OVERVIEW. My research area is cryptography, with a practice-oriented theme. We use cryptography everyday when we browse the web, swipe credit cards, or chat with friends over the phone. But designing practical cryptographic systems is hard, because we have to guard against future unknown attacks, there are often legacy constraints that prevent direct solutions, and we have to meet ever-increasing needs of performance-hungry applications. My research aims to address real-world problems of broad interests via deep theory, bringing ideas from probability theory, algorithms, programming languages, and high-performance computing. This involves, for example, (1) developing probabilistic techniques to capture the exact quantitative security of real-world schemes, (2) using graph-theoretic mechanisms to break encryption standards, (3) leveraging programming-language methods to automate the design of cryptographic constructions, and (4) building performance models to optimize the encryption overhead for MPI communication on high-speed clusters. Recurring themes in my work include questioning and refining threat models to resolve the tension between performance demands and provable-security goals, providing abstraction boundaries for ease of correct use of cryptographic tools, and building systematic frameworks to analyze a variety of problems.

My research group has achieved very high visibility at top-tier conferences in cryptography and computer security, and our output has been comparable to that of established cryptography groups at high-ranked institutions, as indicated, for example, by csrankings.org.

RESEARCH IMPACT. The results of my research have yielded tangible impacts on cryptographic practices. Below are a few examples.

- Ideas in my work of streaming encryption are adopted in Google’s Tink library. Tink is heavily used by many companies such as Google, Square, and Citadel, as well as hundreds of Google Cloud customers and Google Pay partners.

- RFC 8452 relies on my analyses to justify the security of the encryption scheme AES-GCM-SIV that is used for ticket encryption in the QUIC protocol. My study of the GCM encryption is used (for example, by RFC 9001) to prescribe its usage limits in QUIC and TLS. You are using QUIC and TLS when you watch a YouTube video, open Gmail, or request an Uber drive.

- My Swap-or-Not shuffle is a major component in the validator-selection protocol of Ethereum 2.0. Ethereum is the second most popular cryptocurrency after Bitcoin. As of November 2022, its market capitalization is 192.48 billion dollars.

- My attacks on the NIST standards of Format-Preserving Encryption (that are widely used for encrypting credit-card numbers and legacy databases) led to a revision of the standards.

My doctoral thesis lays the practical foundations for garbled circuits (GCs), a central tool in secure distributed computing. It has significantly contributed to the explosive growth of many GC-based applications, provided a cornerstone for subsequent optimizations, and helped realizing bugs in several well-known implementations. As a result, the papers of my thesis have been cited totally for more than 1,000 times and their material has been used in cryptography classes at MIT, Stanford University, UC Berkeley, and many other places.

RECOGNITION. My work has received some recognition from the scientific community. I received the Best Paper Award at Usenix Security 2022 and CCS 2015, and the Best Paper Honorable Mention at EUROCRYPT 2015. My papers at CRYPTO 2013 and CRYPTO 2016 were ranked among the top five
papers in those proceedings and invited to the Journal of Cryptology. Oded Goldreich, a Knuth Prize winner, puts my CRYPTO 2013 paper in his shortlist of important theoretical work. I also won the 2021 NSF CAREER Award, an (unsolicited) gift from Google under their Patch Rewards Program, and the 2019 Faculty Research Award of the CS Department at FSU.

The remaining part of my statement highlights some selected corners of my research.

**Weeding out Heuristics in Applied Cryptography**

To deal with unknown future attacks, a common route is *provable security*: first formalize a threat model, and then design a scheme that provably achieves security in this model. Provable security is not perfect, as the model may not capture all possible attacks, the security proof must be based on a certain hardness assumption (such as factoring), and a bad implementation may completely void all theoretical guarantees. Still, this is usually the best practice.

Unfortunately, the cryptography used daily by billions of people is often designed without a provable-security goal in mind. In many cases, the security validation relies on the resistance to prior known attacks, but there is no theoretical guarantee to guard against future ones. In some other cases, provably-secure designs are too expensive for performance-hungry applications, and practitioners therefore have to resort to heuristic solutions. My research aims to eliminate heuristics in real-world cryptographic schemes by (i) broadening the theory of cryptography to understand the exact security of legacy schemes, and (ii) weeding out insecure constructions by giving attacks. Below, I will give some examples.

**Format-Preserving Encryption (FPE).** FPE is a form of encryption in which ciphertexts have the same format as the plaintexts. For example, you want to encrypt a 16-digit credit-card number so that the ciphertext is also another 16-digit credit-card number. FPE is motivated by the need for encrypting legacy databases, in which the ciphertexts replace the plaintexts, and thus must have the same format as the database schema dictates. Constructing good FPE schemes is challenging because for most applications of FPE, the size of the message space is a small number, and it is not a power of two. The most widely used FPE schemes are NIST standards FF1 and FF3, but some companies (such as Protegrity Corp.) use their own FPE methods and claim higher security than FF1/FF3.

In a series of papers (CCS 2016, CRYPTO 2018, EURORYPT 2019), my coauthors and I give a number of attacks on NIST standards FF1/FF3 and other industrial FPE methods. Our attacks prompted NIST to revise those standards, and our papers were used extensively in ANSI meetings for discussing how to patch the FPE methods. We also break Protegrity’s FPE scheme, forcing Protegirty to switch to FF1. The EUROCRYPT 2019 paper also contains neat, pedagogical ideas for cryptanalysis. Adi Shamir, a Turing Award laureate, handpicked it to teach in his cryptography class at the Weizmann Institute of Science.

**Random number generators (RNGs).** Cryptography ubiquitously relies on the assumption that high-quality randomness is available. Violating this assumption is not merely a theoretical glitch; it actually often leads to security disasters. A good RNG is therefore a crucial part of any cryptographic system. The most popular RNG is the NIST standard CTR-DRBG, but this scheme unfortunately has a troubled history. It is one of the recommended designs in NIST SP 800-90A, together with the infamous NSA-planted Dual EC, and this association breeds suspicion. Its structure is convoluted,

---

1. O. Goldreich. My choices. [https://www.wisdom.weizmann.ac.il/~oded/my-choice.html](https://www.wisdom.weizmann.ac.il/~oded/my-choice.html)
frustrating attempts to justify security. Even worse, some options in the overly flexible specification are known to be problematic, causing vulnerabilities in NIST-compliant implementations.

In my CRYPTO 2020 paper, my student Yaobin Shen and I give the first provable-security treatment for CTR-DRBG. Our paper reveals hidden design insights in the seemingly cumbersome construction. We show that its heuristic tweaks to pass the NIST entropy tests, which were dismissed by prior work, actually improve security. We also point out that some of the parameters in the specification of CTR-DRBG are overly optimistic, and recommend more conservative choices.

Computer-Aided Design for Cryptography

Designing cryptographic schemes is onerous and error-prone. Despite this painstaking process, the products may still have critical vulnerabilities, or the developers may miss opportunities to improve speed or achieve some desirable properties. My research investigates how to use programming-language techniques to automate the design of cryptographic constructions.

In my CCS 2015 paper, my coauthors and I study how to synthesize secure authenticated-encryption (AE) schemes. We build a system that browses through and analyzes the security of millions of possible designs in a given template. The tool returns thousands of provably secure AE modes—most of those are new—and generates explicit attacks on other schemes. We also add some searching mechanism to shortlist modes that are fully parallelizable, or those that use the blockcipher only in the forward direction (to save chip area in hardware realization). Among the new candidates, we find a scheme that is better than OCB, the fastest mode in the literature. Our work received the Best Paper Award at CCS 2015, and our code was incorporated into a software of Galois Inc.

When Standard Encryption Fails: Next-Generation Encryption

Symmetric encryption is the canonical primitive of cryptography, with which the field is often identified in the popular mind. Over time, the primitive has evolved to what is commonly known as authenticated encryption (AE). Standard AE schemes generally work well in practice; they are widely used in network protocols such as TLS, IPSec, and SSH. Recent attacks and applications however motivate another evolution. I am interested in establishing foundations for the next-generation encryption, using these theoretical guidelines to build practical schemes with minimal changes from existing standards, and collaborating with the industry and standard agencies for adoption.

Streaming AE. Standard AE schemes are ill-suited for the streaming setting where one needs to encrypt potentially huge messages as soon as they arrive. This kind of situation arises when Netflix sends its movie streams to users. This is what troubles memory-constrained devices when they have to deal with data that they cannot load entirely. This also happens when interactive applications, such as SSH, need to send immediately each character they receive from the keyboard.

In my CRYPTO 2015 paper, my coauthors and I provide a foundational treatment for streaming encryption, which is the basis for Google to develop an authenticated streaming encryption scheme in their cryptographic library Tink. The tool is widely used at Google, and a major application is to encrypt gigantic files. Still, this use demands security for random-access decryption, but the theory in our paper only offers guarantees for sequential one. In addition, the method has some usability issues; we occasionally receive emails from practitioners asking for how to tweak our mechanism to adapt to

\[2\] According to John Kelsey (personal communication), he only designed CTR-DRBG so that he could not break it, and he had to intertwine the components to pass the NIST entropy tests.
their situations. This also causes the Tink library to deviate from our original blueprint, but Google developers do not offer justification for these changes.

In my CCS 2020 paper, my student Yaobin Shen and I develop a more general framework for streaming encryption, use it to analyze the security of the scheme in Tink, and also suggest how to improve both of its security and efficiency. Our work received an (unsolicited) $10,000 gift via Google’s Patch Rewards Program.

**COMMITTING AE.** There is a common misunderstanding that an AE ciphertext is also a commitment of the key, meaning that decrypting it via another key should result in a rejection. Standard AE schemes however do not deliver this property. The lack of committing security seems harmless at first, but it has led to several recent attacks on the abuse-reporting mechanism of Facebook Messenger, password-based authenticated encryption, Google Subscriptions, and many other applications. There have been several attempts to address this issue, but there is an undesirable dichotomy in current fixes: either one has to add an overhead that is proportional to hashing the message (that makes encryption several times slower), or one has to transmit an additional 32 bytes per ciphertext. This performance drop may stem from a lack of a proper definitional treatment of committing security, as prior papers on committing AE focus on some specific applications, and define security notions only for these ends.

In my EUROCYPFT 2022 paper, my coauthor and I give a systematic definitional treatment of committing AE. We point out that no single definition can cover all applications. One actually needs two notions for committing AE: one for committing just the key, another for committing everything (the key, the message, and the contextualized information). We then provide a simple generic transform to turn a key-committing AE scheme to an all-committing one, and show that the added overhead is optimal. To achieve key-committing security efficiently, we identify misconceptions that led to overkill designs in prior work. We then show how to build a key-committing variant of GCM, a widely used AE standard, that requires only tiny changes on GCM code. Our scheme has nearly the same speed as GCM, and incurs no additional bandwidth cost. Even better, it fixes a fragility issue of GCM that has troubled practitioners for decades. We plan to propose our scheme to NIST as a new encryption standard.

**Practical Systems with Guides from Deep Theory**

Cryptographic protocols are complex and error-prone, and often do not make the best use of the underlying primitives (such as encryption or hash functions), leaving lots of room for improving efficiency. My research aims to provide good abstraction boundaries to reduce errors in protocols, identify and question implicit assumptions to improve security and efficiency, and use theory-guided insights to cut right to the bone, optimizing performance. Below are a few illustrative examples.

**TAMPER-EVIDENT LOGGING.** To investigate and recover from security breaches, people crucially rely on system logs for information of system execution history. Still, logs are only useful as long as attackers cannot tamper with them. But system logs are usually the first target of an experienced attacker to hide the traces of the intrusion. To cope with the situation above, there have been a plethora of secure logging systems, from both academia and the industry. Surprisingly, a recent work demonstrates a race attack that breaks all current logging systems in Linux. Resisting this attack requires that the logging system must operate in the kernel space, producing an authenticity tag for each log message right after it is generated. This levies an exacting performance requirement, eliminating most prior designs. The prior work then builds the first kernel-based logging scheme (KennyLoggings) with reasonable overhead, but it till leaves much to be desired.

In my Usenix Security 2022 paper, my coauthors and I design QuickLog2, another logging system, that
improves KennyLoggings in several fronts. For example, QuickLog2 reduces the logging overhead by a factor of two, and eliminates all the extra storage cost. Even better, this performance gain comes in tandem with stronger security. In particular, KennyLoggings is vulnerable to what is known as the truncation attack, but QuickLog2 can provably resist it. Moreover, KennyLoggings hoards 3.2MB of sensitive material in the kernel memory, making it susceptible to side-channel attacks. In contrast, QuickLog2 only uses 48B of keying material in memory. Thanks to these contributions, our work received the Best Paper Award at Usenix Security 2022.

To achieve such high speed with provable security, we have to surmount several obstacles. A major issue here is that logs are usually very short. Practitioners have long abandoned the provable-security route when they have time-critical applications that need to authenticate mostly very short messages. Such applications (including KennyLoggings) are now usually built on top of SipHash, a Message Authentication Code (MAC) construction whose security is validated by cryptanalysis only. Questioning the wisdom of this direction, we propose a new design paradigm for small-data MAC and build a provably-secure scheme that beats SipHash.

To reduce the storage overhead due to the authenticity tags, we unearth a neat idea for aggregating MAC tags in a prior work that has been overlooked in the literature. Prior logging systems never use this trick because of an obvious attack; as a result, they have to resort to much costlier mechanisms. Surprisingly, we point out that with some mandated restrictions to thwart the race attack above, this trick actually works.

Circuit Garbling. The main goal of my doctoral dissertation is to provide a practice-oriented treatment for garbled circuits (GCs), a central tool in secure distributed computing. GCs were originally invented as a solution for secure multiparty computation (MPC): distrustful parties need to collaborate on the combined data, but none wants to share its secret data with the partners. Thirty years after their birth, GCs have since found numerous uses beyond the scope of MPC, but there remained no definition of what GCs were supposed to deliver. Consequently, each time developers implemented an instantiation of GCs, they had to prove security for a particular setting, and thus deprived other applications of faster GCs. On the other hand, since GCs are complex, proving security of protocols containing this tool is a daunting task. Practitioners therefore often chose a specific instantiation of GCs and claimed that it worked for their protocols without any proof. A few papers did justify the security of their protocols, but the proofs were complex and error-prone.

In a series of papers (CCS 2012, Asiacrypt 2012, S&P 2013), my coauthors and I sever GCs from the intertwined applications, and provide an abstraction that captures what this technique delivers for most of the known applications. This modular approach gives a simple, clean interface between applications and instantiations. Protocol designers can therefore choose a suitable notion of GCs and reduce the goal of the containing protocol to the chosen security guarantee. Our abstraction boundary allows us to identify security bugs in several well-known MPC papers. Finally, we give an implementation of GCs with an unprecedented speed: 7.25 ns for evaluating a gate, while the best previous works run in about 2 µs per gate. This underscores our thesis that once GCs are nicely formalized, the theory will lead to better schemes.

Ten years after the publication of my three papers, they have significantly contributed to the explosive growth of many GC-based applications, provided a cornerstone for subsequent optimizations, and helped realizing bugs in several well-known implementations. As a result, they have received more than 1000 citations, and their material has been used in cryptography classes at MIT, Stanford University, UC Berkeley, and many other universities.
Ongoing and Future Research

One of my current projects is to build FPE schemes that are both provably secure and practical, which is a long-standing question that has challenged a decade of intensive research effort. Such schemes are urgently needed, because the current (revised) FPE standards offer no theoretical guarantees. Even worse, one of them has recently received yet another devastating attack. Some building blocks for the provable-secure FPE schemes that I envision were laid down in my CRYPTO 2012 paper, and some analytic tools were developed in my work at CRYPTO 2017.

In a different direction, frustrated by the convoluted constructions of the random number generators in NIST SP 800-90A and their lack of explanation for design rationale, I am working on clean conceptual boundaries for modularizing their descriptions, a better understanding of why they work, and what they actually achieve. To realize this goal, I envision a new paradigm in building randomness extractors and novel analytic frameworks for these settings.

In another direction, I argue that the conventional way of proving security for symmetric encryption schemes is too restrictive, failing to capture physical aspects of real-world attacks. In particular, our current approach is to bound the success probability of an adversary solely in terms of its running time. But in real attacks, memory is an even more important cost metric, as GPUs or dedicated hardware provide adversaries with lots of computational power but limited space. I aim to find techniques to give tight time-memory tradeoffs for symmetric encryption schemes.