Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 125 (2005) 13–24
www.elsevier.com/locate/entcs

Extending Security Protocol Analysis: New Challenges[1]

Mike Bond[2] Jolyon Clulow[3]

Computer Laboratory
University of Cambridge
Cambridge, UK.

Abstract

We argue that formal analysis tools for security protocols are not achieving their full potential, and give only limited aid to designers of more complex modern protocols, protocols in constrained environments, and security APIs. We believe that typical assumptions such as perfect encryption can and must be relaxed, while other threats, including the partial leakage of information, must be considered if formal tools are to continue to be useful and gain widespread, real world utilisation. Using simple example protocols, we illustrate a number of attacks that are vital to avoid in security API design, but that have yet to be modelled using a formal analysis tool. We seek to extract the basic ideas behind these attacks and package them into a wish list of functionality for future research and tool development.

Keywords: Security APIs, Formal Methods, Protocol Analysis, Perfect Encryption, Information Leakage

1 Introduction

Security protocols have been designed, studied and attacked for over thirty years. Today, formal analysis is becoming a popular tool for assisting in the design process. However, the assumptions that formal tools make and the restrictions they put on the description and analysis of behaviour conspire to

[1] The authors wish to acknowledge the generous funding of the CMI Institute and the Cecil Renaud Educational and Charitable Trust.
[2] Email: Mike.Bond@cl.cam.ac.uk
[3] Email: Jolyon.Clulow@cl.cam.ac.uk

doi:10.1016/j.entcs.2004.09.036

limit their scope – preventing their application to harder protocol design problems of today. In particular, the design of security APIs as well as conventional protocol design in constrained environments (such as within embedded systems) cannot benefit fully from the existing tools because of these impractical assumptions and restrictions. While designers can achieve security through systematic application of rules of thumb for fulfilling the assumptions, the results tend to be over-engineered and impractical to deploy. Instead, we propose that the time has come to extend these tools to relax the assumptions on the models they analyse.

We describe some well-known mistakes that need to be avoided in good protocol and API designs, yet which cannot be reasoned about in the abstract models used by formal tools. These mistakes are of particular significance as they are regularly discovered within security APIs, but are illustrated for simplicity with example security protocols.

We believe that formal reasoning about many of these lower-level attacks is possible, and present this as a challenge for the formal methods community to adapt and extend their tools, to assure their continued usage and eventual widespread acceptance into the design process.

2 Analysing Protocols with Formal Methods

Numerous tools and techniques for formal analysis of security protocols exist; they can be broadly split into model checkers, theorem provers and formal logics.

- *Model checkers* explore a state space, examining methodically whether certain requirements hold in each state of the model. Some can also reason about equivalence between state space models, or use mathematical techniques to reason about entire sets of states simultaneously. Theoretically, they can examine the entire state space and can give a similar assurance of correctness as that provided by a theorem prover (in practise however, the problems set by users are often too hard and defeat analysis, or are deliberately simplified to ensure solubility).
- *Theorem provers*, in contrast, search at a higher level of abstraction for chains of logic that constitute a compelling proof that a particular property always holds. Alternatively, they may find a counter-example in the process. Various proof search strategies are used, often based on the basic resolution strategy proposed by Robinson [19].
- Finally, *formal logics* provide the user with notation and precise definitions of properties, which help the user to perform intuitive reasoning more rig-