Verification of SHA-256 via Deep Specifications

Jonathon Cai
Advisor: Zhong Shao

1 Overview

As software becomes increasingly more complex and critical in our daily lives, software bugs become increasingly more deleterious. Computer security mechanisms, in particular, should ensure that data is protected properly. Bugs in implementations of computer security mechanisms threaten the integrity of software infrastructure that underlies modern society. For example, in April 2014, the HeartBleed vulnerability was discovered in the OpenSSL cryptography library, an implementation of the Transport Layer Security protocol. This bug permitted hackers to steal private keys and other sensitive information from servers. The fundamental cause for this bug was the inclusion of a flawed code extension in the OpenSSL software library in 2011.

This bug affected approximately 17% of the world’s servers [1], compromising around half a million certificates issued by authorities. It is likely that bugs of similar magnitude lie undiscovered in existing code bases. In addition, whenever someone writes new code or pushes a patch, the possibility of introducing bugs always exists. Such grave concerns motivate a formal verification approach to computer security, in which code is formally proved to work correctly by a proof assistant such as Coq.

2 Previous Research

The FLINT research group at Yale aims to develop a novel and practical programming infrastructure for constructing large-scale certified systems software. Recently, a paper by the FLINT group detailed a deep specification approach to verifying the CertiKOS operating system kernel. The authors specify and verify an OS kernel in 37 carefully described abstraction layers, employing modularity to reduce dependencies between kernel layers. The effort took them less than one person year. In contrast, the seL4 team (the first to verify an operating system kernel) used a monolithic approach. The seL4 team encountered kernel interdependencies that result in more complex invariants, which might explain why their verification effort took about 11 person years [4].

Although the layered, deep specification approach was used to verify an operating system kernel, the approach should work well in other software contexts such as computer security. For instance, cryptographic primitives such as the SHA-256 primitive are utilized in the OpenSSL HMAC algorithm. Hence, computer security architecture can be viewed in a modular fashion.

Andrew Appel’s group at Princeton recently published a pair of papers on verification of an implementation of the SHA-256 cryptographic primitive in OpenSSL and the OpenSSL
HMAC algorithm [2] [3]. Appel’s group used the Verified Software Toolchain (VST) developed by their group, which employs separation logic. We believe that separation logic results in a significantly more time-consuming and complex verification effort than our deep specification approach.

3 Project Description

The goal of this project is to verify in Coq the SHA-256 cryptographic primitive via deep specifications, a philosophy on formal verification championed by the FLINT research group at Yale. A verification of SHA-256 has already been performed by Andrew Appel in 2015, using the Verified Software Toolchain (VST). VST employs separation logic, whereas deep specifications do not. By implementing a verification of SHA-256 with deep specifications, we may compare the efficacy of the two philosophies directly. We suspect that verifying security algorithms via deep specifications should enjoy similar advantages already demonstrated in [4].

4 Checkpoints

1. Understanding existing VST approach to verifying SHA-256; the source code is readily available online at https://github.com/PrincetonUniversity/VST/tree/master/sha.
2. Familiarize oneself with the FLINT group’s philosophy on deep specifications.
3. Formally prove, in Coq, the SHA-256 primitive using deep specifications.
4. Audit the deep specification approach by comparing directly to the VST verification of SHA-256.

5 Timeline and Deliverables

The project will last over the course of approximately 10 weeks. The precise timeline is difficult to determine beforehand. The project will involve researching both the theoretical background and Coq implementations of two formal verification philosophies, separation logic in VST and deep specifications in CertiKOS. Coq code will be written in attempts to verify SHA-256 via the FLINT group’s philosophy. It may turn out that I complete the Coq code quickly, and hence more time could be spent on comparing the two approaches.

At the end of the semester, I will write a final report on my results.

References

