Poster: Secure Multi-Party Computation as a Tool for Privacy-Preserving Data Analysis

Samuel Havron
University of Virginia
havron@virginia.edu

***Abstract*—A *Secure multi-party computation* (MPC) protocol allows two or more parties to compute a function on sensitive input data provided by both parties, without revealing anything about the inputs (other than what can be inferred from the revealed output result). Social scientists often work with private datasets that cannot be shared due to legal restrictions and ownership issues, but many interesting studies could be enabled if MPC allows joint data analyses on private data analysis. We are exploring opportunities for using MPC to enable scientific research that would not otherwise be possible. We have developed tools and libraries that enable joint scientific data analyses with private data, and report on preliminary results using MPC to enable linear regression analyses over private data.**

I. Introduction

Many social scientists and researchers need to perform statistical analyses across large, independently-owned datasets for their work, but they are often met with difficulties in obtaining sensitive data and computing results in a safe manner. For instance, an education researcher may be interested in using statistical methods to analyze the relationship between family income and grade point average for a particular school system or collection of school systems. Obtaining such sensitive data is often difficult to do without trusting one or more parties with some of the private input data, as well as considering ownership and legal restrictions on having clear access to private data. One government agency has information about incomes, but cannot share it without violating regulations (and compromising privacy); a local school district has information about students' grades, but cannot correlate that with their family incomes. Secure multi-party computation (MPC) is a protocol which can be used as a tool for carrying out large-scale scientific data analysis in these situations without compromising the privacy of any party's data, or risking it being exposed.

II. Approach

Several approaches to MPC have been developed, including application-specific custom protocols and generic, universal techniques that can be used to compute any function privately. We use universal techniques since it is important that new functions can be developed quickly and without needing new security proofs, and we anticipate needing to incorporate auditing and other functionality into the MPC. The most common universal MPC techniques are based on secret sharing, homomorphic encryption, and garbled boolean circuits using Yao's protocol. We use garbled circuits, which are generally the most scalable and high performance MPC approach currently known.

Obliv-C (http://oblivc.org) is a programming language which allows an application developer to quickly implement scalable, secure MPC protocols, using the language's API or writing specific functionality by extending the language's existing library as well as experimenting with the implementation of library protocols [8]. Obliv-C provides an implementation of Yao's garbled circuit protocol for use with semi-honest adversaries, although it can also be used to implement other protocols. The language is compiled and built on top of the standard C language, allowing for developers to integrate C tools and libraries with Obliv-C seamlessly. This is important for our goals, since we want the programming required to build an MPC protocol to be as little as possible, while allowing programmers to integrate existing tools and libraries.

An excerpt of Obliv-C code below demonstrates input and revealing the encrypted correlation coefficient of a linear regression:

```
for(int i = 0; i < n; i++) {
  oArr[i] = feedOblivInt(iArr[i],party);
}
obliv int orsqr = getOblivRSquared(oArr);
revealOblivInt(&io -> rsqr,orsqr,0);
```

The `obliv` qualifier built into Obliv-Cs type system ensures the variable used is encrypted with the garbled circuit scheme [8]. `iArr` is an integer array which is converted from plain integers into `obliv` integers through the `feedOblivInt()` API, which synchronously executes by both parties. Explicit typing rules and control flow handling can be found in the language's repository (https://github.com/samee/obliv-c). Following the last two lines of this example, the variable is the correlation coefficient of linear regression analysis, obtained through some function call to a method within the MPC protocol that requires the previously converted array of `obliv` values. The result is revealed to specified parties through the API `revealOblivInt()`, which works by cooperatively decrypting the encrypted value into a normal C struct. Revealing an integer stores the result into a struct which is accessible to specified parties ("0" means all parties in this context).

III. Preliminary Results

In our experiments so far, we have implemented a linear regression application as an MPC where data is split between two parties (see repository at https://goo.gl/jQE3QY). One node instance provided independent (x) data points, while the other provided dependent (y) data points; data points used were 32-bit integers, using fixed-point mathematics to convert raw data values into scaled integers, as Obliv-C does not currently support floating point numbers. The output of the computation is the coefficients that result from the linear regression, which are publicly revealed to both parties.

Testing was done using c4.large *Elastic Compute Cloud* (EC2) nodes from *Amazon Web Services* (AWS) [1], which provide high-frequency Intel Xeon E5-2666 v3 (Haswell) processors optimized specifically for EC2, two vCPUs, and 3.75 GiB of DRAM. Two c4.large instances were launched and connected through Obliv-C's API for TCP/IP connections. The instances were both located in the same datacenter in Oregon, with approximately 1 Gbps measured bandwidth between the nodes.

The time needed to execute the MPC between instances appears to scale linearly with the size of the data input; 100K data points finished execution in 12.7 minutes on average, 500K completed in 63.7 minutes, and 1 million data points finished execution in just over 127 minutes on average. Given the relative cost of using a c4.large instance (as of writing, \$0.105 per hour), executing this program over larger inputs, such as 10 million data points, will only incur about \$4.45 between two instances in the estimated runtime of 21 hours. (All numbers are averages over 5 executions between c4.large instances.)

Artificial data was generated for testing the scalability of input size, and considerations to automated data match-ups between two separate datasets were not implemented; the artifical data was presumed to already be matched and sorted properly. To provide a clear example of the utility of Obliv-C for analyzing sensitive datasets, additional data for computation was obtained from the public New York State Department of Health dataset of Hospital Inpatient Discharges from 2011 [6] and is currently being tested. This dataset is publicly available and in a single database, but provides a reasonable model of data that is sensitive and would be kept privately in different subsets kept by different organizations.

The datasets used in our preliminary experiments assume that data is matched, sorted, and comprises of comparable values. In a realistic scenario, it is more likely that the two parties have data for different individuals, and first need to match up the corresponding records. This can be done using private set intersection, which has been efficiently implemented as an MPC [5].

IV. Related Work

A similar approach in taking distinct federal datasets and comparing them was done by Bogdanov et al., where correlations between working hours and failure to graduate on time in Estonia was investigated, matching over 10 million tax records and 500K education records [2]. This analysis utilized the researchers' own framework for secure computation, *ShareMind*, a database and analytics system which uses three somewhat-trusted parties to carry out computations, and uses arithmetic manipulations to implement secure MPC, rather than boolean circuit evaluation [3]. The model where sensitive data is split between three parties that are trusted to not collude or disrupt the protocol enables many efficiencies not possible in standard MPC, but requires three parties in which both participants have a high degree of trust.

Another privacy-preserving approach to analyzing millions of records uses a combination of homomorphic encryption (for linear computations) and Yao circuits (for non-linear computations) in order to compute ridge regression [7]. This approach is designed for many users to send data to a central server that performs the bulk of the computation, in contrast to the two-party model used in MPC. Yet another approach used for secure multiple linear regression relies on protocols based on homomorphic secret sharing, and data which is partitioned across several databases [4].

V. Conclusion

Using MPCs for scientific analysis of large datasets is promising for social scientists and researchers whom would otherwise need to reveal some of one party's input to another party in order to analyze their data or be unable to perform analysis due to legal restrictions and ownership issues. We hope progress in MPC implementation will enable new kinds of scientific data analysis, and envision tools that will make these techniques accessible to social scientists. Future work invites a closer examination of automatic data-matching between separate datasets with private set intersection, improving fixed-point integer conversion for decimal data values used in computation, and other privacy-preserving applications.

References

[1] Amazon Web Services. Amazon EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/.

[2] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and Taxes: a Privacy-Preserving Social Study Using Secure Computation. Cryptology ePrint Archive, Report 2015/1159, 2015.

[3] Dan Bogdanov, S. Laur, and J. Willemson. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *European Symposium on Research in Computer Security (ESORICS)*, 2008.

[4] Rob Hall, Stephen E. Fienberg, and Yuval Nardi. Secure Multiple Linear Regression Based on Homomorphic Encryption. *Journal of Official Statistics*, (4), 2011.

[5] Yan Huang, David Evans, and Jonathan Katz. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? In *Network and Distributed Systems Security Symposium*, 2012.

[6] New York Department of Health. New York State Health Data. https://health.data.ny.gov/.

[7] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-Preserving Ridge Regression on Hundreds of Millions of Records. In *IEEE Symposium on Security and Privacy*, 2013.

[8] Samee Zahur and David Evans. Obliv-C: A Lightweight Compiler for Data-Oblivious Computation. Cryptology ePrint Archive, Report 2015/1153, 2015. http://oblivc.org.