Towards Practical Multiparty Computation in Large Networks

Mahdi Zamani
University of New Mexico
zamani@cs.unm.edu

Mahnush Movahedi
University of New Mexico
movahedi@cs.unm.edu

Jared Saia
University of New Mexico
saia@cs.unm.edu

Abstract

We describe a general Multi-Party Computation (MPC) protocol for arithmetic circuits that is secure against a computationally-bounded malicious adversary statically corrupting up to a 1/10 fraction of the parties. The protocol requires each party to send an average of $O(\frac{m}{n}\log^3 n)$ bits, and compute $O(\frac{m}{n}\log^4 n)$ operations in a network of size n, where m is the size of circuit. This is achieved by increasing latency from constant to $O(d)$, where d is the depth of the circuit. Our protocol has a setup phase that is independent of the circuit and relies on Threshold Fully Homomorphic Encryption (TFHE). The setup requires each party to send $\tilde{O}(\kappa^2)$ messages and to compute $\tilde{O}(\kappa^2)$ operations. We provide results from microbenchmarks conducted over a sorting network showing that our protocol may be practical for deployment in large networks. For example, we consider a network of size 2^{25} (over 33 million) where each party has an input item of size 20 bytes. To securely sort the items, our protocol requires each party on average to send 5 kilobytes per item sorted.

1 Introduction

In secure Multi-Party Computation (MPC), a set of n parties, each having a secret value (input), want to compute a common function represented as a circuit over their inputs, without revealing the inputs to each other. In the last three decades, a large body of work has been devoted to designing MPC protocols for the active adversarial setting. Unfortunately, most of these protocols are inefficient for the case where the number of parties is large.

One approach to solve MPC is to use *Fully Homomorphic Encryption (FHE)* (e.g., Asharov et al. [1]). In this approach, each party first encrypts its input under FHE and send the ciphertext to others. The parties then evaluate the desired function on the encrypted data and finally perform a distributed decryption on the final encrypted data to get the results. Due to the homomorphic properties of FHE, the decrypted value is the correct evaluation of the function over the inputs. FHE-based MPC protocols are usually optimal in terms of communication cost, however computation cost is usually expensive. Unfortunately, current FHE schemes are very slow and can only evaluate circuits of small depth.

In this paper, we specifically address the problem of MPC for any-depth circuit and when the number of parties is large. We believe this problem is of increasing importance with the growth of modern networks. For example, how can peers in BitTorrent auction off resources without hiring an auctioneer? How can we design a decentralized Twitter that enables provably anonymous broadcast of messages? How can we perform data mining over data spread over large numbers of machines?

We also provide results from microbenchmarks for the problem of secure Multi-Party Sorting (MPS). MPS is useful in many applications such as anonymous communication [4] and privacy-preserving statistical analysis [9]. It is often important for these applications to be run among *many* parties. For example, MPS is a critical component of communications algorithms that could enable the creation of large anonymous microblogging services without requiring trusted authorities (e.g., an anonymous Twitter).

1.1 Our Results

Consider a network of n parties, where there is a private and authenticated communication channel between every pair of parties. The following main theorem is secure under the τ-*Strong Diffie-Hellman (*τ*-SDH)* and τ-*polynomial Diffie Hellman (*τ*-polyDH)* hardness assumptions, where $\tau \leq (1/3 - \varepsilon)N$.

Theorem 1. *Let f be any deterministic function over n inputs in $\mathbb{Z}_p$ for prime $p = \mathsf{poly}(n)$, and $\mathscr{C}$ be a circuit with m gates and depth d that computes f. There exists an n-party synchronous protocol that securely computes f, tolerates up to $t < (1/10 - \varepsilon)n$ malicious parties, and has the following properties:*

- *Each party sends $O\left(\frac{m}{n}\log^3 n\right)$ messages of size $O(\log p)$ bits and computes $O\left(\frac{m}{n}\log^4 n\right)$ operations.*
- *The protocol has a setup phase that is secure in the CRS model and requires each party to send $\tilde{O}(\kappa^2)$ bits and compute $\tilde{O}(\kappa^2)$ operations.*
- *The latency of the protocol is $O(d)$.*

2 Our Approach

Our goal is to reduce both communication and computation complexities, and to this end we make a trade-off between these complexities and latency. We are inspired by the unconditionally-secure MPC algorithm of Dani et al. [8]. However, we make extensive use of cryptographic tools in order to reduce costs in practice. We reduce the costs further by performing local communications in polylogarithmic-size groups of parties called *quorum*s, where the number of adversary-controlled parties in each quorum is a certain fraction. The quorums are created in a one-time setup phase that is secure in the *Common Reference String (CRS)* model. The setup phase uses the quorum building algorithm of Santoni et al. [7] and the fully homomorphic encryption scheme of Brakerski et al. [6] to generate a number of parameters required for our protocol. Our online phase combines the circuit randomization technique of Beaver [3] and the efficient verifiable secret sharing scheme of Kate et al. [10] to perform fast computations on secret-shared values.

In our protocol, each gate of the circuit is assigned a quorum Q and the parties in Q are responsible for computing the function associated with that gate. Then, they send the result of this computation to any quorums associated with gates that need this result as input. Let Q' be one such quorum. It is necessary to securely send the output from Q to Q' without revealing any information to any individual party (or to any coalition of adversarial parties). This is a technically challenging problem.

Dani et al. [8] handle this problem by masking the result in Q and unmasking the result in Q'. This method is expensive in practice since parties in Q' need to reconstruct the masks jointly for each input. Boyle et al. [5] handle this problem by sending all the inputs to only one quorum which does all of the computation. This results in large computation and communication costs for parties in that quorum.

We introduce a new solution for this problem. We let each party in Q hold shares of the inputs to the gate associated with Q. Using homomorphic property of these shares, each party can run the gate function on their input to evaluate a share of the output. It is essential that parties in each quorum have a method to send the shares of the result to Q'. These shares cannot be the same shares because this would leak information to the adversary after multiple steps. Thus, in our algorithm, parties in Q jointly generate a *fresh* re-sharing of the output of the gate for Q'. Performing this fresh re-sharing correctly is one of the main technical challenges of our algorithm.

Microbenchmarks. Multi-Party Sorting (MPS) can be performed efficiently using sorting networks. A *sorting network* is a network of *comparators*. Each comparator has two input wires and two output wires. When two values enter a comparator, it outputs the lower value on the top output wire, and the higher value on the bottom output wire. Batcher [2] proposes an efficient and simple sorting network with depth $1/2\log n(1+\log n) = O(\log^2 n)$. In our simulations, we use Batcher's sorting network over n inputs. Each input is provided by a different party.

The circuit is computed in $\mathbb{Z}_p$ for a 160-bit prime p, with about 80-bit security. We set parameters to ensure error probability of at most 10^{-5} for the quorum formation algorithm. We ran the setup protocol once and then used the setup information to sort 100 vectors of random values, i.e., the online protocol was repeated 100 times with the same setup parameters. Consider n parties each with an arbitrary input from $\mathbb{Z}_p$. Let s be the total number of bits sent in the setup phase, and c be the total number of bits sent in the online phase for sorting 100 vectors. Let a_n be the average number of bits sent by each party for each sorted element received, in a network of size n. This is calculated from $a_n = \big(s+c\big)/100n^2$.

We repeated the experiment for network sizes ranging from $n = 2^{10}$ to $n = 2^{30}$. Figure 1 depicts the log-log plot for a_n as n varies. In both plots, we give the average number of kilobytes sent per party for each sorted element. In the left plot, we give an average that includes the entire setup phase. In the right plot, we give an average that

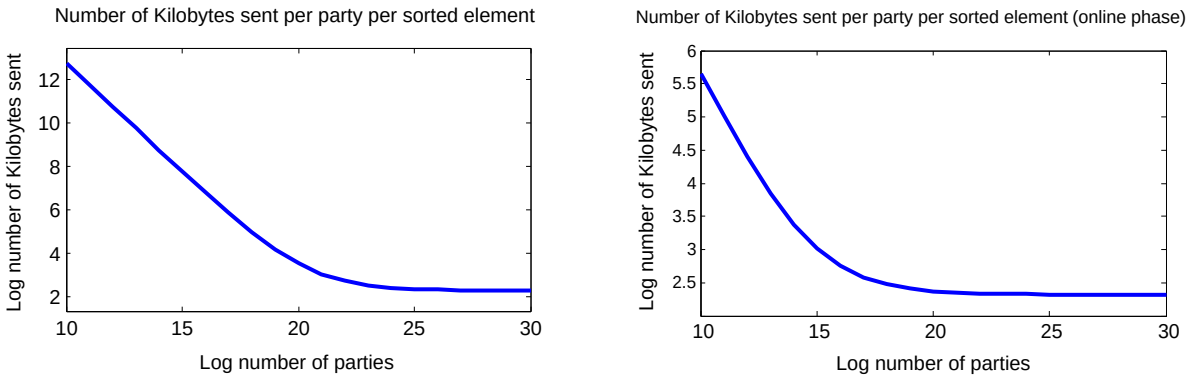

Figure 1: Communication cost for multiparty sorting

does not include this the setup phase. For example, for sorting a vector of 2^{25} elements of $\mathbb{Z}_p$ in a network of size $n = 2^{25}$ (over 33 million parties), each party sends an average of 5 kilobytes per each element of the sorted vector, as the left plot shows. The one-time setup for such a network requires an average of 21 kilobytes of communication per party. After this setup, the communication for the sorting of one vector is an average of 5 kilobytes per party.

3 Open Problems

Several open problems remain to be solved in our protocol. First, can we improve performance even further by detecting and blacklisting parties that exhibit adversarial behavior? We believe that such an approach could lead to significant practical improvements. Second, can we adopt our results to the asynchronous model of communication? We believe that this is possible for a suitably chosen upper bound on the fraction of faulty parties. Finally, can we adopt our results to a model that is more in line with fully-distributed peer-to-peer networks? In such networks, it is unlikely that each party knows the identities of every other party.

References

[1] ASHAROV, G., JAIN, A., LÓPEZ-ALT, A., TROMER, E., VAIKUNTANATHAN, V., AND WICHS, D. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology – EUROCRYPT 2012*, vol. 7237 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 483–501.

[2] BATCHER, K. E. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference* (New York, NY, USA, 1968), AFIPS '68 (Spring), ACM, pp. 307–314.

[3] BEAVER, D. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology – CRYPTO '91*, J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1991, pp. 420–432.

[4] BERMAN, R., FIAT, A., AND TA-SHMA, A. Provable unlinkability against traffic analysis. In *Financial Cryptography*, A. Juels, Ed., vol. 3110 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 266–280.

[5] BOYLE, E., GOLDWASSER, S., AND TESSARO, S. Communication locality in secure multi-party computation: how to run sublinear algorithms in a distributed setting. In *Proceedings of the 10th theory of cryptography conference on Theory of Cryptography* (Berlin, Heidelberg, 2013), TCC'13, Springer-Verlag, pp. 356–376.

[6] BRAKERSKI, Z., GENTRY, C., AND VAIKUNTANATHAN, V. Fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (2012), ITCS '12, ACM, pp. 309–325.

[7] BRAUD-SANTONI, N., GUERRAOUI, R., AND HUC, F. Fast Byzantine agreement. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing* (New York, NY, USA, 2013), PODC '13, ACM, pp. 57–64.

[8] DANI, V., KING, V., MOVAHEDI, M., AND SAIA, J. Quorums quicken queries: Efficient asynchronous secure multiparty computation. In *Distributed Computing and Networking*, M. Chatterjee, J.-n. Cao, K. Kothapalli, and S. Rajsbaum, Eds., vol. 8314 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014, pp. 242–256.

[9] DU, W., AND ATALLAH, M. J. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 Workshop on New Security Paradigms* (2001), NSPW '01, ACM, pp. 13–22.

[10] KATE, A., ZAVERUCHA, G. M., AND GOLDBERG, I. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology – ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security* (2010), vol. 6477 of *Lecture Notes in Computer Science*, Springer, pp. 177–194.