have a short, easily verifiable certificate that their statistics were obtained from specific computations on specific data sets. Thus, SNARKs provide a novel means of allowing the research community to verify and validate statistical computations, without actually reproducing the computations in the traditional sense. The cryptocurrency ZCash (Hopwood et al. 2018) uses SNARKs (based on the libsnark library (Lab n.d.)) to enhance user privacy.

C. Secret disclosure controls

To limit inferential disclosure and prevent linkage attacks, MPC protocols can be made to implement complex mechanisms like differential privacy, or simple mechanisms like pruning outliers or suppressing the output of calculations that do not depend on enough records. In some situations, however, the disclosure control rules themselves must remain private. This can occur when the database deals with classified mate-

rials, and a user without proper clearance should not be able to identify which types of queries are being censored. IARPA's SPAR program[7] sought to develop and implement protocols that allowed users to query a database privately, while simultaneously enforcing query policy compliance. The SPAR program was primarily concerned with SELECT queries (e.g. SELECT name, occupation WHERE age > 35), whereas we envision supporting statistical queries, so the SPAR technology cannot be used as-is. Although in many applications there is no need to maintain the privacy of the disclosure control mechanisms themselves, MPC has the ability to provide this strong notion of privacy, and thus is potentially useful in settings that demand even more privacy.

VI. MPC Software

A number of open-source software "compilers" have been developed that allow users to write code in a high-level language that will then be executed as a secure multiparty computation. These tools make it much easier to prototype and deploy MPC computations, however, they still require some cryptographic expertise to use and deploy. These compilers are academic software projects and not "enterprise-ready" software. **Table 2** shows a few of the most efficient, usable and actively developed open-source MPC compilers.

Many other compilers exist, but these are currently some of the most efficient, usable and actively developed. Docker containers containing working examples for these (and other) MPC compilers can be found online.[14] Several libraries for Fully Homomorphic Encryption have also been developed, including HElib,[15] PALISADE[16] and SEAL.[17]

VII. Conclusion

MPC technology is now sufficiently advanced to be an efficient and practical method for certain types of statistical calculations. This paper serves to identify and outline some of the cryptographic tools that would most empower researchers, satisfy the privacy requirements of data owners, and builds a roadmap of approaches to meeting these requirements that can be implemented by cryptographers.

In the near future, we expect to see many novel real-world applications of MPC supporting scientific research. To expedite this process, we have laid out a skeleton of the types of statistical techniques and MPC protocols that are most likely to be of use this in this arena. Developing the most promising use-cases and building practical and efficient MPC protocols to support these use-cases will require collaboration between cryptographers and data scientists.

Table 2: MPC compilers.

System	Number of parties	Security model	Function Language
SCALE-MAMBA[8]	2+	Covert[9]	Python-like
PICCO[10]	3+	Semi-honest	C
EMP-Toolkit[11]	2+	Semi-honest or Malicious	C
Obliv-C[12]	2	Semi-honest	C
ABY[13]	2	Semi-honest	C++

[7] IARPA's Security and Privacy Assurance Research (SPAR) program was active from 2011-2014. http://www.iarpa.gov/index.php/research-programs/spar (accessed 03/09/2015).
[8] https://homes.esat.kuleuven.be/~nsmart/SCALE/.
[9] The "covert" security model is slightly weaker than a fully malicious security model. In the malicious model, users are explicitly prevented from deviating from the prescribed protocol by the protocol itself. In the covert security model users can cheat, but if they deviate from the protocol, other users will be able to see (and later prove) that the user cheated. Thus cheating users can be caught and penalized outside of the protocol itself (e.g. by legal means).
[10] https://github.com/PICCO-Team/picco.
[11] https://github.com/emp-toolkit.
[12] https://oblivc.org/.
[13] https://github.com/encryptogroup/ABY.
[14] https://github.com/MPC-SoK/frameworks.
[15] https://shaih.github.io/HElib/.
[16] https://git.njit.edu/palisade/PALISADE.
[17] https://www.microsoft.com/en-us/research/project/simple-encrypted-arithmetic-library/.

One of the primary goals of this paper is to begin building bridges between these two communities that will lead to new opportunities to extract useful information from the wealth of data being collected by governments and the private sector without compromising the privacy of individuals or corporations. An MPC system for statistical analysis will thusly unleash the potential of enormous quantities of data that are currently restricted by confidentiality, privacy, and secrecy concerns.

Acknowledgements

The authors gratefully acknowledge funding from the Alfred P. Sloan foundation for "Computing Statistics from Private Data" (Grant Number G-2014-13803), which supported the workshops that resulted in this paper. We are also grateful to the participants in those workshops for their insights and comments on early drafts of this paper: Brian Athey, Mark Flood, Shafi Goldwasser, Marcelline Harris, Mary Hill, Susan Jekielek, Xiaoqian Jiang, John Marcotte, Thomas Murphy, Brian Perron, Trivellore Raghunathan, Kurt Rohloff, Alessandra Scafuro, Barbara Schneider, Vitaly Shmatikov, Jaideep S. Vaidya, Vinod Vaikuntanathan, Shuang Wang, Xiao Wang, Kristine Witkowski.

Competing Interests

The authors have no competing interests to declare.

References

Abbe, EA, Khandani, AE and **Lo, AW.** 2012. Privacy-Preserving Methods for Sharing Financial Risk Exposures. *American Economic Review*, 102(3): 65–70. DOI: https://doi.org/10.1257/aer.102.3.65

Aly, A, et al. SCALE-MAMBA Software. Available at: https://homes.esat.kuleuven.be/~nsmart/SCALE/ [Accessed November 1, 2018].

Anon. SCANNER. Available at: http://scanner.ucsd.edu/ [Accessed October 31, 2018a].

Anon. Space Data Association. Available at: http://www.space-data.org/sda/ [Accessed October 31, 2018b].

Archer, DW, et al. 2016. Maturity and Performance of Programmable Secure Computation. *IEEE security & privacy*, 14(5): 48–56. DOI: https://doi.org/10.1109/MSP.2016.97

Batcher, KE. 1968. Sorting networks and their applications. In: *Proceedings of the April 30–May 2, 1968, spring joint computer conference on – AFIPS '68 (Spring)*. DOI: https://doi.org/10.1145/1468075.1468121

Ben-Or, M, Goldwasser, S and **Wigderson, A.** 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing – STOC '88*. DOI: https://doi.org/10.1145/62212.62213

Ben-Sasson, E, et al. 2013. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In: *CRYPTO*, 90–108. DOI: https://doi.org/10.1007/978-3-642-40084-1_6

Blakley, GR. 1979. Safeguarding cryptographic keys. In: *Managing Requirements Knowledge, International Workshop on*. Available at: https://www.computer.org/csdl/proceedings-article/1979/afips/50870313/12OmNCeK2a1 [Accessed October 31, 2018].

Bogdanov, D, et al. 2018. Rmind: A Tool for Cryptographically Secure Statistical Analysis. *IEEE Transactions on Dependable and Secure Computing*, 15(3): 481–495. DOI: https://doi.org/10.1109/TDSC.2016.2587623

Bogdanov, D, Jõemets, M, et al. 2016. *Privacy-preserving tax fraud detection in the cloud with realistic data volumes*. Cybernetica. Available at: https://cyber.ee/uploads/2013/05/T-4-24-Privacy-preserving-tax-fraud-detection-in-the-cloud-with-realistic-data-volumes-1.pdf.

Bogdanov, D, Kamm, L, et al. 2016. Students and Taxes: A Privacy-Preserving Study Using Secure Computation. *Proceedings on Privacy Enhancing Technologies*, 2016(3): 117–135. DOI: https://doi.org/10.1515/popets-2016-0019

Bogdanov, D, Talviste, R and **Willemson, J.** 2012. Deploying Secure Multi-Party Computation for Financial Data Analysis. In: *Financial Cryptography*, 57–64. DOI: https://doi.org/10.1007/978-3-642-32946-3_5

Bogetoft, P, et al. 2009. Secure Multiparty Computation Goes Live. In: *Financial Cryptography*, 325–343. DOI: https://doi.org/10.1007/978-3-642-03549-4_20

Cho, H, Wu, DJ and **Berger, B.** 2018. Secure genome-wide association analysis using multiparty computation. *Nature biotechnology*, 36(6): 547–551. DOI: https://doi.org/10.1038/nbt.4108

Cox, DR. 1972. Regression Models and Life Tables. *Journal of the Royal Statistical Society*, 34(2): 187–220.

Damgård, I, Groth, J and **Salomonsen, G.** 2003. The Theory and Implementation of an Electronic Voting System. In: *Advances in Information Security*, 77–99. DOI: https://doi.org/10.1007/978-1-4615-0239-5_6

Doerner, J, Evans, D and **Shelat, A.** 2016. Secure Stable Matching at Scale. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security – CCS'16*. DOI: https://doi.org/10.1145/2976749.2978373

Du, W and **Atallah, MJ.** Privacy-preserving cooperative scientific computations. In: *Proceedings 14th IEEE Computer Security Foundations Workshop, 2001*. DOI: https://doi.org/10.1109/csfw.2001.930152

Du, W, Han, YS and **Chen, S.** 2004. Privacy-Preserving Multivariate Statistical Analysis: Linear Regression and Classification. In: *Proceedings of the 2004 SIAM International Conference on Data Mining*, 222–233. DOI: https://doi.org/10.1137/1.9781611972740.21

Dwork, C. 2008. Differential Privacy: A Survey of Results. In: *Theory and Applications of Models of Computation*, 1–19. DOI: https://doi.org/10.1007/978-3-540-79228-4_1

Esperança, PM, Aslett, LJM and **Holmes, CC.** 2017. Encrypted accelerated least squares regression. Available at: https://arxiv.org/abs/1703.00839.

Flood, MD, et al. 2013. Cryptography and the Economics of Supervisory Information: Balancing Transparency and Confidentiality. *SSRN Electronic Journal*. DOI: https://doi.org/10.2139/ssrn.2320795

Galvan, DA, et al. 2014. *Satellite Anomalies: Benefits of a Centralized Anomaly Database and Methods for Securely Sharing Information Among Satellite Operators*. Rand Corporation.

Gascón, A, Schoppmann, P, Balle, B, et al. 2017. Privacy-Preserving Distributed Linear Regression on High-Dimensional Data. *Proceedings on Privacy Enhancing Technologies*, 2017(4): 345–364. DOI: https://doi.org/10.1515/popets-2017-0053

Gascón, A, Schoppmann, P, Balle, B, Raykova, M, et al. 2017. Secure Linear Regression on Vertically Partitioned Datasets. In: *PoPETs*, 345–364.

Gentry, C. 2009. Fully homomorphic encryption using ideal lattices. In: *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing – STOC '09*. DOI: https://doi.org/10.1145/1536414.1536440

Goldreich, O, Micali, S and **Wigderson, A.** 1987. How to play ANY mental game. In: *Proceedings of the nineteenth annual ACM conference on Theory of computing – STOC '87*. DOI: https://doi.org/10.1145/28395.28420

Groth, J. 2005. Non-interactive Zero-Knowledge Arguments for Voting. In: *Lecture Notes in Computer Science*, 467–482. DOI: https://doi.org/10.1007/11496137_32

Hamada, K, et al. 2013. Practically Efficient Multi-party Sorting Protocols from Comparison Sort Algorithms. In: *Lecture Notes in Computer Science*, 202–216. DOI: https://doi.org/10.1007/978-3-642-37682-5_15

Heffetz, O and **Ligett, K.** 2013. *Privacy and Data-Based Research*. NBER. DOI: https://doi.org/10.3386/w19433

Hemenway, B, et al. 2016. High-Precision Secure Computation of Satellite Collision Probabilities. In: *Security and Cryptography in Networks*, 169–187. DOI: https://doi.org/10.1007/978-3-319-44618-9_9

Hopwood, D, et al. 2018. *Zcash Protocol Specication Version 2018.0-beta-32 [Overwinter + Sapling]*. Available at: https://github.com/zcash/zips [Accessed November 2, 2018].

Ishai, Y, Prabhakaran, M and **Sahai, A.** 2008. Founding Cryptography on Oblivious Transfer – Efficiently. In: *CRYPTO*, 572–591. DOI: https://doi.org/10.1007/978-3-540-85174-5_32

Jagadeesh, KA, et al. 2017. Deriving genomic diagnoses without revealing patient genomes. *Science*, 357(6352): 692–695. DOI: https://doi.org/10.1126/science.aam9710

Jones, JL. 2005. The Sound's wealthiest zip codes. *Puget Sound Business Journal*. Available at: https://www.bizjournals.com/seattle/stories/2005/02/07/focus1.html [Accessed November 1, 2018].

Jónsson, KV, Kreitz, G and **Uddin, M.** 2011. Secure Multi-Party Sorting and Applications. Available at: https://eprint.iacr.org/2011/122.

Kamm, L, et al. 2013. A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics*, 29(7): 886–893. DOI: https://doi.org/10.1093/bioinformatics/btt066

Kamm, L and **Willemson, J.** 2014. Secure floating point arithmetic and private satellite collision analysis. *International Journal of Information Security*, 14(6): 531–548. DOI: https://doi.org/10.1007/s10207-014-0271-8

Keller, M, Scholl, P and **Smart, NP.** 2013. An architecture for practical actively secure MPC with dishonest majority. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security – CCS '13*. DOI: https://doi.org/10.1145/2508859.2516744

Lab, S. libsnark: A C++ library for zkSNARK proofs. Available at: https://github.com/zcash/libsnark [Accessed November 2, 2018].

Lapets, A, et al. 2018. Accessible Privacy-Preserving Web-Based Data Analysis for Assessing and Addressing Economic Inequalities. In: *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS) – COMPASS '18*. DOI: https://doi.org/10.1145/3209811.3212701

Lindell, Y and **Pinkas, B.** 2009. Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, 1(1). DOI: https://doi.org/10.29012/jpc.v1i1.566

Lu, C-L, et al. 2015. WebDISCO: A web service for distributed cox model learning without patient-level data sharing. *Journal of the American Medical Informatics Association: JAMIA*, 22(6): 1212–1219. DOI: https://doi.org/10.1093/jamia/ocv083

Lu, W, Kawasaki, S and **Sakuma, J.** 2017. Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data. In: *NDSS*. Available at: https://eprint.iacr.org/2016/1163.

Narayan, A, Papadimitriou, A and **Haeberlen, A.** 2014. Compute globally, act locally: Protecting federated systems from systemic threats. In: *USENIX*.

Nikolaenko, V, et al. 2013. Privacy-Preserving Ridge Regression on Hundreds of Millions of Records. In: *2013 IEEE Symposium on Security and Privacy*. DOI: https://doi.org/10.1109/sp.2013.30

Parno, B, et al. 2013. Pinocchio: Nearly Practical Verifiable Computation. In: *2013 IEEE Symposium on Security and Privacy*. DOI: https://doi.org/10.1109/sp.2013.47

Pettai, M and **Laud, P.** 2015. Combining Differential Privacy and Secure Multiparty Computation. In: *Proceedings of the 31st Annual Computer Security Applications Conference on – ACSAC 2015*. DOI: https://doi.org/10.1145/2818000.2818027

Rajan, A, et al. 2018. Callisto: A Cryptographic Approach to Detecting Serial Perpetrators of Sexual Misconduct. In: *COMPASS*. DOI: https://doi.org/10.1145/3209811.3212699

Ryan, G and **Rohloff, K.** The PALISADE Lattice Cryptography Library. *GitLab/PALISADE*. Available at: https://git.njit.edu/palisade/PALISADE [Accessed November 2, 2018].

Shamir, A. 1979. How to share a secret. *Communications of the ACM*, 22(11): 612–613. DOI: https://doi.org/10.1145/359168.359176

Wang, X, Ranellucci, S and **Katz, J.** 2017. Global-Scale Secure Multiparty Computation. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security – CCS '17*. DOI: https://doi.org/10.1145/3133956.3133979

Yao, AC. 1982. Protocols for secure computations. In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. DOI: https://doi.org/10.1109/sfcs.1982.38

Yao, AC-C. 1986. How to generate and exchange secrets. In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. DOI: https://doi.org/10.1109/sfcs.1986.25

Zahur, S and **Evans, D.** 2015. Obliv-C: A Language for Extensible Data-Oblivious Computation. Available at: https://eprint.iacr.org/2015/1153.

Zhang, Y, Steele, A and **Blanton, M.** 2013. PICCO: A general-purpose compiler for private distributed computation. In: *CCS*. DOI: https://doi.org/10.1145/2508859.2516752

How to cite this article: Alter, G, Falk, BH, Lu, S and Ostrovsky, R. 2018. Computing Statistics from Private Data. *Data Science Journal*, 17: 31, pp. 1–16. DOI: https://doi.org/10.5334/dsj-2018-031

Submitted: 30 October 2016 **Accepted:** 25 October 2018 **Published:** 12 December 2018

]u[*Data Science Journal* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS