Graduate Research Plan: Unified Formal Verification of the Signal Protocol

Background: End-to-end encryption represents the foundation for security, privacy, trust, and compliance for all services on the internet. Many end-to-end encryption protocols exist, all serving different purposes and use cases. In particular, the Signal protocol is ubiquitous in secure instant messaging, and variations of the protocol are employed in virtually every modern instant messaging application, cumulatively serving billions of users. Since Signal was introduced in 2014, security properties such as forward secrecy, post-compromise security, and authentication have been proven about the protocol using several proof methods and with a wide range of abstractions and assumptions. The Signal protocol has been proven secure by abstracting the primitives to algebraic terms as well as using game-based computational hardness reductions with respect to an arbitrary polynomial-time attacker, both using hand-written [1] and mechanized arguments [2]. Additionally, several prominent implementations of Signal exist, most notably the Signal Foundation's Rust implementation of the protocol.

The rich line of work formally reasoning abrout Signal has consistently sought to close the specification-implementation semantic "gap" between the mathematical formalisms used to reason about Signal's cryptographic scheme and its respective real-world implementation, ultimately seeking a highly usable and functional implementation of the protocol that maintains provable high-level security properties. And yet, little protocol verification work exists that concurrently considers high-level cryptographic specifications and the real-world protocol implementation [3]. In general, cryptographers prove protocols and primitives maintain security properties in an abstract, theoretical settings, and implementers generally only prove their code is correct at the functional level, if at all. In this respect, theoretical cryptographers and cryptographic engineers are relatively siloed off from each other. I argue, in order to maximize the real-world security of Signal, the cryptographic specification and the protocol implementation must be formally reasoned about simultaneously, such that high-level desirable cryptographic properties can be effectively propagated to the implementation.

Proposal: How do we refine high-level cryptographic specifications and proofs in an abstract setting to specifications and proofs at the implementation level? To approach this problem, I propose a layered methodology for reasoning about the Signal protocol, combining symbolic methods for reasoning at the extrinsic cryptographic specification level and Hoare logic-based specification techniques at the intrinsic implementation level. My research objective is to tighten the specification-implementation semantic gap for the Signal protocol, propagating its abstract cryptographic security proofs to the real-world protocol implementation, thereby maximizing the effective security guarantees of the resulting implementation.

***I* – Base Signal Specification and Implementation:** The first phase of my research will focus on building a base specification and implementation of which to base the rest of this project on. For the high-level cryptographic specification, I will adapt an existing Signal model in F* with formal security proofs constructed using a dependent types-based formalization of symbolic cryptography [4]. I choose F*, a dependent types-based verification language, as it maintains first-class support for refinement types, formally verified implementation extractions with robust toolchains, and a history of success in cryptographic protocol verification [7]. I will validate and refine the F* model by comparing the formalization and proven lemmas to other symbolic and computational formalizations of Signal [1, 2, 9].

As for the implementation, I will employ a chosen version of the Signal Foundation's Rust implementation of the Signal protocol. Building off prior work such as the Vale project [3], which verified cryptographic assembly code using Dafny's Hoare-logic SMT-based intrinsic specifications [5], I will employ Verus, a tool for writing Dafny-style intrinsic specifications and proofs for Rust code [6]. Using Verus, I will construct a set of functional specifications and proofs for the Signal implementation. In conjunction with this, I will analyze the selected Signal implementation with the Presti static analyzer and Kani model checker to ensure my implementation validation is thorough and complete.

***II* – Bi-directional Extraction:** A crucial step in unifying the specification-implementation semantic gap between cryptographic specification and protocol implementation will be exploring bi-directional extraction. This approach will entail extracting a Rust implementation from F*, and an F* model from the Rust implementation. F* is one of the few languages that supports such verified extraction processes, allowing movement between high-level models in F* and lower-level implementation code [7]. I will

utilize existing F* and Rust toolchains to perform this extraction [7, 8], then I will compare the extracted F* and Rust code with the respective hand-written specifications in both F* lemmas and Verus. By iteratively refining both directions of the extraction processes, I will tightly align the formal cryptographic properties of the Signal protocol with the real-world implementation as closely as possible.

***III* – Cryptographic Guarantee-carrying Extractions:** The final phase of my project is to enable cryptographic guarantee-carrying extractions between cryptographic proofs in F* and the Rust implementation's intrinsic specification in Verus. This will involve extending F*'s existing extraction infrastructure to carry security guarantees in the forms of functional specifications. Specifically, I will develop a methodology for extracting the formal lemmas proven in F* to Verus specifications within the Rust codebase, ensuring that the high-level cryptographic properties are preserved during this translation.

By reasoning about the semantics of the symbolic cryptography formalization in F* and their correspondence to Verus's intrinsic specifications, I will construct a formal argument, similar in flavor to what was done for Vale [3], that the cryptographic security guarantees proven about Signal in F* are maintained within the Rust implementation. This methodology will result in a sound, consistent framework for reasoning about cryptographic protocols from specification to implementation.

Once my extraction framework is complete, I will open-source my Signal model and implementation, as well as collaborate with the Signal open-source community to discuss and integrate my research. I will also seek to publish my results in top security conferences.

Intellectual Merit: Retaining properties of interest when traversing levels of abstraction and expressibility is a rich and highly general problem in computer science. Although this work proposes tackling a narrow and focused version of this problem — propagating cryptographic specifications to protocol implementations — the novel approach proposed in this work of explicitly studying the relationship between extrinsic specifications with high-level properties and intrinsic, functional specifications in order to close the ultimate specification-implementation semantic gap has the potential to be generalizable. Besides cryptographic protocols such as Signal, plenty of other critical computer systems exhibit high-level properties of strong interest that are only reasoned about in an abstract, extrinsic setting. For example, distributed and concurrent systems are often specified with process algebras, automata, temporal logics, or concurrent separation logics and reason about safety or liveness properties in abstraction from the respective implementation. If the approach I propose is effective for Signal and its associated symbolic specification, a natural next step, besides applying the approach to more cryptographic protocols, is to consider distributed and concurrent systems.

Broader Impacts: This research will have profound implications for the security and privacy of global communications. By closing the semantic gap between cryptographic specifications and real-world protocol implementations, this work directly addresses a core challenge in ensuring the reliability and security of widely-used encryption protocols, such as the Signal protocol. Given that Signal and its derivatives secure the communications of billions of users, any advancements in the verifiability of its implementation will improve the security and privacy of everyday digital interactions for a substantial portion of the global population.

References:

[1] K. Cohn-Gordon et al., “A Formal Security Analysis of the Signal Messaging Protocol.” S&P 2017.
[2] K. Bhargavan, C. Jacomme, F. Kiefer, and R. Schmidt, “Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging.” USENIX 2024.
[3] B. Bond et al., “Vale: Verifying High-Performance Cryptographic Assembly Code.” USENIX 2017.
[4] K. Bhargavan et al., “DY*: A Modular Symbolic Verification Framework for Executable Cryptographic Protocol Code.” S&P 2023.
[5] K. Rustan et al., “Dafny: An Automatic Program Verifier for Functional Correctness.” LPAR 2010.
[6] A. Lattuada et al., “Verus: Verifying Rust Programs using Linear Ghost Types.” OOPLSA 2024.
[7] J. Protzenko et al., “Verified Low-Level Programming Embedded in F*.” ICFP 2017.
[8] D. Merigoux, denismerigoux/rox-star. Available: https://github.com/denismerigoux/rox-star
[9] **J. Ginesin** and C. Nita-Rotaru, “The Matrix Reloaded: A Mechanized Symbolic Analysis of the Matrix Cryptographic Protocol Suite” arXiv2408.12743