Viet Tung Hoang **Research Statement**

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, and programming languages. 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, or (3) leveraging programming-language methods to automate the design of cryptographic constructions. 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 the libsodium library and 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. Libsodium is also widely used by companies (such as Facebook, Adobe Commerce, and Splunk) and open-source applications (such as Discord, WordPress, and Vim).

- 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 October 2024, its market capitalization is 314 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 work on robust encryption gives a design paradigm for resisting several kinds of misuse in implementation, and for achieving the best possible authenticity in applications where we have to reduce the ciphertext length due to legacy issues or performance constraints. This has led to NIST's ongoing standardization of the Accordion mode of encryption.

- My doctoral thesis lays 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 1,300 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.[1] 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, cryptographers often follow the provable-security paradigm: first formalize a threat model, and then design a scheme that provably achieves security in this model. Provable security is not perfect, as (i) the model may not capture all possible attacks, (ii) the security proof must be based on a certain hardness assumption (such as factoring), and (iii) 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.

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

[1] O. Goldreich. My choices. `https://www.wisdom.weizmann.ac.il/~oded/my-choice.html`

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, 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.

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. Thanks to these contributions, our paper received the Best Paper Award at Usenix Security 2022, and was taught in a security class at UIUC.

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.

Ongoing and Future Research

Moving forward, I want to continue broadening the theory of cryptography to build practical systems. Below, I sketch a few directions for my ongoing and future work.

Crypto for big systems. Servers in Big Tech companies constantly have to deal with a massive amount of data and events, leading to unprecedented challenges for cryptography. Below, I give two illustrative examples that I have been actively working on.

▷ **Encryption:** For standard encryption schemes like GCM, you can only encrypt about 2^{52} bytes per key. Going beyond that would lead to a noticeable likelihood of disasters. There are tricks that can help to increase the threshold to, say 2^{68} bytes, but major providers like Amazon Web Services (AWS) still need to change key (for thousands of accounts) on a weekly basis, and the cost of key rotation is significant. Finding a more scalable encryption (without hurting performance) has been a pressing concern. Indeed, Amazon developers recently submit a request to NIST for a better encryption standard. In a preliminary work, I give a design that achieves nearly the speed of GCM but allows one to encrypt up to 2^{100} bytes of data, which reaches the theoretical limit of AES-based encryption. Even better, the scheme addresses several security glitches of GCM. I have been working on a proposal to Amazon to fund the project and more importantly, to pitch them for deploying it in AWS.

▷ **Logging:** Existing logging systems in Linux have been built on top of the Linux audit daemon `auditd`, but a recent work points out that on busy servers, `auditd` may slow down workloads from 2x to 8x, and as a result, lose nearly 95% of the log messages during peak time. This prior work suggests a promising approach to build logging systems via the eBPF framework that has been recently added to Linux. Unfortunately, the eBPF environment poses a lot of challenges for building a practical tamper-evident logging system, because hardware-accelerated cryptographic operations like AES-NI are not available, and the eBPF programming interface is severely restricted. Our preliminary work finds a way to overcome those obstacles, achieving just 2% overhead during peak time, with no data loss.

Concrete security for MPC. In this Big Data era, companies want to harvest more and more data from their customers, but people now are acutely aware of the importance of their online privacy. As a result, companies like Google and J.P. Morgan have increasingly used MPC tools so that they can collect certain statistics from users, but nothing beyond that. Still, most MPC work only provide asymptotic guarantees: a scheme would be deemed secure when the security parameter λ is big enough. The asymptotic approach is however at odd with practice, because one has to pick a specific value for λ and figure out how secure the resulting instance is. (It's desirable that λ is as small as possible, because the performance is inversely propositional to λ.) My goal is to give a concrete-security overhaul of MPC: (1) providing simple definitional frameworks so that it's easy to give concrete security bounds and catch implementation bugs, and (2) giving tighter analyses for major tools to get better security/performance tradeoffs.

Further directions. My work has focused on building better cryptographic tools, but in practice, regardless of how misuse-resistant the tools are, practitioners still make disastrous mistakes when they have to build new and complex systems. An example is end-to-end encrypted (EE2E) cloud storage where major providers like MEGA and Nextcloud were recently found to have multiple critical bugs in their designs. I plan to re-align a part of my research efforts towards designing and building practical, provable-secure, and deployable systems, starting with EE2E cloud storage.