DTU Informatics 02239 Data Security
Protocol Security Lab

Hound-out: September 14, 2022
Hand-in: October 12, 2022 **before noon**

Group hand-in: form groups of 3 to 4 participants
on DTU Learn (self-enrollment)

The lab exercises use the protocol analyzer OFMC 2022 that is found on DTU Learn. The distribution includes executables for Windows, Mac and Linux.[1] For the exercises please use OFMC with the following command line:

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

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

There is also a GUI for OFMC written by a former DTU student:

```
https://github.com/ulfur88/OFMC-GUI/releases/tag/v1.2.1
```

We have observed that this has trouble on some machines with the latest Java; should work best with JDK11.

Both of the following two exercises are real-world protocols (with minor modifications), both protocols had received a thorough security analysis, and both had attacks that the analysts missed. Have fun finding and fixing them!

Exercise 1: Selfie.AnB This protocol, which is on the DTU Learn page of the course, has been designed as a key update protocol: two parties already share a secret key and they want to update it for a new shared key. The shared secret key they have before the protocol is `exp(exp(g,secretk(A)),secretk(B))` where `secretk(A)` and `secretk(B)` are long-term secret keys of `A` and `B`, respectively. (Note that `A` does not know the long-term secret of `B`.) Moreover, `kdf` is a key-derivation function, to build a new key from given arguments, and `mac` is a keyed hash function (Message Authentication Code). As a cryptographic blackbox it suffices to say that from the function result, one cannot obtain the arguments anymore, but everybody who knows the arguments can apply the function.

1. Try to describe and explain the protocol: explain the shape of the initial key, what the new key is, what the purpose of the nonces and MACs are, and why the goals are meaningful.

2. OFMC returns an attack on `Selfie.AnB`. Describe the attack: what happens, what is each agent "thinking" what happens, why does this violate the goals? Essentially: what went wrong here?

3. Fix the protocol and verify the fixed version for two sessions with OFMC.

[1] For compiling the sources yourself, you need the Glasgow Haskell Compiler.

4. Suppose that after the agents have executed the protocol, the intruder would find out `secretk(A)`. Explain why this would break the secrecy goal of the protocol. Is it possible to modify the protocol so that secrecy would still hold as long as `secretk(A)` is only discovered *after* the execution of the protocol? Note that you cannot check this in AnB/OFMC because of the "after" restriction.

Exercise 2: Calling Home

1. Consider the example `call-home.AnB` on DTU Learn. Try to describe the protocol in your own words—what does it try to achieve, how does it work?

2. Analyze the protocol with OFMC and explain the attack: what does the intruder do, what went wrong?

3. How can one fix the protocol by only changing messages in the Action part? Hint: only make changes to the message from `home` to `B`.

4. After fixing the protocol, suppose now $pw(A, s)$ is *not* a strong cryptographic key, but a poorly chosen password. For that matter, uncomment the currently commented-out goal

   ```
   pw(A,home) guessable secret between A,home
   ```

   Explain the attack that OFMC finds on this goal and explain what is the problem.

5. Let us consider the password to be strong again (comment out the guessing goal again). Note that the party `home` is a fixed *honest* (*trustworthy*) server. Let us replace `home` by `Home`, i.e., a normal role that can be instantiated by the intruder. Why does the protocol have an attack then?