Reconciling Provable Security and Practical Cryptography: A Programming Language Perspective

Submitted by Katie Dey on Thu, 04/23/2015 - 6:44pm. Contributor: Gilles Barth

Abstract:
We need cryptographic software that we can trust. However, our trust in widely used cryptographic software has been repeatedly undermined by implementation bugs and side-channel attacks. At the heart of these issues lies an uncomfortable gap between provable security and practical cryptography; in short, provable security has been immensely successful in analyzing the security of cryptographic constructions, but relies on an idealized model (the computational model of cryptography) that elides many common causes of attacks. Real-world cryptography aims to address this problem, by applying the principles of provable security to analyze widely used protocols under the light of more precise security definitions and adversary models. This is an essential step towards making provable security practically relevant, but models from real-world cryptography remain too abstract to capture side-channel attacks and implementation bugs. In the talk, I will outline a further step to narrow the gap between provable security and practical cryptography. Our approach builds on our earlier work in computer-aided cryptography, and in particular on EasyCrypt, a computer-aided tool which supports deductive verification of reductionist proofs in cryptography and has been used to verify prominent examples of cryptographic constructions and protocols, including PKCS-OAEP encryption and the Naxos protocol, and is currently applied to multi-party and verifiable computation, and to voting systems. In addition, our approach uses some recent development in programming languages and formal verification, and more specifically certified static analyses, verified compilers, and type systems and program logics for low-level languages.

In its simplest form, our approach consists of three steps: i. verify black-box security of a cryptographic system in the computational model, using tools for computer-aided security like EasyCrypt; ii. carry black-box security to assembly implementations, using verified compilers; iii. prove that the assembly implementation does not leak, using certified static analysis. I will illustrate the effectiveness of our approach by instantiating it to the notable cases of masked implementations (protection against DPA attacks) and constant-time implementations (protection against cache attacks). Focusing on preventing leakage of implementations, I will present the principles and experimental results for a new verifying compiler that generates efficient masked implementations of blockciphers (AES, Simon. . . ) at arbitrary orders, and uses a fine-grained information flow type system to verify that the generated code is secure. The correctness of the certifying compiler rests on a deep connections between leakage in the threshold probing model of Ishai, Sahai and Wagner (which has recently been proved equivalent to the nosiy leakage model of Chari et al), and a generalization of probabilistic non-interference, and on a stronger notion of non-interference, that justifies the compositional reasoning of the type system.

I will further provide a broader perspective on applications of our approach to more complex cryptographic systems, and with challenges for verified programming languages and proof engineering.

For further background information, please consult: www.easycrypt.info.

Biography:
Gilles Barthe is Research Professor at IMDEA Software Institute in Madrid since 2008. His research interests include programming languages and program verification, software and system security, cryptography, formal methods and foundations of mathematics and computer science. His latest research has focused on building foundations and tools for computer-aided cryptography and privacy.