Protocol exercises: the Scyther Tool

C.J.F. Cremers, November 2013

The Scyther tool can be downloaded from:

`http://users.ox.ac.uk/~coml0529/scyther/`

For the exercises, we will work from some example input files which are available from the website:

```
protocol0.spdl
protocol0symm.spdl
protocol1.spdl
protocol2.spdl
```

Please copy them into your own directory or desktop before you start the exercises.

Exercise 1
Once you have started the Scyther tool, close the 'about' window. Go to 'File' and select 'Open file'. Here, choose the following file:

`protocol0symm.spdl`

a) Verify the security claims in the protocol using Scyther.

b) Explain the results.

Exercise 2
Once you have started the Scyther tool, close the 'about' window. Go to 'File' and select 'Open file'. Here, choose the following file:

`protocol0.spdl`

a) Verify the security claims in the protocol using Scyther.

b) Explain the difference between the two claims by using the attack you find.

Exercise 3
Now open the following file in Scyther:

`protocol1.spdl`

a) Verify the security claims in the protocol using Scyther. You find several attacks. Explain the attacks: why is the property violated in each case?

b) Copy the protocol file to your own directory, and call it `protocol1fixed.spdl`. In the file, make sure you change 'protocol1' to 'protocol1fixed'. Improve the protocol such that the property now holds, and use Scyther to show that your improved protocol indeed meets the requirements.

   Hint: Examine the first message: $\{\mathbf{R}, ni\}pk(R)$ or compare the protocol to the fixed Needham-Schroeder protocol shown in the lecture.

Exercise 4
Open the following protocol file:

```
protocol2.spdl
```

This protocol contains a rather large messages and contains many random numbers (nonces) and hash functions. Not all of these elements are necessary to guarantee the correctness of the protocol.

a) Suggest five efficiency improvements (in terms of message size or complexity) for the protocol, and motivate your choice. Test each suggested improvement using Scyther. If any of your suggestions fails, explain why.

Exercise 5
Here we recall the informal description of the Yahalom protocol. The aim of this protocol is to exchange a new fresh symmetric key between two principals with the help of a server.

1. $I \longrightarrow R \ : I, N_I$
2. $R \longrightarrow S \ : R, \{\ I, N_I, N_R\ \}_{k(R,S)}$
3. $S \longrightarrow I \ : \{\ R, \mathit{sessionkey}, N_I, N_R\ \}_{k(I,S)}, \{\ I, \mathit{sessionkey}\ \}_{k(R,S)}$
4. $I \longrightarrow R \ : \{\ I, \mathit{sessionkey}\ \}_{k(R,S)}, \{\ N_R\ \}_{\mathit{sessionkey}}$

Note that agents must be able to decrypt messages they receive. If they cannot, such messages are often called "tickets" and are captured in variables of type `Ticket` (a predefined type in Scyther). For session keys, introduce a user-defined type in Scyther, which is globally declared by `usertype SessionKey;`.

a) Model the protocol including secrecy and authentication claims in Scyther.

b) Check the protocol claims with the tool and interpret the results you get.

Exercise 6
Updating a protocol to a new version, while some users are still using the old one, can lead to problems.

a) Verify the concatenation of the original (broken) protocol from exercise 1, and your fixed version, using Scyther. If there are violations of the security claims, report the attacks and explain them. If it is correct, explain why.

b) In general it is safe to label the protocols using unique strings. Consider the protocol from exercise 1. Add the following declaration to the start of the protocol file:

```
usertype String;
const Version1: String;
const Version2: String;
```

These global constants will be used to represent version strings in your protocol. For the protocols, add the version strings like this:

```
    send_1(I,R, { ... }pk(R) );
```

becomes:

```
    send_1(I,R, Version1, { ... }pk(R) );
```

Also add the declarations to your protocol (`protocol1fixed.spdl`). Modify the messages also to include version strings, but use `Version2`.

Concatenate the two input files. You can do this by typing on the command-line:

```
cat protocol1.spdl protocol1fixed.spdl > both.spdl
```

Open the resulting file `both.spdl` in Scyther and verify it. Explain the result.

c) If you found an attack in the previous exercise, apply the labels in a different way such that the concatenation is correct. Explain why your new method does work.