8

Conclusion

In the past decade or so, MPC made dramatic strides, developing from a theoretical curiosity to a versatile tool for building privacy-preserving applications. For most uses, the key metric is cost, and the cost of deploying MPC has declined by 3–9 orders of magnitude in the past decade.

The first reported 2PC system, Fairplay (Malkhi *et al.*, 2004), executed a 4383-gate circuit in the semi-honest model, taking over 7 seconds on a local area network at a rate of about 625 gates per second. Modern 2PC frameworks can execute about 3 million gates per second on a 1Gbps LAN, and scale to circuits with hundreds of billions of gates.

Cost improvements for malicious secure MPC have been even more dramatic. The first substantial attempt to implement malicious secure generic MPC was Lindell *et al.* (2008), intriguingly titled "Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries". It reports malicious evaluation of the 16-bit comparison circuit, consisting of fifteen 3-to-1 gates and one 2-to-1 gate, in between 135 to 362 seconds depending on the security parameter settings. This evaluation rate corresponds to about 0.13 gates per second. An implementation of the authenticated garbling scheme (Section 6.7) reports malicious security 2PC at over 0.8 million gates per second on a 10Gbps LAN (Wang *et al.*, 2017b). This corresponds to over a

6 million *factor* performance improvement in under a decade! With honest-majority malicious 3PC the performance that can be achieved is even more remarkable (Section 7.1.2). Araki *et al.* (2017) report a 3-machine 20-core cluster that can evaluate a *billion* gates per second over a 10Gbps channel.

It is fair to say that the progress in the field of applied MPC is truly astounding, and things that would have been considered impossible a few years ago, and now considered routine.

Despite this progress, and many emerging real world uses, MPC is not yet widespread in practice. There remain several challenges to overcome before MPC can be deployed for a wide range of privacy-preserving applications. We discuss a few, and possible approaches for overcoming them next.

Cost. Despite dramatic advances in the 2PC and MPC technology in the past decade, secure function evaluation still may incur several orders of magnitude cost penalty over standard (non-private) execution, especially when protecting against malicious players. The exact overhead varies greatly from virtually non-existent to unacceptable, and mostly depends on the computed function.

In particular, for generic MPC, the protocols described in this book (which are, arguably, the ones that are most scalable today in typical settings) all require bandwidth that scales linearly in the size of the circuit. Bandwidth within a data center is inexpensive (indeed, many cloud providers do not charge customers anything for bandwidth between nodes within the same data center), but it requires a strong trust model to assume all participants in an MPC would be willing to outsource their computation to the same cloud provider. In some use cases, this linear bandwidth cost may be prohibitively expensive. Making bandwidth cost sublinear in circuit size requires a very different paradigm. Although it has been shown to be possible with threshold-based FHE schemes (Asharov *et al.*, 2012), such schemes are a long way from being practical. Recent results have shown that function-sharing schemes can be used to build lower-bandwidth MPC protocols for certain classes of functions (Boyle *et al.*, 2016a; Boyle *et al.*, 2018).

The solution to this bandwidth cost seems to require hybrid protocols that combining MPC with custom protocols or homomorphic encryption to enable secure computation without linear bandwidth cost. We have covered several approaches that incorporate these strategies in MPC including private set

intersection (Section 3.8.1), RAM-MPC (Chapter 5) and ABY (Section 4.3), but the limits of this approach are not known. Custom protocols can offer much lower costs, but developing custom protocols is tedious and only cost-effective for the most important and performance-critical operations. Future work may find more principled and automated was to combine generic MPC with homomorphic encryption and custom protocols to enable very efficient solutions without compromising security properties.

Another direction for dramatically reducing the cost of MPC is to employ secure hardware such as Intel's SGX. This assumes a somewhat awkward (but often realistic in practice) threat model where a vendor is trusted to implement a secure enclave and manage its keys, but not trusted fully to be an trusted third party. Signal recently released a private contact discovery service using SGX (Marlinspike, 2017), and many researchers are pursuing MPC designs that take advantage of SGX (Bahmani *et al.*, 2017; Shaon *et al.*, 2017; Priebe *et al.*, 2018; Mishra *et al.*, 2018).

Leakage trade-offs. Clearly, the MPC performance race will continue, although specific improvement areas may shift. For example, we have already achieved several MPC milestones, such as being able to fully utilize a very capable 10Gbps communication channel. Further, there exist several barriers for algorithmic performance improvement, such as the need for two ciphertexts for AND gate encryption, shown by Zahur *et al.* (2015). This may limit expected performance gains in basic generic Yao-based MPC.

Indicative of the recognition of these barriers and the data processing demands, a line of work emerged addressing the trade-off between MPC efficiency and achieved security guarantees, which we partially covered in Chapter 7. Another possible direction for leakage trade-offs is within the core protocols, where large efficiency gains may be achieved by giving up the strong security guarantees targeted by current protocols for more flexible information disclosure limits.

In terms of which MPC operation is in most need of improvement, achieving secure access to random memory locations stands out. Despite all the improvements described in Chapter 5, today's ORAM protocols are still concretely inefficient and are the main limit in scaling MPC to large data applications. An alternative to ORAM, in the absence of a satisfactory

fully secure solution, could be security-aware leakage of data access patterns, accompanied by a formal analysis of how this information satisfies security and privacy requirements. It is plausible that in many scenarios it can be formally shown, e.g., using tools from programming languages among others, that revealing some information about the access pattern is acceptable.

Output leakage. The goal of MPC is to protect the privacy of inputs and intermediate results, but at the end of the protocol the output of the function is revealed. A separate research field has developed around the complementary problem where there is no need to protect the input data, but the output must be controlled to limit what an adversary can infer about the private data from the output. The dominant model for controlling output leakage is *differential privacy* (Dwork and Roth, 2014) which adds privacy-preserving noise to outputs before they are revealed. A few works have explored combining MPC with differential privacy to provide end-to-end privacy for computations with distributed data (Pettai and Laud, 2015; Kairouz *et al.*, 2015; He *et al.*, 2017), but this area is in its infancy and many challenging problems need to be solved before the impacts of different types of leakage are well understood.

Meaningful trust. When cryptographers analyze protocols, they assume each party has a way to executed their part of the protocol that they fully trust. In reality, protocols are executed by complex software and hardware, incorporating thousands of components provided by different vendors and programmers. The malicious secure protocols we discussed in this book provide a user with strong guarantees that if their own system executes its role in the protocol correctly, no matter what the other participants do the security guarantees are established. But, in order to fully trust an MPC application, a user also needs to fully trust that the system executing the protocol on its behalf, which is given full access to their private data in cleartext, will execute the protocol correctly. This includes issues like ensuring the implementation has no side channels that might leak the user's data, as well as knowing that it performs all protocol steps correctly and does not attempt to leak the user's private input.

Providing full confidence in a computing system is a longstanding and challenging problem that applies to all secure uses of computing, but is

particularly sharp in considering the security of MPC protocols. In many deployments, all participants end up running the same software, provided by a single trusted party, and without any auditing. In practice, this provides better security that just giving all the data to the software provider; but in theory, it is little different from just handing over all the plaintext data to the software (or hardware) provider.

One appealing aspect of MPC techniques is they can be used to alleviate the need to place any trust in a computing system—if private data is split into shares and computed on using two systems using MPC, the user no longer needs to worry about implementation bugs in either systems exposing the data since the MPC protocol ensures that, regardless of the type of bug in the system even including hardware side channels, there is no risk of data exposure since the private data is always protected by cryptography withing the MPC protocol and is not visible to the system at all.

Finding ways to meaningfully convey to end users how their data will be exposed, and what they are trusting in providing it, is a major challenge for future computing systems. Making progress on this is essential for enabling privacy-enhancing technologies like MPC to provide meaningful and understandable benefits to the vast majority of computing users, rather than just to sophisticated organizations with teams of cryptographers and program analysts.

We wish to conclude this book on an optimistic note. The awareness that personal data can be compromised in a data breach, or can be abused by companies whose interests do not align with those of their users, is increasing. New regulations, including the European Union's *General Data Protection Regulation*, are making holding personal data a liability risk for companies. MPC has emerged as a powerful and versatile primitive that can provide organizations and individuals with a range of options for designing privacy-preserving applications. Many challenges remain, but MPC (and secure computation broadly) is a young and vibrant field, with much opportunity to create, develop and apply.