
Véronique Cortier  
LORIA, CNRS  
cortier@loria.fr

Steve Kremer  
INRIA Nancy  
steve.kremer@inria.fr
Editorial Scope

Topics

Foundations and Trends® in Programming Languages publishes survey and tutorial articles in the following topics:

- Abstract interpretation
- Compilation and interpretation techniques
- Domain specific languages
- Formal semantics, including lambda calculi, process calculi, and process algebra
- Language paradigms
- Mechanical proof checking
- Memory management
- Partial evaluation
- Program logic
- Programming language implementation
- Programming language security
- Programming languages for concurrency
- Programming languages for parallelism
- Program synthesis
- Program transformations and optimizations
- Program verification
- Runtime techniques for programming languages
- Software model checking
- Static and dynamic program analysis
- Type theory and type systems

Information for Librarians

Foundations and Trends® in Programming Languages, 2014, Volume 1, 4 issues. ISSN paper version 2325-1107. ISSN online version 2325-1131. Also available as a combined paper and online subscription.

Véronique Cortier
LORIA, CNRS
cortier@loria.fr

Steve Kremer
INRIA Nancy
steve.kremer@inria.fr
Contents

1 Introduction ................................................. 2

2 Running example ........................................... 5
   2.1 The Needham Schroeder public key protocol ............. 5
   2.2 Lowe’s man in the middle attack ......................... 7

3 Messages and deduction ..................................... 10
   3.1 Terms ................................................. 10
   3.2 Message deduction ..................................... 13
   3.3 An algorithm to decide message deduction ............. 15
   3.4 Exercises .............................................. 18

4 Equational theory and static equivalence ................... 20
   4.1 Equational theories ..................................... 20
   4.2 Static equivalence ..................................... 24
   4.3 Exercises .............................................. 30

5 A cryptographic process calculus ........................... 32
   5.1 Syntax and informal semantics ........................... 33
   5.2 Modelling protocols as processes ......................... 35
   5.3 Formal semantics ....................................... 38
   5.4 Exercises .............................................. 48
6 Security properties
   6.1 Events .................................................. 50
   6.2 Secrecy .................................................. 51
   6.3 Authentication ......................................... 54
   6.4 Equivalence properties ................................ 60
   6.5 Exercises ............................................... 70

7 Automated verification: bounded case .......................... 72
   7.1 From protocols to constraint systems .................. 73
   7.2 Constraint solving ...................................... 77
   7.3 Exercises ............................................... 83

8 Automated verification: unbounded case ......................... 85
   8.1 Undecidability .......................................... 87
   8.2 Analysis of protocols with Horn clauses .............. 88
   8.3 Exercises ............................................... 100

9 Further readings and conclusion ................................ 101

References ....................................................... 107
Abstract

Security protocols are distributed programs that aim at securing communications by the means of cryptography. They are for instance used to secure electronic payments, home banking and more recently electronic elections. Given the financial and societal impact in case of failure, and the long history of design flaws in such protocols, formal verification is a necessity. A major difference from other safety critical systems is that the properties of security protocols must hold in the presence of an arbitrary adversary. The aim of this paper is to provide a tutorial to some modern approaches for formally modeling protocols, their goals and automatically verifying them.

DOI: 10.1561/2500000001.
Security protocols are used to protect electronic transactions. The probably most used security protocol is the SSL/TLS protocol which underlies the https protocol in web browsers. It may be used for electronic commerce, or simply to encrypt web search queries on their way between the host and the search engine. There are of course many other protocols in use, e.g. to authenticate to providers on mobile phones or withdraw cash on an ATM. Moreover, the digitalization of our modern society requires the use of security protocols in an increasing number of contexts, such as electronic passports that may include RFID chips, electronic elections that allow for Internet voting, etc.

We may think of security protocols as distributed programs that make use of cryptography, e.g. encryption, to achieve a security property, such as confidentiality of some data, e.g. your credit card number. Given the difficulty of designing correct distributed systems in general, it is not surprising that many flaws were discovered in security protocols, even without breaking the underlying cryptography. During the last 30 years many research efforts were spent on designing techniques and tools to analyze security protocols. One may trace this line of work back to the seminal work of Dolev and Yao [1981] who
pioneered the ideas of an attacker who completely controls the communication network, has an unbounded computational power, but manipulates protocol messages according to some predefined rules, idealizing the protections offered by cryptography. These techniques not only allowed to better understand the principles underlying secure protocol design, but also resulted in mature tools, for automated protocol analysis, and the discovery of many attacks. For example, while designing a formal model of Google’s Single Sign-On protocol, that allows a user to identify himself only once and then access various applications (such as Gmail or Google calendar), [Armando et al., 2008] discovered that a dishonest service provider could impersonate any of its users at another service provider. This flaw has been corrected since. [Basin et al., 2012] have identified flaws and proposed fixes for the ISO/IEC 9798 standard for entity authentication, using automated protocol verification tools. The standard has been revised to include their proposed amendments. [Bortolozzo et al., 2010] designed a dedicated analysis tool for hardware security tokens that implement the PKCS#11 standard. The tool automatically reverse-engineers the tokens to extract its configuration, builds an abstract model to be analyzed and verifies the attack on the token if an attack is found. They were able to find unknown attacks on more than 10 commercial tokens.

This paper proposes a tutorial, presenting modern techniques to model and automatically analyze security protocols. Given the large body of work in this area we do not aim to be exhaustive and only present some selected methods and results. We expect that this tutorial could serve as a basis for a master, or graduate course, or allow researchers from different areas to get an overview of the kinds of techniques that are used. The outline of the tutorial is as follows.

- We first present an informal description of our running example, the Needham Schroeder public key protocol that we used for illustration purposes in the remainder of the paper.

- Then, we explain how protocol messages can be modeled as first order terms, and how adversary capabilities can be modeled by an inference system. We also provide a decision algorithm for deduction, i.e. the adversary’s capability to construct new messages.
Introduction

- Next, we introduce a more general model, based on equational theories. We revisit deduction and define a notion of message indistinguishability, called static equivalence. We again provide a decision procedure for static equivalence for a simple equational theory representing symmetric encryption.

- We continue by introducing a process calculus, the applied pi calculus, which we use to model protocols. One of the main differences with the original pi calculus is that the calculus allows communication of messages represented by terms, rather than only names. We illustrate how protocols can be conveniently modeled in this formalism.

- Next we discuss how we can express security properties of protocols modelled in the applied pi calculus. We cover different flavors of confidentiality, authentication, but also anonymity properties, expressed as behavioral equivalences of processes.

- We go on discussing automated verification. We first consider the case when protocol participants only execute a bounded number of sessions. We present a decision procedure based on constraint solving which allows to decide secrecy in this setting.

- Finally, we show that the general case, where the number of sessions is unbounded, is undecidable. We show that nevertheless it is possible to design tools that are able to analyze protocols. This comes at the cost that termination is not guaranteed. In particular, we present an approach based on a representation of the protocol and the adversary as Horn clauses and describe a resolution based procedure implemented in the ProVerif tool.

- We conclude the tutorial by briefly discussing some other approaches for automated verification and other directions in this research area.


References


References


References


References


References


References


