Cryptographically Secure Computation

Emily Shen, Mayank Varia, and Robert K. Cunningham, MIT Lincoln Laboratory

W. Konrad Vesey, Elkridge Security

Researchers are making secure multiparty computation—a cryptographic technique that enables information sharing and analysis while keeping sensitive inputs secret—faster and easier to use for application software developers.

Big data analytics creates a tension between data sharing and data confidentiality that is best allayed through concerted action. For instance, coalitions have formed between companies and between private industry and the federal government in several economic sectors including finance,[1] retail,[2,3] and aviation[4] to share information about cyber threats in order to predict and thwart future cyberattacks. An executive order from President Obama in February 2015 encourages the voluntary formation of such coalitions, stating that "organizations engaged in the sharing of information related to cybersecurity risks and incidents play an invaluable role in the collective cybersecurity of the United States."[5]

While everyone benefits from a more comprehensive understanding of cyber threats, information sharing raises privacy and security concerns. As President Obama noted in his executive order, "information sharing must be conducted in a manner that protects the privacy and civil liberties of individuals, that preserves business confidentiality, that safeguards the information being shared, and that protects the ability of the Government to detect, investigate, prevent, and respond to cyber threats."

For problems like these, we would ideally like technology that lets people learn the result of a joint computation without needing to reveal their own inputs to the computation. Such technology would not only provide strong security for existing data-sharing applications but also promote greater data sharing by enabling new applications that aren't currently possible because entities won't or can't share raw data—for example, for legal reasons.

SECURE MULTIPARTY COMPUTATION

Cryptographers have been developing this technology, known as secure multiparty computation (MPC), for the past three decades. Secure MPC guarantees that everyone learns the correct output of a joint computation but nothing else about anyone else's inputs, even when some of the people performing the computation might be actively or passively malicious.

Secure MPC can be done for arbitrary computations and for any number of parties. Hence, we can view secure MPC protocols as compilers that take as input a specification of a function and output a protocol that computes the function securely.

To see how this works, consider a technique called *secret sharing*. An m-of-n secret-sharing scheme splits a secret input into n pieces, or shares, that are held by different people in such a way that m people can combine their shares to reconstruct the secret, but any group of fewer than m people can't learn anything about the secret. For example, we can create a two-of-three secret-sharing

0018-9162/15/$31.00 © 2015 IEEE

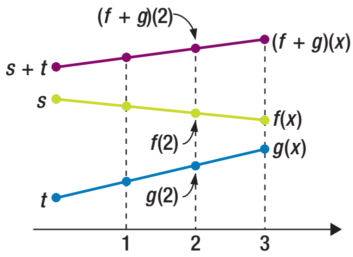


Figure 1. Graphical illustration of the addition of shares in a secret-sharing scheme.

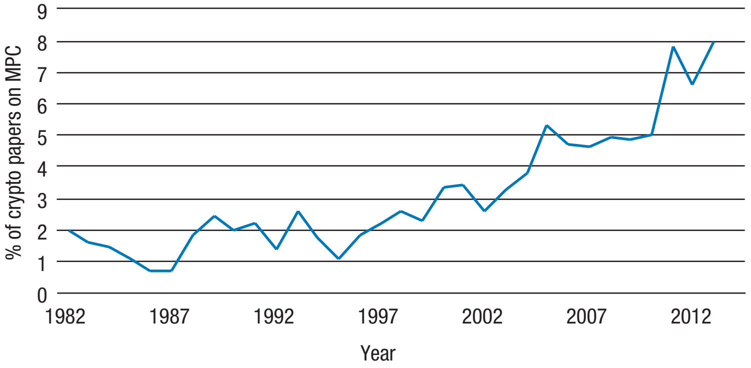


Figure 2. Secure multiparty computation (MPC) research by year as a fraction of overall cryptography research, based on Google Scholar data.

scheme using lines in two-dimensional space. To share a secret *s* among three people, we choose a random line *f* whose *y*-intercept equals *s*. Each person's share of the secret is a distinct point on the line. Since any two points uniquely define a line, any two people together can compute the secret; conversely, a single share reveals nothing about the secret. This technique can be generalized to any threshold *m* using polynomials of degree $m - 1$.

If two secrets *s* and *t* have been shared, then people can compute shares of the sum $s + t$ by simply adding their own shares together; this is shown graphically in Figure 1. They can also compute shares of the product $s \times t$ through a more complicated manipulation of the shares of *s* and *t*.

Since addition and multiplication form a logically complete set of gates, secret sharing makes it possible to perform any joint computation securely: each participant in the scheme secret-shares his or her own input, everyone jointly computes the desired function of all inputs, and then everyone can jointly reconstruct the final result.

EVOLUTION OF SECURE MPC

The field of secure MPC began in the early 1980s with the invention of the first secure two-party protocol[6] and multiparty protocols.[7,8] Since then, it has been an active area of cryptography research, as Figure 2 shows. Most secure MPC research to date has proceeded along three fronts: designing general-purpose protocols, optimizing protocols for particular functions of interest, and implementing secure MPC in software.

First, researchers have explored and expanded the space of parameters that influence the design of secure MPC protocols: adversary type, number of malicious parties, computational and network resources, security and functionality requirements, and so on. Figure 3 illustrates some of these parameters. Researchers have determined which parameter combinations are possible to achieve in a generic, function-agnostic manner with perfect security within some specified security model. Also, they've designed secure MPC protocols that cover much of the feasible space.

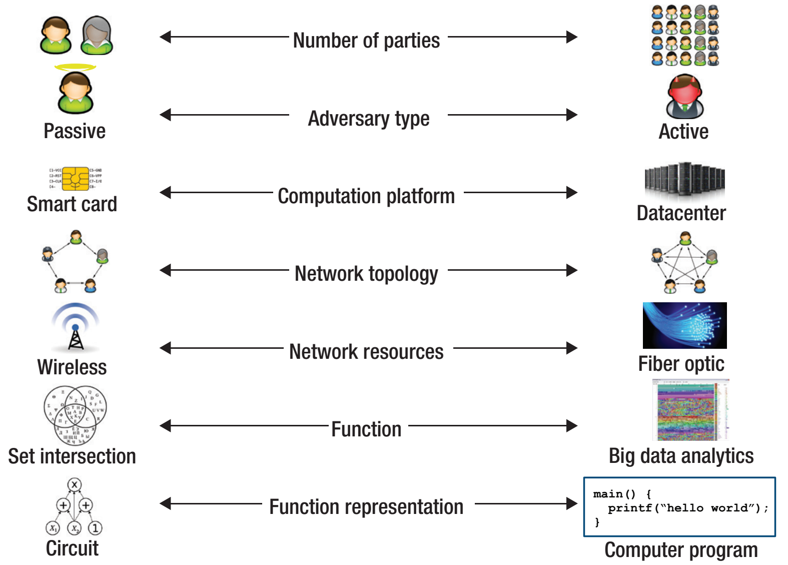
Second, researchers have tailored secure MPC protocols to specific applications and threat models. Specialized protocols can be much faster than generic ones for two reasons: they need only work for a restricted set of functions, and they can tolerate security imperfections that are acceptable in the context of the specific application. Specialized protocols have been developed for applications such as set intersection, substring search, and secure database queries.

Third, cryptographers have implemented frameworks that enable developers to produce secure code in a high-level language like Java using libraries that implement secret sharing and other secure MPC techniques. As most of these frameworks don't abstract away the details of secure MPC, developers must understand how secure MPC works and change the way they write code accordingly.

The Security and Privacy Assurance Research (SPAR) project run by the Intelligence Advanced Research Projects Activity (IARPA) is a recent example of a project that leveraged research in all three of these areas. SPAR developed specialized software for privacy-preserving SQL databases and publish-subscribe systems. Testing and evaluation by MIT Lincoln Laboratory showed that SPAR software often performed within a 3× overhead of a non-privacy-preserving database like MySQL while providing strong security guarantees.[9]

REMAINING CHALLENGES

While cryptographers have created a plethora of secure MPC protocols, optimizing a protocol for a specific application remains difficult.



```
main() {
  printf("hello world");
}
```

Figure 3. Parameters that influence the design of secure MPC protocols.

Imagine a software engineer working with a cryptographer to design and develop secure software. Their interaction proceeds as shown in Figure 4. Initially, the software engineer explains her functionality, security, and performance requirements. Then, the cryptographer surveys existing secure MPC protocols, chooses components and techniques appropriate to the requirements, and designs a protocol from these. Next, the cryptographer explains the security guarantees provided by the protocol. The software engineer and cryptographer iteratively refine the requirements and the protocol design to strike a desirable balance between security and performance for the given application. Finally, the cryptographer explains how the customized protocol works and how it must interact with other software to guarantee security, and the software engineer implements the protocol.

Currently, this process is long and highly sensitive to changes in application requirements. For instance, the SPAR program took three years to create usable, efficient secure database search systems, and it would likely take a comparable amount of time and effort to extend the requirements of the program—for example, add support for additional query types—or port its ideas to another environment such as a NoSQL database.

CURRENT EFFORTS

To contemplate the future of secure MPC technology, IARPA and MIT Lincoln Laboratory held the Security and Privacy Assurance Research–Multiparty Computation (SPAR-MPC) Workshop in May 2014 (https://events.ll.mit.edu/spar-mpc-workshop). More than 40 experts attended the workshop, presented their visions for secure MPC, and described several research avenues to make secure MPC more amenable to use in real applications.

Attendees discussed various ways in which researchers are improving the design and implementation of secure MPC protocols. First, cryptographers are developing protocols for more of the parameter space. Particularly important are protocols that reflect aspects of modern computing environments including multicore systems, fast RAM, and communication locality in large-scale environments like the cloud. Second, the research community is implementing, profiling, and optimizing secure MPC protocols and building blocks, and using lessons learned from these efforts to better understand performance bottlenecks.

Additionally, workshop attendees described current efforts to begin addressing the challenges of the secure computation development loop shown in Figure 4. In particular, a few frameworks now available for secure MPC software development, including Wysteria[10] and Sharemind,[11] require little or no knowledge of cryptography. These enable a developer to provide a functional description of the desired computation in a high-level, special-purpose language, along with annotations stating which data is permitted to be revealed. The compiler then transforms this high-level description into a low-level representation (such as a circuit) that is fed to the underlying secure MPC protocol. These programming tools separate the tasks of software engineers and cryptographers, simplify their interaction, improve usability, and reduce development time.

LONG-TERM VISION

We envision a future with secure MPC compilers containing a "virtual cryptographer" that handles security behind the scenes—that is, the compiler will automate the tasks performed by the cryptographer shown in Figure 4. Based on discussions at the workshop, we believe this will require advances both within cryptography and at the interface between cryptographers and software engineers.

Within cryptography, we believe future researchers will understand the MPC security/performance tradeoff space rigorously and comprehensively. This will have two benefits. First, it will enable compilers to compare cryptographic protocols automatically and systematically to determine the best ones for a particular use case. Second, it will lead to a mechanism that composes cryptographic building blocks automatically while respecting specified security constraints. These two benefits will improve existing cryptographic protocols' adaptability and interoperability.

At the interface between cryptographers and software engineers, our long-term vision is for secure MPC compilers to provide tunable, adaptable tradeoffs between performance and security so that software engineers can make educated choices about their security needs. An important aspect of this work will be automating the process of understanding the semantic meaning and implications of a cryptographic protocol's security properties. A compiler with this ability will be able to explain security tradeoffs automatically to the engineer without the need for a cryptographer in the development loop. For example, in SPAR's database setting, a future compiler might state, "I can hide when a client makes two queries that share a common record, at the expense of a 10× performance slowdown."

While it will ultimately be up to the engineer to determine the optimal balance between security and performance for a given application, we believe secure MPC tools of the future will enable rapid software development that allows the engineer to focus on the "what" rather than the "how" of security.

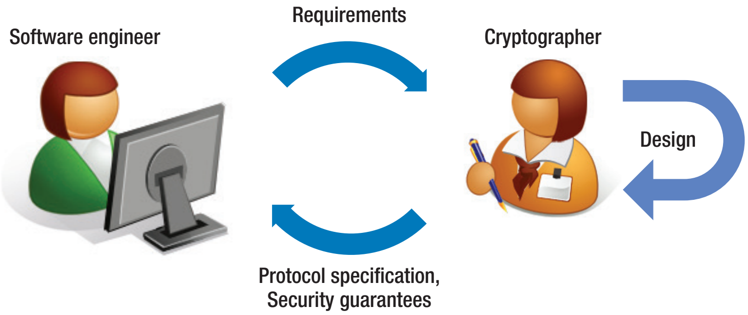


Figure 4. Interaction between a software engineer and cryptographer to develop a customized secure application.

ACKNOWLEDGMENTS

MIT Lincoln Laboratory's work in support of the SPAR program and the SPAR-MPC workshop was sponsored by the Intelligence Advanced Research Projects Activity, and the Lincoln Laboratory portion of this work was sponsored by the Department of the Air Force, under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

REFERENCES

1. H. Kuchler, "US Financial Industry Launches Platform to Thwart Cyber Attacks," *Financial Times*, 24 Sept. 2014; www.ft.com/intl/cms/s/0/080092b2-437a-11e4-8a43-00144feabdc0.html#axzz3TRbYCj1J.
2. Nat'l Retail Federation, "National Retail Federation Announces Information-Sharing Platform," press release, 14 Apr. 2014; https://nrf.com/media/press-releases/national-retail-federation-announces-information-sharing-platform.
3. Retail Industry Leaders Assoc., "Retailers Launch Comprehensive Cyber Intelligence Sharing Center," press release, 14 May 2014; www.rila.org/news/topnews/Pages/RetailersLaunchComprehensiveCyberIntelligenceSharingCenter.aspx.
4. R. King, "Aviation Industry and Government to Share Cyber Threats in New Intelligence Center," *The Wall Street J.*, 15 Apr 2014; http://blogs.wsj.com/cio/2014/04/15/aviation-industry-and-government-to-share-cyberthreats-in-new-intelligence-center.
5. Executive Office of the President, "Promoting Private Sector Cybersecurity Information Sharing," Executive Order 13691 of 13 Feb. 2015, 80 FR 9347, *Federal Register*, 20 Feb. 2015; https://federalregister.gov/a/2015-03714.
6. A.C. Yao, "Protocols for Secure Computations," *Proc. 23rd Ann. Symp. Foundations of Computer Science* (FOCS 82), 1982, pp. 160–164.
7. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computation," *Proc. 20th Ann. ACM Symp. Theory of Computing* (STOC 88), 1988, pp. 1–10.
8. O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game, or a Completeness Theorem for Protocols with Honest Majority," *Proc. 19th Ann. ACM Symp. Theory of Computing* (STOC 87), 1987, pp. 218–229.
9. M. Varia et al., "Automated Assessment of Secure Search Systems," *ACM SIGOPS Operating Systems Rev.*, vol. 49, no. 1, 2015, pp. 22–30.
10. A. Rastogi, M.A. Hammer, and M. Hicks, "WYSTERIA: A Programming Language for Generic, Mixed-Mode Multiparty Computations," *Proc. IEEE Symp. Security and Privacy* (SP 14), 2014, pp. 655–670.
11. D. Bogdanov, "Sharemind: Programmable Secure Computations with Practical Applications," PhD dissertation, Institute of Computer Science, Univ. of Tartu, 2013.

EMILY SHEN is a technical staff member in the Secure Resilient Systems and Technology group at MIT Lincoln Laboratory. Contact her at emily.shen@ll.mit.edu.

MAYANK VARIA is a technical staff member in the Secure Resilient Systems and Technology group at MIT Lincoln Laboratory. Contact him at mayank.varia@ll.mit.edu.

ROBERT K. CUNNINGHAM is the leader of the Secure Resilient Systems and Technology group at MIT Lincoln Laboratory. Contact him at rkc@ll.mit.edu.

W. KONRAD VESEY is an independent consultant with Elkridge Security (www.elkridgesecurity.com) and has more than 25 years of experience in the cybersecurity field. Contact him at konrad@elkridgesecurity.com.