Formal Methods for Secure Devices

Robert Cordery
Pitney Bowes Inc.

Protocols implemented in hardware impact long lead-time items and must be finalized early in product development process. Development phases [1] are: conception; specification and architecture; detailed engineering; support process development; and qualification. With pressure for short development times, these steps usually overlap. Developers must finalize components with long lead-time early in the development process, and these parts are expensive to change. They, therefore, specify and test the component design using formal tools. ASICs, including those that support trusted protocols, must undergo extensive design verification before fabrication. Protocols that impact ASIC design must be finalized early in the detailed engineering phase. Unfortunately, this early in development, the system requirements are not always final. Tools that routinely need outside experts to operate do not fit this development process well.

Companies developing secure devices define protocols early in the process. In some cases the feasibility of the project depends on providing needed functionality while enforcing the security policy. The developer must avoid a high risk of rework and long schedule delays caused by errors in the protocol design. Formal methods must, therefore, support early specification and verification.

True or false answers to a set of questions about protocol security do not really help. There are too many alternatives to fix the problem. Rota [2] observes that "Verification alone does not give us a clue as to the role of a statement within the theory; it does not explain the relevance of the statement. In short, the logical truth of a statement does not enlighten us as to the sense of the statement." Mechanical proofs do not always elucidate how the protocol supports the security policy.

Understanding is as important as verification in product development. Improved understanding leads to simplified, stronger protocols. Analyzing protocols by considering possible threats does not prove anything; it does enlighten. Observing how a protocol fails or survives under attack clarifies the meaning of the steps in the protocol. Frequently, experts in protocol analysis and protocol logic improve and compare their formal method tools by examining how they fail to detect a possible attack.

Abadi and Needham [3] provided prudent design principles that are neither sufficient nor necessary for security. Protocol developers need such “Design for Security” principles. Tools and methods that foster correct protocol design before formal verificaton help developers avoid early errors.

Specifying the protocol in an unambiguous language, and considering the implications of each principle for each message, eliminates many errors before analysis. Performing by hand a logic-based analysis of the protocol, the developer can compare what beliefs are true at each step with what beliefs he expected. This effort provides verification of some security properties of the protocol within the chosen logic. As important, it provides a deeper understanding of the working of the protocol.

Formal policies and specifications provide an unambiguous logical interpretation of ambiguous natural language policies and specifications. Errors result from selection of inappropriate interpretations. This is yet another reason that formal methods should clarify the meaning and security of a protocol.

System architecture is often defined in terms of Objects. Formal tools should support Object-Oriented development. The developer must interpret the system security policy as Object requirements. Reused Objects have known security properties. How can the developer reuse Object security properties in support of system security policy?

A simple core security policy provides a basis for designing the protocol. The policy should consist of only a few very clear and mandatory requirements. An early formal specification of this policy and the proposed protocol clarifies the function of the elements in messages.

References

1. M. McGrath, M. Anthony, A. Shapiro, Product Development: Success Through Product and Cycle-time Excellence, Butterworth-Heinemann, Boston, 1992.
2. G-C Rota, Indiscreet Thoughts, Birkhauser, Boston, Basel, Berlin, 1997.
3. M. Abadi and R. Needham, Prudent Engineering Practice for Cryptographic Protocols, 1994 IEEE Computer Society Symposium on Research in Security and Privacy, pp 122-136, May 1994.