Modelling and Analysing of Security Protocol: Lecture 7

Automatically Checking Protocols

Tom Chothia
CWI
Demo. of adding protocols to the JAPE tool for BAN logic.
The rest of the course

• Today: 5th Oct, Automatic protocol verification

• 12th Oct, Protocols for anonymity
  (homework on BAN logic and ProVerif)

• 19th Oct, Model Checking & Fair exchange protocols.

• 26th Oct moved to 29th Oct, 11:15 to 13:00

• 29th Oct & 2nd, 9th, 16th, 23rd Nov
  Student presentations

• 30th Nov, Summary Lecture.
Introduction

• This lecture describes a tool for automatically checking protocol.

• Analysis cannot be prefect (protocol security is equivalent to the halting problem).

• One of the main advantages of using a tool is to help you think hard about a protocol!
Automatic Protocol Tools

We will look at ProVerif, other tools *include*:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Model</th>
<th>Who</th>
</tr>
</thead>
<tbody>
<tr>
<td>Athena</td>
<td>Strand spaces</td>
<td>Mitire</td>
</tr>
<tr>
<td>FDR</td>
<td>CSP</td>
<td>Oxford Univ.</td>
</tr>
<tr>
<td>Analyzer</td>
<td>own</td>
<td>N.R.L.</td>
</tr>
<tr>
<td>PaMoChSa</td>
<td>Crypto-CCS</td>
<td>Pisa Univ.</td>
</tr>
<tr>
<td>Murphi</td>
<td>model-checking</td>
<td>Stanford</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Automatic tools

• To use an automatic tool you must understand:
  – How to specify your protocol.
  – What is being checked.
ProVerif

• To use an automatic tool you must understand:
  – How to specify your protocol.
    • Using Prolog style rules
    • Or applied pi-calculus format
  – What is being checked.
    • Secrecy of a message
    • Correspondence assertions.
This Lecture

• Quick introduction to Prolog
• A protocol as Prolog rules
• From Prolog to ProVerif
• Checking secrecy
BREAK
• From secrecy to authenticity
• Writing protocols in the pi-calculus
• Examples
Prolog

• A very powerful and very inefficient language.

• To write a program in Prolog, you tell the computer what you want, not do to do it.
Prolog: Facts and Rules

parent_of(eve, bob).  male(bob).
male(peter).  female(alice).  female(eve).

grandparent_of (X, Y) :- parent(X,Z),parent(Z,Y)
grandmother_of(X,Y) :- grandparent(X,Y),female(X).
sibling (X,Y) :- parent(Z,X), parent(Z,Y).
brother_of(X,Y) :- male(X), siblings (X,Y).
Prolog: Facts and Rules

> grandparent (eve, alice)
True

> Grandparent (eve, X)
  X = alice?
  > N
  > N
  X = peter?
  > N
  False
Prolog: Sort

sort (In, Out) :- permutation (In,Out), is_sorted (Out).
sort (In, Out) :- permutation (In,Out), is_sorted (Out).

is_sorted([x]).
is_sorted(x:y:xs) :- x <= y, sorted (y:ys)
Prolog: Sort

sort (In, Out) :- permutation (In,Out), is_sorted (Out).

is_sorted([x]).
is_sorted(x:y:xs) :- x =< y, sorted (y:ys)

permutation([], []).
permutation(x:xs,ys) :- member (x,ys), remove(x,ys,zs), permutation(xs,zs).
What Does This Have To Do With Protocols?

- We can write protocols as prolog rules and try to “solve” these to find a secret.

- If we can’t find the secrecy by any application of the rules then the protocol is “safe”.

- Problem: Prolog will not terminate.
ProVerif

• The facts in ProVerif describe what the attackers knows.

• The rules in ProVerif describe how the attacker can learn new facts...
  ... including learning new facts by using the protocol.

• The tool then tries to apply all the rules to learn a secret.
See handouts
The Denning-Sacco Public-Key Protocol

• A → B : $E_B(K_{AB}, T_A, \text{Sign}(K_{AB}, T_A))$

To test as a “secret” in ProVerif we model the protocol:

1. A → B : $E_B(\text{Sign}(K_{AB}))$
2. B → A : { secret }$^K_{AB}$
See handouts
Avoiding Non-Termination

ProVerif may not terminate, but almost always does.

- New values, such as nonce, are parameterized on current knowledge.
- ProVerif chains the rules together, rather than generating new facts and discards unhelpful rules.
- See sections 3 and 4 of the handout for more details.
This Lecture

• Quick introduction to Prolog
• A protocol as Prolog rules
• From Prolog to ProVerif
• Checking secrecy

BREAK

• From secrecy to authenticity
• Writing protocols in the pi-calculus
• Examples