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 each party, without revealing anything about the inputs (other than what can be inferred from the revealed output result). The use of MPCs as a tool for large-scale scientific data analysis can give many researchers and social scientists the ability to work with private datasets and conduct studies that would otherwise be difficult to do securely without revealing some input data to one or more parties of the computation. Using the Obliv-C language, researchers can develop their own application-specific MPCs and quickly deploy them for use in privacy-preserving data analysis. A demonstration of the scalability and performance of Obliv-C for scientific data analysis and a discussion of related work highlights the utility of the language for researchers who have little to no experience in cryptographic protocols.**

I. Introduction

Many social scientists and researchers need to perform statistical analyses across large, distinct 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 a student's family income and the student's 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. Secure multi-party computation (MPC) is a protocol which can be used as a tool for carrying out large-scale scientific data analysis.

Most current implementations of MPC work by executing instructions in a *data-oblivious* manner, where the control flow of the program is independent of the inputs provided by each party and the program executes without any knowledge of the cleartext data it is operating on. This effectively creates a black box, such that semi-honest adversaries cannot gain addition insight about the source data, receiving only the computated results. Several different approaches to MPCs exist, with tradeoffs in security, speed, and involvement of each party. The most common are based on secret sharing, homomorphic encryption, and garbled boolean circuit construction. Secret sharing, which scatters private inputs onto multiple different servers and uses a distributed computing protocol to safely reach a result, assumes a majority of the servers do not collude with each other. The homomorphic encryption model generally executes computations over encrypted data and relies on a trusted third party to reveal the final decrypted result, without revealing any intermediary results. The garbled circuit approach allows for a final result to be obtained without revealing any other data about any party involved. Many researchers and social scientists require MPCs in order to analyze datasets that are both large and sensitive, and they often need to rely on using an underlying secure computation software framework to execute such protocols. The Approach section discusses the utility of the Obliv-C language, which is based on the garbled circuit evaluation MPC.

```
obliv int orsqr = fixed_div(osqr(ocov),ostddevs);
revealOblivInt(&io -> rsqr,orsqr,0);
revealOblivInt(&io -> m,om,0);
revealOblivInt(&io -> b,ob,0);
```

Fig. 1. Code snippet showing *obliv* qualifier and reveal API. Revealing an integer stores the result into a struct which is accessible to specified parties.

II. Approach

Obliv-C is a programming language which allows an application developer to quickly implement scalable, secure MPC protocols, using the languages API or writing specific functionality by extending the language's existing library as well as experimenting with the implementation of library protocols [7]. 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. Obliv-C implements secure MPC through optimizations of Yao's garbled circuit protocol for use with semi-honest adversaries. Using this language to write applications that can analyze large, privacy-preserving datasets results of building one such application for linear regression analysis are shown in Preliminary Results.

The goal of using Obliv-C as a framework for secure computation is to demonstrate its performance capabilities and ease of use to developers whom have little knowledge of cryptography or circuit structures, but would greatly benefit from using secure MPCs to carry out privacy-preserving dataset analysis (perhaps for social science research). An example of Obliv-C code which calculates and reveals results of a linear regression are seen in Figure 1.

III. Preliminary Results

A. Scalability

A linear regression data analysis program was developed to test the scalability and speed of Obliv-C as a tool for

implementing MPC programs. Testing was done using c4.large *Elastic Compute Cloud* (EC2) nodes from *Amazon Web Services* (AWS) [1], which feature 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 via Oblivious Transfer protocol. The instances were both located in the same cloud cluster in Oregon; exploring network latency impositions with secure MPCs is also an area of interest.

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 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.

Artificial data was generated for testing the scalability of input size, and considerations to automated data matchups 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 [2]. Comparisons between fields such as "Length of Days stayed" and "Total Costs" over approximately 2.6 million data points are currently being tested. This dataset is particularly amenable to analysis, as all data is already matched properly and each data value is a comparable number.

B. Discussion

All numbers are averages over 5 executions between c4.large instances. The datasets used in Preliminary Results assume that data is matched, sorted, and comprises of comparable values. Using private set intersection to assist matching data with a unique identifier and automatically filtering out incompatible data points is considered for future work.

IV. Related Work

A similar approach in taking distinct federal datasets and comparing them is seen in Dan 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 [4]. This analysis utilized the researchers' own framework for secure computation, *ShareMind*, a database and analytics system which almost exclusively uses three-parties to carry out computations, and uses arithmetic manipulations to implement secure MPC, rather than boolean circuit evaluation [3].

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 [6]. This approach is designed for many users to send data to a central server called the *Evaluator*, in contrast to the primarily two-party model presented in the Obliv-C language. 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 [5].

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. The Obliv-C language is particularly well-suited for implementing such scalable, sensitive data analysis between two or more parties, and shows the practicality of using secure MPCs for processing large datasets. 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] https://aws.amazon.com/ec2/instance-types/.

[2] https://health.data.ny.gov/.

[3] Dan Bogdanov, S. Laur, and J. Willemson, *Sharemind: A framework for fast privacypreserving computations*, In 13th European Symposium on Research in Computer Security (ESORICS), volume 5283 of LNCS, pages 192206. Springer, 2008. http://sharemind.cyber.ee.

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

[5] Rob Hall, Stephen E. Fienberg, and Yuval Nardi, *Secure multiple linear regression based on homomorphic encryption*, J. Official Statistics, vol. 27, no. 4, 2011.

[6] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, Nina Taft, *Privacy-Preserving Ridge Regression on Hundreds of Millions of Records*, in 2013 IEEE Symposium on Security and Privacy (S&P 2013), pp. 334-348, IEEE Computer Society, 2013, http://www.technicolorbayarea.com/papers/2013/NWIJBT13garbled.pdf.

[7] Samee Zahur and David Evans, *Obliv-C: A Language for Extensible Data-Oblivious Computation*, Cryptology ePrint Archive, Report 2015/1153, 2015, http://eprint.iacr.org/.