DTU Informatics
Protocol Security Lab

02239 Data Security

Hound-out: September 27, 2017
Hand-in: October 24, 2017

We encourage group work, but the reports **MUST** be written individually. Submit via campusnet

The lab exercises use the protocol analyzer OFMC from

`http://www.imm.dtu.dk/~samo`

The distribution includes binaries for Windows and Mac. For compiling the sources yourself, you need the Glasgow Haskell Compiler

`http://hackage.haskell.org/platform/`

For the exercises please use OFMC with the following command line:[1]

```
ofmc --numSess 2 filename
```

You can test OFMC on the lecture example `nspk.AnB` in `examples/cj/6.7...`.

Exercise 1: Kerberos PKInit Kerberos (see for example Pfleeger & Pfleeger sec. 7.3) is a protocol for authentication in distributed system. A user first authenticates itself to an authentication server, e.g. using a password. The authentication server replies with a *ticket* and the client can then get access to other services with that ticket.

A ticket has typically a form like $\{|C, a, s, K, T, \dots|\}_{kas}$ where

- C is the client to whom the ticket is issued
- a is the issueing server
- s is the server that will accept the ticket
- K is a fresh symmetric key for communication between C and s
- T is a timestamp (the ticket may be valid only for a few minutes)
- Further, the ticket may contain a description of the services/resources the ticket gives access to
- The ticket is encrypted with kas which is a long-term symmetric key shared between a and s

[1] This bounds the state space to two sessions (i.e., each role can be instantiated by at most 2 participants).

There is a variant PKInit-Kerberos where the client uses public key cryptography (instead of a password) to authenticate itself to the authentication server. The formalization is found in `examples/classic/PKINIT.AnB`. This represents an early version that has an attack. **For simplicity** there is also a shorter version of the protocol in `examples/classic/PKINIT-short.AnB` that contains only those steps that are relevant for the attack. You can use this simplified version for **question 2 and 3**.

1. Describe the protocol from the AnB specification: which of the messages can the client C decrypt, which keys does C ever learn? How does the protocol prevent illegitimate access? Note: this question needs to be answered for the **full version** `PKINIT.AnB`!
2. Explain the attack that is found by OFMC, find a fix for it, and verify the fixed protocol with OFMC. Hint: you do not need to introduce new cryptographic constructs, just somewhere a message has too few information. For this exercise, you must not change initial knowledge and goals.
3. Design a variant of PKInit where the key $Ktemp$ is not generated by ath but is obtained from a Diffie-Hellman key-exchange. Verify your variant with OFMC. For this, you need to add the public Diffie-Hellman group to the initial knowledge of all participants and adapt the goals.

Exercise 2: AMP Consider the example `AMP.AnB` on Campusnet.

1. Describe and explain the protocol in the following regards:
   - What security relationship do the parties have initially, according to the initial knowledge?
   - What new security relationship does the protocol (try to) establish?
   - How does the protocol (try to) achieve this?
2. Analyze AMP with OFMC and explain the attack: what does the intruder do, what went wrong?
3. Suggest a fix for the protocol and verify the fixed version for OFMC (again with 2 sessions). Important: the fix must not change the initial knowledge nor modify the goals.
4. Note that the party `s` is a fixed *honest* (*trustworthy*) server. Let us replace `s` by `S`, i.e., a normal role that can be instantiated by the intruder.
   - Why does even the fixed protocol have an attack?
   - Is this new version fixable, i.e., can there be *any* protocol with the same initial knowledge and the same goals that is secure?