The study of modern cryptography began nearly twenty years before the introduction of the World Wide Web, and nearly thirty years before the invention of the smart phone. At that time, the most immediate questions in cryptography dealt with sending private, authenticated messages. Today, our use of the Internet is much more varied; we upload our data and outsource our computation to remote, untrusted servers, giving them control over the very information we wish to protect. I have two main focus areas in research. First, I strive to explore the theoretical foundations of cryptography as designed for a modern environment, where simple encryption and digital signatures no longer suffice. Secondly, I aim to find examples where yesterday's theory can be implemented today. Specifically, I aim to develop *practical* cryptography, with the goal of impacting other branches of computer science research, and society at large.

Much of my research has been in the area of secure computation, which enables users with private data to jointly compute some function of their choice, while maintaining the privacy of their input, along with other important security properties. Secure computation could potentially enable rich collaboration while maintaining privacy. For example, it could allow governments to compare intelligence without opening their files, or hospitals to compare health records without violating the rights of their patients. Through secure computation, users could enable data service providers to mine their text messages and phone locations, facilitating federated learning that benefits large populations, while receiving a provable guarantee that their own data is never exposed.

Although the theoretical results are more than thirty years old, secure computation is only now beginning to have direct impact on society. The first implementation of secure computation was published in 2004, and the system supported 30 circuit gates per second [14].[1] Today, there are implementations computing as many as 2.5 million gates per second [15], and frameworks supporting a variety of programming languages [16, 17, 18, 19, 20, 21, 22], as well many systems that optimize particular classes of computations [23, 24]. Due to these improvements, we are finally seeing wider deployment of the technology. In the last few years, it has been used annually by the Boston Women's Workforce Council to compute the gender pay gap using salary data from several hundred companies [26]. These companies would not have participated in such a study without the privacy guarantees offered by secure computation. Google has started using secure computation to help advertisers compute the value of their ads, and they will soon start using it to securely construct machine learning classifiers from mobile user data [27]. Since the start of the Covid-19 viral pandemic, there have been a few attempts to use secure computation for secure contact tracing, notifying citizens if they have been exposed to the virus, without anyone learning where they have been, or with whom they have interacted. In May, 2017, Senator Ron Wyden wrote an open letter to the commission on evidence-based policymaking urging that secure computation be employed by "agencies and organizations that seek to draw public policy related insights from the private data of Americans [28]." Senator Wyden's office has sponsored a bill calling for the use of secure computation to help students evaluate the quality of different university degrees by privately combining data from the IRS, the Census and the Department of Education [29].

[1]Most results in secure computation require us to represent the computation as a circuit of addition and multiplication gates over a ring or field. For this reason, the cost is usually measured in gates.

1 Secure Computation on Large Datasets

A minimal requirement in guaranteeing the privacy of user data is that the protocol execution is data oblivious. That is, the parties computing on the encrypted data should see a fixed sequences of memory accesses, independent of the data content. General solutions provide this guarantee by operating in the circuit model of computation, but this is prohibitively slow for certain computations, since it is inherently linear in the input size, and often much worse. As a postdoc, I helped build the first system for secure computation in the RAM model. This allowed us to demonstrate sub-linear runtime for many natural computational problems, providing big improvements in secure computation over big datasets [30]. This work has been cited 150 times since 2012, and has sparked a line of research on secure computation in the RAM model. In my own followup work, I built a system for securely emulating the MIPS architecture, supporting the secure execution of arbitrary code that was written in a language that can compile to MIPS [1].

More recently, together my PhD student, Sahar Mazloom, I introduce a new model for securely computing on large data [2]. My prior work required the use of oblivious RAM, which facilitates sub-linear computation through the use of fully oblivious memory access patterns, but still introduces substantial overhead. Taking a new direction, we relax the requirement that our computation is data oblivious, and instead allow some information to leak to the computing parties. However, we prove a bound on the leakage, showing that it preserves *differential privacy* [31] for the users that contribute data. Our novel security model combines two important and fruitful lines of research. It allowed us to demonstrate a 20X improvement in computation time for some important machine learning tasks, including recommendation systems, page-rank, histograms, and others. This work raises interesting theoretical questions about whether *differentially oblivious* algorithms, which are nearly indistinguishable on similar inputs, are more efficient, fundamentally, than fully oblivious algorithms.

With my postdoc, Samuel Ranellucci, I developed a new MPC protocol for four parties, with security against a single malicious actor. The protocol has extremely low communication, which makes it ideally suited for a setting where many thousands or millions of users privately outsource a computation to four entities. Leveraging and extending this result to support fixed point computation, and by designing an efficient four-party oblivious shuffle, together with my students Sahar Mazloom, and Phi Hung Le, and my postdoc Samuel Ranellucci, we further improved the performance of my work on differentially oblivious MPC [2]. For example, when performing sparse matrix factorization on 1 million user inputs, the best fully oblivious construction [32] requires about 13 hours of computation; our work from 2018 requires about 2 hours of computation; our recent result [4] performs the same computation in about 25 seconds.

[1] X.S. Wang, S. D. Gordon, A. McIntosh, J. Katz, Secure Computation of MIPS Machine Code. In *ESORICS*, 2016.

[2] S. Mazloom, S. D. Gordon, Differentially Private Access Patterns in Secure Computation. In *ACM Conference on Computer and Communications Security*, 2018.

[3] S. D. Gordon, S. Ranellucci, X. Wang Secure Computation with Low Communication From Cross-checking. In *IACR, Asiacrypt*, 2018.

[4] S. Mazloom, P.H. Le, S. Ranellucci, and S. D Gordon Secure Parallel Computation on National Scale Volumes of Data. In *Usenix Security Symposium*, 2020.

Private set intersection: One problem in secure computation that has received targeted attention is *private set intersection*. Here, two (or more) parties each hold a private data set, and they wish to compute the intersection, or possibly to compute some function of the intersection, without revealing their input. Together with my student, Phi Hung Le, and my postdoc, Samuel Ranellucci, we designed a new construction for computing on the intersection, relying on an untrusted third party to improve the asymptotic performance [5]. By leaking the size of the intersection to this third party, we can shave a log factor off the communication complexity. This has led to two follow-up works with my student, Phi Hung. In the first, we provide new improvements in the case where one party has a much smaller input set than the other [6]. This is especially useful in applications such as *contact discovery* in which a user wants to find out from an untrusted central server which of their phone contacts are using the same application as them. In the second work, which is still in progress, we use recent advances in zero knowledge proofs to redesign old protocols for set intersection [7].

[5] P.H. Le, S.Ranellucci, S. D. Gordon Two-party Private Set Intersection with an Untrusted Third Party. In *ACM Conference on Computer and Communications Security*, 2019.

[6] P.H. Le, S. D. Gordon Malicious Secure Private Set Intersection From Vector OLE. In submission.

[7] C. Hazay, P.H. Le, S. D. Gordon Private Set Intersection from polynomial evaluation, revisted. In progress.

2 Secure Computation For Large Numbers of Parties

While it often suffices to privately outsource large-scale computations to a handful of parties, ideally our data would never leave our devices. In a recent line of work, I have been pushing the boundaries on the number of parties that can efficiently engage in a secure computation. When trying to scale secure computation to a large number of parties that compute over a wide area network, reducing latency requirements, bandwidth requirements, and providing robustness to failures all become essential. Most existing constructions of multi-party computation have a round complexity that grows linearly with the depth of the circuit that is being computed, and communication complexity that grows linearly in the size of the circuit. Furthermore, many protocols do not guarantee output delivery, and, what's worse, they do not even allow the honest parties to identify the faulty party, making it impossible to prevent failures in the future. Until recently, the largest reported experiments involved a few hundred parties [33].

Three of my recent publications help push the boundary of what is practically feasible. Together with my postdoc, Samuel Ranelluci, I constructed a new protocol for large-scale secure computation with multiple important security guarantees [8]. Our construction has a communication cost that is only a constant factor worse that the size of the program description (with only an additive term that grows with the number of parties), and guarantees output delivery as long as a minority of parties are corrupt. While it is already known how to provide either one of these properties, our protocol is the first to offer them simultaneously. In another work, we explore the use of MPC in the Tor network [9]. Tor is an anonymous communication network that aims to help citizens living under oppressive governments to bypass Internet censorship. Tor administrators would greatly benefit from network traffic analysis, but clearly must gather statistics while ensuring the privacy of the user. There are almost 7000 Tor routers that help route traffic in the network. We designed, implemented and tested new MPC protocols for this setting, showing concretely, for the first time, the cost of using MPC at this scale. Additionally, we provided several new approaches for making such computations robust to failures, which is crucial when engaging so many parties at one time. Finally, in a new work that is currently in submission, we construct a new protocol with per-party communication complexity that reduces as more participants join [10]. Although there are other protocols with this property, ours has the best complexity to date. The protocol is highly efficient in practice, and could reasonably be used by millions of parties to securely compute on private data. Our estimate is that it would support 320 million gates per second when there are a million participants, providing 100X improvement over what is currently the best implementation.

In all three of the works just described, the round complexity grows with the depth of the circuit being computed. This can easily grow to be several hundred rounds of communication, raising the possibility that latency will become the bottleneck, rather than the communication complexity. In a slightly older work, we provided the first 3-round protocol with a guaranteed output delivery [11]. This protocol is interesting from a theoretical perspective, but the computational costs are quite high. Constructing a constant round protocol that guarantees output and scales, practically, to thousands of parties remains a very interesting open question.

[8] D. Genkin, S. D. Gordon, S. Ranellucci, Best of Both Worlds in Secure Computation, with Low Communication Overhead. In *ACNS*, 2019.

[9] R. Wails, A. Johnson, D. Starin, A. Yerukhimovich, and S. D. Gordon, Stormy: Statistics in Tor by Measuring Securely. In *ACM Conference on Computer and Communications Security*, 2019.

[10] S. D. Gordon, D. Starin, and A. Yerukhimovich, The More The Merrier: Reducing the Cost of Large Scale MPC In *Submission*, 2020.

[11] S. Dov Gordon, Feng-Hao Liu, Elaine Shi, Constant-Round MPC with Fairness and Guarantee of Output Delivery. In *CRYPTO*, 2015.

3 Differentially Oblivious Computation

My work leveraging differential privacy as a security relaxation in secure computation raises some very interesting theoretical questions, some of which were mentioned briefly above. In two ongoing works I have been investigating whether this relaxation might be useful in the specific case of *oblivious shuffling*. In an oblivious shuffle, users each provide a single private input, and after executing a protocol, they each receive a random permutation of those values, without learning the permutation. This is useful as a building block for many security applications, such as anonymous messaging, the evaluations of privacy preserving statistics, and currency mixing.

Together with my student, Mingyu Liang, my colleague Foteini Baldimtsi, and her student, Ioanna Karantaidou, I introduce the idea of a *differentially oblivious*— shuffle, in which the permutation does not remain completely oblivious, but rather it guarantees that neighboring permutations have similar probability weights, even after the protocol execution leaks some information. I then demonstrate that for the application of mixing the coins of a crypto-currency, this relaxation gives asymptotic improvement in round complexity over other known solutions [12]. Motivated by this result, I have been looking at the communication complexity of implementing this relaxed shuffle, together with my student Mingyu Liang and my postdoc Jiayu Xu. This work is still in progress, but we believe we will soon demonstrate a lower bound, proving that asymptotic improvement over fully oblivious shuffles is impossible [13].

[12] F. Baldimtsi, S. D Gordon, I. Karantaidou, M. Liang, and M. Varia, Differentially Private Mixing forCryptocurrencies. In Submission, 2020.

[13] S. D Gordon, J. Katz, M. Liang, and J. Xu, Differentially Oblivious Shuffling. In Progress, 2020.

4 Future Research

Trading Security for Efficiency in Secure Computation. My work leveraging differential privacy as a security relaxation demonstrates that, for certain computations, we can improve on existing results, both asymptotically and concretely, if we are willing to weaken our security requirements. As we approach theoretical limits on communication and computational costs in secure computation, such relaxations may play an important role in facilitating the application of the research to increasingly large volumes of data. The precise value of allowing differentially private leakage is still very unclear. I intend to investigate the power of this relaxation, exploring where it allows us bypass known lower bounds, and looking for new lower bounds that apply in this model. Additionally, I intend to explore open questions relating to several other known security relaxations, such as *covert security*, which does not prevent malicious behavior, but ensures that it is caught, and publicly exposed, with high probability.

Scaling to Larger Numbers of Parties. Above I described several of my results that have helped extend secure computation to support larger numbers of participants. Several important questions remain, mainly related to ensuring fault tolerance, and preventing malicious denial of service. From a theoretical perspective, we know that guaranteeing output is possible, if less than half of the network fails. However, when looking at the concrete performance of such protocols, they do not compare against those that abandon such guarantees. I plan to close this gap, both by inventing new protocols that meet the strongest security guarantees, and by developing new security models that provide weaker, but sufficient guarantees. My work on using MPC in the Tor network provides a nice starting point for developing such a relaxation. While we don't formalize this there, in that work we treat honest failures separately from malicious deviations, providing resilience against the first, and aborting in the face of the latter. Formalizing such a notion and developing new protocols to satisfy this definition will help scale MPC to larger numbers of parties.

5 Teaching Statement

My primary aim in education is to make formal proof techniques accessible and exciting for students of all backgrounds and interests. I have always been drawn to computer theory, and I am most motivated by research that helps resolve what is and is not possible in cryptography, and computation more generally. Nevertheless, since completing my PhD I have made a conscious decision to study more applied problems as well, because I recognized that the practical application of secure computation would appeal to a wider swath of graduate students than my earlier, more theoretical research. Only a small fraction of students in computer science wish to be theorists, and a large fraction of such students go to a small fraction of our universities. By focusing on new *theories* that make secure computation more *practical*, I aim to make theoretical cryptography accessible to students from a wide array of backgrounds. My first two graduate students have backgrounds in ML and computer vision, and both were introduced to secure computation through the applied aspects of the research [2, 3, 4]. By thinking about efficiency in a hands-on manner, while maintaining privacy and security, the students quickly developed an appreciation for the theory.

At the same time, my research remains rooted in the theoretical aspects of secure computation. The theory behind the first of these results [2] attracted my third PhD student, who has a background in computer theory, to join George Mason. In his first year, he has begun doing excellent research on several of the more theoretical problems described above. Focusing on the intersection of theory and application allows me to reach a broad audience, while still engaging with the theoretical questions that continue to motivate me.

When teaching in the classroom, it is challenging to excite a broad audience about computer theory. Part of the challenge is that it is difficult to allow students to experiment with formalization in a hands-on manner, the way so many educators in other areas of computer science have successfully done. As a result, theorists have mostly ignored modern classroom techniques that directly involve the students, relying instead on traditional white-board presentations. In teaching *Formal Methods and Models* (CS 330), and *Introduction to Cryptography* (CS 499), I have been experimenting with a new approach. I have created 15 short videos, each 5-10 minutes in length, capturing my computer screen as I walk through the lines of a formal proof. Student feedback has been extremely positive, and I have added more videos in direct response to requests. Once I have a sufficiently large library of these videos, I will assign them as homework, prior to the relevant lecture, so that students come to class already exposed to the relevant techniques, and more able to follow and inquire as I do similar proofs on the board, in the classroom. Ultimately, this will also free-up classroom time for group efforts at proof writing. The aim is to design a "semi-inverted classroom," which does not replace the white-board proof, but supplements it, allowing each student to go at their own pace at home, facilitating greater engagement while I am at the board, and allowing students to experiment with formal techniques in a group setting.

Future Teaching Plans. While the use of videos will help students to better understand formal proof techniques, it still leaves computer theory looking very different from other areas of computer science. In the future, I hope to develop new teaching techniques that will allow computer science students to apply their programming skills in theory classes. I am especially excited about one technique in particular. In cryptography, students are often asked to prove that an encryption scheme is secure by showing that there does not exist any polynomial-time adversary that can "win" in a specified security game. Proofs of non-existence are subtle, and it would be nearly impossible to automate the process of checking these proofs, as I mentioned previously. However, at least as often, students are asked to prove that some encryption scheme is *insecure*, by demonstrating the *existence* of an algorithm that does win in the specified security game. Verifying that their proposed algorithm wins such a game could certainly be automated. I plan to develop a framework that will allow instructors in cryptography to specify encryption schemes (as well as other primitives, such as message authentication codes, pseudorandom functions, etc.), and that will then engage student algorithms in the prescribed security game, testing whether they indeed win in that game. The test is a simple statistical test, and can be easily translated into a score: student code would only need to be submitted or analyzed if the instructor wishes to prevent code sharing. Such a framework will make the assignments more fun for the students, while also helping to scale the grading to larger classes. I believe that such a framework could find use in many universities around the world.

Additional References

[14] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[15] J. Furukawa, and Y. Lindell, Two-Thirds Honest-Majority MPC for Malicious Adversaries at Almost the Cost of Semi-Honest, In *ACM Conference on Computer and Communications Security*, 2019

[16] C. Liu, X.S. Wang, K. Nayak, Y. Huang, E. Shi, Oblivm: A programming framework for secure computation. In *IEEE Security & Privacy*, 2015.

[17] A. Holzer, M. Franz, S. Katzenbeisser, H. Veith, Secure two-party computations in ANSI C. In *ACM Conference on Computer and Communications Security*, 2012.

[18] Y. Zhang, A. Steele, M. Blanton, PICCO: a general-purpose compiler for private distributed computation. In *ACM Conference on Computer and Communications Security*, 2013.

[19] D. Demmler, T. Schneider, M. Zohner, ABY—A framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[20] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: tool for automating secure two-party computations. In *ACM Conference on Computer and Communications Security*, 2010.

[21] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, 2008.

[22] University of Bristol, Multiparty computation with SPDZ online phase and MASCOT offline phase `https://github.com/bristolcrypto/SPDZ-2`

[23] V. Kolesnikov, R. Kumaresan, M. Rosulek, N. Trieu, Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *ACM Conference on Computer and Communications Security*, 2016.

[24] J. Doerner, D. Evans, a. Shelat, Secure Stable Matching at Scale. In *ACM Conference on Computer and Communications Security*, 2016.

[25] Boston Women's Workforce Council Report, 2017. Retrieved August, 2020 from https://www.boston.gov/sites/default/files/document-file-01-2018/bwwc_2017_report.pdf.

[26] B. Kreuter, Secure multiparty computation at google. In *Real World Crypto*, 2017. Retrieved August, 2020 from https://www.youtube.com/watch?v=ee7oRsDnNNc.

[27] R. Wyden, Letter to commission on evidence-based policymaking. Retrieved August, 2020 from https://www.wyden.senate.gov/download/?id=B10146F5-EDEB-4A2C-AD5E-812B363EE0DC&download=1, 2017, U.S. Senate.

[28] R. Wyden, M. Rubio, and M. Warner Retrieved August, 2020 from https://www.congress.gov/bill/116th-congress/senate-bill/681.

[29] S. D. Gordon, J. Katz, V. Kolesnikov, F. Krell, T. Malkin, M. Raykova, and Y. Vahlis, Secure two-party computation in sublinear (amortized) time. In *ACM Conference on Computer and Communications Security*, 2012.

[30] Cynthia Dwork, Aaron Roth, The Algorithmic Foundations of Differential Privacy. In *Foundations and Trends in Theoretical Computer Science*, 2014.

[31] K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, E. Shi, GraphSC: Parallel Secure Computation Made Easy. IEEE Symposium on Security and Privacy 2015: 377-394

[32] X. Wang, S. Ranellucci, and J. Katz Global-Scale Secure Multiparty Computation. ACM Conference on Computer and Communications Security, 2017