Attack-preserving Program Transformations

K. Bhargavan

Internship Proposal 2010
(2-3 months)

Background The security of a distributed application typically depends on the correctness of an underlying *cryptographic protocol* that protects messages exchanged butween different participants. In previous work, we verified implementations of widely-deployed cryptographic protocols, such as TLS [Bhargavan et al., 2008a] and Windows Cardspace [Bhargavan et al., 2008b], using specialized state-of-the-art cryptographic provers, such as ProVerif [Blanchet, 2001].

Whole-program security verification does not scale beyond a few thousand lines of code. Even with careful rewriting of the source code, verification may take hours or sometimes not terminate. However, large portions of protocol implementations are irrelevant for security, for example, the code for message formatting and parsing. By carefully separating the security-critical core of an implementation and by conservatively approximating the rest of its code (treating it as untrusted), we can significantly reduce the code passed to specialized protocol verifiers.

In earlier work, Hui and Lowe[Hui and Lowe, 2001] show how to transform a cryptographic protocol model into a simpler model such that if there is is any attack on the original model, then that attack is also possible on the transformed model. They demonstrate that by applying such transformations, complex cryptographic protocol models can be brought within the reach of automated verification techniques. We propose to extend this idea and apply it directly to protocol source code.

Project In this internship, we shall design and implement program transformations that preserve all attacks on the source program while reducing the size and complexity of the verified model. Hence, by verifying the security of the transformed program, we can prove the security of the source program. We shall prove the correctness of these transformations using a combination of standard results in type theory, program analysis, formal cryptography, and programming language semantics. We shall evaluate our verification method through an extended case study of a protocol implementation for web services security.

Keywords: Functional programming, Security protocols, Cryptography, Type theory, Program verification.

References

K. Bhargavan, C. Fournet, R. Corin, and E. Zalinescu. Cryptographically verified implementations for TLS. In *15th ACM conference on Computer and Communications Security (CCS'08)*, pages 459–468. ACM, 2008a. PDF.

K. Bhargavan, C. Fournet, A. D. Gordon, and N. Swamy. Verified implementations of the Information Card federated identity-management protocol. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pages 123–135. ACM, 2008b. PDF.

B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 82–96. IEEE Computer Society Press, 2001. ProVerif Homepage.

M. Hui and G. Lowe. Fault-preserving simplifying transformations for security protocols. *Journal of Computer Security*, 9(1/2):3–46, 2001.