Research Statement

Wenting Zheng

The recent revolution in advanced data analytics gave rise to a growing demand among organizations for high quality data. However, in many domains such as finance and medicine, organizations have encountered obstacles in data acquisition because their target applications need sensitive data that reside across multiple parties. For example, many banks want to train machine learning models for detecting fraud, but any single bank does not possess a holistic enough dataset to compute an accurate model since fraudulent transactions happen across banks. One promising solution to this data scarcity problem is *collaborative computation*, where several organizations pool together their data and compute on the joint dataset. This type of computation enables parties to acquire *a larger volume* of data, as well as *more diverse* data. Unfortunately, organizations are often unwilling or unable to share their data in plaintext due to business competition or government regulation. My graduate work aims to solve this problem by building systems that enable secure collaborative computation. My systems allow multiple organizations to run complex computations on their joint dataset *without revealing their sensitive data to each other*.

My approach. My research approach starts with problem selection. I take inspiration from the privacy and security challenges that people face today, as well as the potential for enabling better communication and collaboration across trust domains. For example, my research thesis was motivated by the real problems that my industry collaborators were facing due to their inability to share sensitive data with other organizations.

Second, I solve these research problems by building practical and secure systems via a *co-design of systems and cryptography*. Though the field of cryptography has made amazing progress over the past decades, few of these theoretical advances have been incorporated into practical systems. I bridge this gap between theory and practice by taking a systems-oriented approach that is backed by a deep understanding of cryptography. Throughout my work, I not only utilize a wide range of tools from both systems (distributed systems, hardware enclaves, query planning, consistency, compilers) and cryptography (oblivious computation, maliciously secure multi-party computation, zero-knowledge proofs), but also innovate them to make practical solutions possible. My systems provide strong and provable security guarantees and are often orders of magnitude faster compared to prior work or the more straightforward ways of integrating cryptography into systems.

Finally, as a system security researcher, one of my goals is to design, build, and open source system artifacts that provide strong security guarantees. I have open sourced my research work and organizations like IBM Research, Ericsson, Alibaba, and Microsoft have either used it internally, in production or have contributed to it.

Overview. My graduate research [13, 15, 16, 7, 14, 10, 8, 9, 17] focuses on enabling secure collaborative computation by preventing any party from seeing any other party's input data or seeing the intermediate outputs of a computation. For my thesis, I chose to secure two important workloads: analytics and machine learning. My system designs are rooted in strong cryptographic foundations, and are built out of both hardware and software mechanisms. For example, in Opaque [13], I approached the secure analytics problem by utilizing trusted hardware enclaves [5], which can protect sensitive data and securely run arbitrary computation on this data in the presence of a malicious operating system. At the same time, they are also prone to an important attack vector called access pattern leakage. Using novel insights from both cryptography and systems, Opaque is able to protect against this leakage and achieves 2300× performance gain compared to a state-of-the-art oblivious system. On the other hand, since hardware enclaves are relatively new, their availability as well as the underlying security assumptions may not fit many organizations. Therefore, I also tackled the machine learning workloads using a software-based only approach that utilizes secure multi-party computation (MPC) [1, 4, 12]. In Helen [16], I designed and implemented a tailored secure cryptographic protocol for training regularized linear models among multiple parties. Helen is the first specialized training protocol under a setting where an adversary can not only compromise a majority of the parties but also deviate from the protocol. Compared to implementing linear model training in a state-of-the-art generic framework with the same security guarantees, Helen is able to reduce the training time by up to four orders of magnitude.

Thesis work

Secure computation for analytics

Opaque: analytics processing. Many applications store and analyze data using databases, and SQL is a popular interface for a wide range of applications. Recently, many organizations have begun to run SQL analytics over their sensitive data on the cloud. However, cloud computing's efficiency and elasticity come at the cost of privacy because organizations' sensitive data are stored in plaintext at these third-party providers. Hardware enclaves [5] have been proposed as a possible solution as they can encrypt sensitive data and securely run arbitrary programs over the encrypted data on a machine with a malicious operating system. Unfortunately, enclaves are also prone to an important attack vector in which an adversary infers sensitive information by observing memory and network access patterns of encrypted data [11]. To protect against such attacks, I designed and implemented Opaque [13], a hardware enclave-based secure SQL analytics platform that also protects against memory and network access pattern leakage. This is a challenging research problem because naïvely applying cryptographic techniques can introduce extremely high overheads. To address this challenge, we proposed a two-part solution. First, we used cryptography to design new distributed *oblivious* SQL operators that hide data access patterns. Second, we also developed novel query planning techniques to further optimize the performance. Compared to prior state-of-the-art oblivious systems, Opaque achieves up to three orders of magnitude performance improvement. Though originally designed for the single client scenario, multiple organizations can also use Opaque for collaborative analytics by pooling their data together on the cloud (under the same key or under different keys) and run analytics on everyone's joint dataset. The Opaque code is open source and has garnered interest from several industry partners. IBM Research deployed Opaque, and Ericsson and Alibaba used Opaque for internal use cases. Microsoft is also contributing to our open source effort. and Microsoft's Azure Confidential Computing and Scotiabank have a contract to deploy anti-money laundering on top of Opaque and a secure learning project from the RISELab.

MiniCrypt: secure key-value storage. In addition to analytics query execution, applications also use databases for storage. However, one of the main drawbacks of encrypted databases and storage engines is their memory inefficiency because encrypted data is not compressible. My project MiniCrypt [15] investigates how to compress and encrypt cloud-outsourced distributed key-value stores. While the idea of compressing data before encryption is very intuitive, the challenge is that encrypting multiple records together with the client's key removes the server's ability to manage key-value pairs and maintain consistency semantics. Therefore, we develop a new protocol to handle reads and updates of *packed records* while maintaining the key-value store's server-side consistency semantics. By utilizing compression to reduce the key-value store's memory footprint, our evaluation shows that MiniCrypt can increase the server's throughput by up to two orders of magnitude by fitting more data in main memory.

Secure collaborative machine learning

Helen: maliciously secure training for linear models. Though enclaves are powerful tools for secure computation, solutions that are hardware agnostic are also essential for organizations that do not possess such hardware or do not want to rely on hardware assumptions. From the cryptographic perspective, the collaborative computation setting fits within a well known framework called *secure multi-party computation* (MPC). In this framework, parties are able to collaboratively compute a function on everyone's input data without revealing their input data to the other participants; in fact, the parties only learn the final output. Many organizations (such as banks) who want to use MPC are concerned about the behavior of the other participants, and hence want to only trust themselves. However, most of the prior work in secure collaborative training have focused on the *semi-honest* threat model, which assumes that the adversary never deviates from the protocol. A more realistic threat model is protection against a *malicious adversary* who can arbitrarily deviate from the protocol. While there exist generic maliciously secure MPC frameworks, implementing a computation in a straightforward manner using these frameworks can result in a very inefficient protocol. In Helen [16], I designed and built the first specialized maliciously secure MPC protocol for training a regularized linear model. By combining insights from systems, cryptography, and machine learning, we show that it is possible to reformulate the training process so that the expensive cryptographic computation scales independently of the number of training samples. With our insights, Helen is able to achieve up to four orders of magnitude of performance improvement when compared to a common training protocol implemented using an existing

state-of-the-art generic maliciously MPC framework. For example, Helen is able to train a LASSO model (with the same accuracy) in less than 3 hours instead of an estimated 3 months.

Delphi: secure neural network inference. Inference is also an important workload to secure because many companies provide neural network prediction services. However, current prediction systems compromise one party's privacy: either the user has to send sensitive inputs to the service provider for classification, or the service provider must store its proprietary neural networks on the user's device. The former harms the personal privacy of the user, while the latter reveals the service provider's proprietary model. I worked with other collaborators from Berkeley on Delphi [7], a secure prediction system that allows two parties to execute neural network inference without revealing either party's data. Delphi simultaneously co-designs cryptography and machine learning. Compared to prior work, our new hybrid cryptographic protocol improves upon the communication and computation costs by pushing expensive cryptography to a preprocessing phase. We also develop a planner that automatically generates neural network architecture configurations that navigate the performance-accuracy trade-offs of our hybrid protocol. These techniques allow us to achieve a 22× improvement in online prediction latency compared to the state-of-the-art prior work.

Cerebro: end-to-end platform for collaborative training and inference. My overarching goal for collaborative machine learning is to build a system that is usable by people with no expertise in cryptography to execute a learning task securely. While specialized systems like Helen or Delphi are very efficient, a practical system must address two important obstacles to meet the needs of its users. First, users often want the ability to customize their learning tasks, which means that the collaborative learning platform needs to expose a programming interface for users to write arbitrary programs. Moreover, since the users are not experts in cryptography, the platform should be able to automatically optimize and plan the secure execution of their programs. Second, such a platform also needs to take into account the complex economic relationships of the participants who are potentially competing with each other. Cerebro[14] is a platform that addresses these two challenges. First, Cerebro provides a cryptographic compiler that is able to automatically compile and optimize a program written in a high-level language into an MPC protocol. Second, Cerebro also introduces runtime compute policies (which control if/how the result is released) and post-runtime cryptographic auditing (which allows a third party to audit the parties' input using MPC). Together, these mechanisms allow organizations to ensure that their incentives and constraints are met before the result of a learning task is released, and also enables participants to identify the source of malicious input data. By taking an end-to-end approach to the system design, Cerebro allows multiple parties with complex economic relationships to safely run collaborative machine learning.

DIZK: distributed zero-knowledge proof generation. Zero-knowledge proof is a cryptographic primitive that is often used in secure collaborative computation to ensure computation integrity. In DIZK [10], my co-authors and I tackle the problem of scalable proof generation for a special type of zero-knowledge proofs called zkSNARKs [2, 6, 3]. The cryptography behind zkSNARKs allows a party to prove correctness of any computation that can be expressed as *logical circuits*, and the generated proof only consists of a constant number of group elements. However, the trade off is that the prover needs to do heavy computation in order to generate these short proofs. The proof generation uses all intermediate outputs of a computation, thereby consuming a great amount of memory and compute power. Prior zkSNARK systems could only execute on a single machine, and only supported circuits up to $10-20$ million gates. DIZK distributes the zkSNARK prover to multiple machines and is able to compute circuits that are 100× larger than prior systems.

Other work

SCL [8] is a project that rethinks the meaning of controller consistency in software-defined networks (SDN). Prior SDN systems utilize Paxos/Raft or even stronger mechanisms to maintain consistent controllers. In reality, operators and users care that certain properties or invariants are obeyed by the forwarding state installed in switches, and are not directly concerned about control plane consistency. With this in mind, we redesigned controllers with a mechanism called SCL (Simple Coordination Layer) that achieves *eventual consistency*. This design allows for a simple and efficient control plane that guarantees high availability and still enables well-behaved decisions under normal failure models.

During my master's at MIT, I worked on Silo [9], a fast multicore database that can execute millions to tens of millions of transactions per second. However, the original Silo database was not robust to failures. My master's thesis work [17] brought persistence to Silo by adding logging, checkpointing, and recovery. The design of the recovery system is interesting in that every part of the persistent process utilizes concurrency. Such parallelism allows us to log and checkpoint with a very small reduction in runtime performance, as well as a replay speed that closely matches the maximum speed of disk I/O.

Future work

I aim to work towards a world where the capabilities of cryptography are made accessible to everyone. By incorporating the latest theoretical advances into practical systems, I hope that people can better collaborate and communicate across trust domains with few compromises in data privacy and security. To enable this vision, I intend to build secure systems and tools that bridge the gap between theory and practice. I want to develop systems to enable users who have no expertise in cryptography to efficiently address their applications' needs and constraints while enjoying the benefits of provable and verifiable security. At the same time, I also want to build tools that enable cryptographers to instantiate their novel theoretical work into real implementations. My graduate work has laid out some initial steps towards achieving these goals, and I am excited to use my skills to continue solving more interesting problems in this area.

Better systems tools for MPC. MPC research over the past decades has seen dramatic improvement in asymptotic performance from a theoretic perspective. However, there is a rich set of systems tools that can help make MPC more efficient and more usable. One existing problem is that many MPC implementations assume that the protocol only operates on a single machine at each party. Therefore, an interesting direction of research is to look into how to parallelize and distribute various MPC protocols within each trust domain to take advantage of their local cluster compute resources. Another problem is that there exist many generic MPC protocols, but no one protocol is best suited to all applications. I want to build an extensible platform that can automatically utilize new MPC backends. Cryptographers should be able to easily integrate their new MPC frameworks into this platform, and it should automatically optimize and plan high-level programs using the new backends.

Theoretical guarantees for incentivizing collaboration. Most prior work in secure collaborative computation assumes that the parties are ready to collaborate, but it is still unclear what actions need to be taken to incentivize these parties to collaborate in the first place. I want to use mechanism design to define economic policies and provide theoretical guarantees for these policies. Combining game theory with MPC also leads to an interesting threat model: what if we assume that the participants are all rational actors with utility functions, instead of a setting where an external adversary can compromise a subset of the parties?

Automated synthesis and verification of secure protocols. Cerebro shows some initial opportunities for automated compilation and optimization of MPC. Using this as a starting point, I want to explore how to automatically generate an efficient secure computation program such that its performance is close to a hand-tuned protocol. I want to focus on two specific kinds of secure protocols: oblivious computation and MPC. I think there are very interesting opportunities for applying program synthesis techniques for optimization by exploring random transformations of the program while ensuring that the transformations are still provably secure. Moreover, many cryptographic systems claim provable security, but that aspect only comes from the system design. Real world implementations of cryptography often have bugs. I want to explore program verification techniques that can help generate provable secure implementations of protocols. I also hope to extend verification to the design level and build tools that aid theoreticians in creating and checking cryptographic proofs for their protocols.

References

[1] Ben-Or, M., Goldwasser, S., and Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing* (1988), ACM, pp. 1–10.

[2] Bitansky, N., Canetti, R., Chiesa, A., and Tromer, E. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (2012), ACM, pp. 326–349.

[3] Gentry, C., and Wichs, D. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing* (2011), ACM, pp. 99–108.

[4] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (1987), ACM, pp. 218–229.

[5] McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., and Savagaonkar, U. R. Innovative instructions and software model for isolated execution. *HASP@ ISCA 10* (2013).

[6] Micali, S. Computationally sound proofs. *SIAM Journal on Computing 30*, 4 (2000), 1253–1298.

[7] Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., and Popa, R. A. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium (USENIX Security 20)* (2020).

[8] Panda, A., Zheng, W., Hu, X., Krishnamurthy, A., and Shenker, S. SCL: Simplifying distributed sdn control planes. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (2017), pp. 329–345.

[9] Tu, S., Zheng, W., Kohler, E., Liskov, B., and Madden, S. Speedy transactions in multicore in-memory databases. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (2013), ACM, pp. 18–32.

[10] Wu, H., Zheng, W., Chiesa, A., Popa, R. A., and Stoica, I. DIZK: A distributed zero knowledge proof system. In *27th USENIX Security Symposium (USENIX Security 18)* (2018), pp. 675–692.

[11] Xu, Y., Cui, W., and Peinado, M. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Security and Privacy (SP), 2015 IEEE Symposium on* (2015), IEEE, pp. 640–656.

[12] Yao, A. C. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on* (1982), IEEE, pp. 160–164.

[13] Zheng, W., Dave, A., Beekman, J. G., Popa, R. A., Gonzalez, J. E., and Stoica, I. Opaque: An oblivious and encrypted distributed analytics platform. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (2017), pp. 283–298.

[14] Zheng, W., Deng, R., Chen, W., Popa, R. A., Panda, A., and Stoica, I. Cerebro: A platform for multi-party cryptographic collaborative learning. *Under submission*.

[15] Zheng, W., Li, F., Popa, R. A., Stoica, I., and Agarwal, R. MiniCrypt: Reconciling encryption and compression for big data stores. In *Proceedings of the 12th European Conference on Computer Systems* (2017), ACM, pp. 191–204.

[16] Zheng, W., Popa, R. A., Gonzalez, J., and Stoica, I. Helen: Maliciously secure coopetitive learning for linear models. In *S&P'19* (2019).

[17] Zheng, W., Tu, S., Kohler, E., and Liskov, B. Fast databases with fast durability and recovery through multicore parallelism. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)* (2014), pp. 465–477.