

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

244 WOOD STREET
LEXINGTON, MASSACHUSETTS 02420-9185

SCIENCE OF SECURITY CYBERSECURITY PAPER COMPETITION -- NOMINATION

TITLE: Adversary Safety by Construction in a Language of Cryptographic Protocols

AUTHORS: Timothy Braje, Martine Kalke, Adam Chlipala, Robert Cunningham, et al.

Please consider this write-up as an enthusiastic nomination for this important, foundational work to be recognized by the science of security award.

Correctly implementing cryptographic protocols is notoriously difficult. In recent years, researchers have begun to more widely apply formal verification techniques to help prevent exploits due to both protocol design problems as well as implementation issues. To put it succinctly, once a cryptographic protocol has been analyzed by a cryptographer and is believed to be secure, there are a number of things that can go wrong between that design and what you actually run on your computer. To make matters worse, developers are often asked to assemble custom cryptographic protocols using off-the-shelf primitives. Even if the developer uses secure components, bugs can be (and are) introduced when those primitives are combined in incorrect ways or used improperly. Researchers have developed a variety of verification tools to try and help with this problem; however, we argue that the state of the art is still too difficult to use. The most widely used tools, for example, require a protocol analyzer to write down a description of the protocol using message passing charts, model the ways in which an adversary could possibly interfere, and then run the tool – none of this syntax is familiar, nor does it look like programming.

In this paper, the authors set out to answer an important question “can we build a custom language for cryptographic protocols that reduces adversary safety checks to something akin to type-checking in a statically typed language like java?” They designed a language, formalized it within the Coq proof assistant, and proved that any protocol written in that language and that passes their “type-checker” will have the same outcome whether there is no adversary or one running arbitrary code. They developed a prototype that could become a foundation for developer-friendly tooling which helps developers with no formal verification experience implement cryptographic protocols and maintain proof-assistant level security guarantees automatically.

After an introduction and motivation, the paper walks through (in Section II) what steps a developer may perform when faced with implementing a new cryptographic protocol. The paper gives a flavor of the typical kinds of mistakes a developer may make, and then shows how their tool catches these mistakes at “compile time.” At the same time, the paper gives a flavor of what it is like developing a protocol within their language (called SPICY), showing that it has a familiar syntax similar to common programming languages. The next section discusses the technical details of the language formalization, the safety rules the type checker verifies, and the overall safety result (Strong Preservation Theorem) which states their proven safety guarantees. In the evaluation section, the paper goes through a handful of realistic real-world protocols that were verified using SPICY and discusses strengths and limitations of the tooling. The paper then compares to related work and discuss plans for the future.

Overall, the paper is an important step in creating the rigorous mathematical foundations and increasing the practicality of leveraging formal methods for cryptographic protocol design and implementation, and incorporating formal methods into the continuous integration process.

Published in: IEEE Computer Security Foundations 2022

<https://www.computer.org/csdl/proceedings-article/csf/2022/841700a396/1F9QlqtU5EI>

[Roger Khazan, PhD](#)

Leader of [Secure, Resilient Systems and Technology Group](#)