Getting Started

Prof. David Basin
Information Security Lab — Sofia Giampietro
October 2022 — Xenia Hofmeier

The lab module 3 will use the Tamarin prover. The next sections contain instructions and hint on installing Tamarin on your machine. Before starting with the project, check, that Tamarin is working correctly, see the instructions below.

Installing Tamarin on your machine

- Instructions to install and use Tamarin can be found in the manual: `https://tamarin-prover.github.io/manual/`
- If you use Windows as your main operating system, you will need to either use a VM or install Tamarin via the Linux for Windows Subsystem: `https://docs.microsoft.com/en-us/windows/wsl/install`. (some common issues with installing Tamarin on WSL are listed below.)
- If you compile Tamarin directly or use Homebrew, be sure to check the resulting binary is on your system `$PATH`.
- We tested the installation using Homebrew on Apple's M1 chips but not on M2.

Troubble Shooting Tamarin on WSL

- Tamarin only runs on WSL2. Check your version with `wsl -l -v`. You can change the version to WSL2 with `wsl --set-version <distro name> 2`.
- If you manually install the Haskell Stack, you could encounter problems regarding `zlib`. In that case, install `libghc-zlib-dev` with your package manager.
- If you manually install Maude, make sure that you install Maude 2.7.1. Installing Maude with your package manager might result in an earlier version. Also, Tamarin does not support newer versions, such as 3.2.
- After manually installing made, check that it works by calling `maude.linux64`. Install missing packages with your package manager.
- Make sure to add Maude to your path.

Checking Tamarin is working correctly

Once you have installed the tool, create a file `sig.spthy` with the code below. Note that copy pasting from the pdf might introduce spaces that make the code incorrect. You can also download the file from Moodle:
`https://moodle-app2.let.ethz.ch/mod/resource/view.php?id=816523`

```
theory Sig

begin

builtins: signing

rule LtkGen://PKI
  [ Fr(~ltk) ]
  -->
```

```
  [ !Ltk($A, ~ltk), !Pk($A, pk(~ltk)), Out(pk(~ltk)) ]

//protocol rules
rule Send_Signature:
  [ Fr(~n), !Ltk($A, ltkA) ]
  --[ Send($A, ~n) ]->
  [ Out(<~n, sign{~n}ltkA>) ]


rule Recv_Signature:
  [ !Pk($A, pkA), In(<n, sig>) ]
  --[ Recv($A, n), Eq( verify(sig, n, pkA), true ) ]->
  [ ]

restriction equal://needed for signature verification
  "All a b #i. Eq(a, b)@i ==> a = b"

lemma executable://sanity check
  exists-trace "Ex A n #i #j. Send(A, n)@i & Recv(A,n)@j"

lemma signature_sent_by_agent://property to be verified
  "All A n #i. Recv(A, n)@i ==> Ex #j. Send(A, n)@j"

end
```

Run `tamarin-prover --prove sig.spthy` and you should get the outcome:

```
  [...]
==============================================================================
summary of summaries:

analyzed: sig.spthy

  executable (exists-trace): verified (6 steps)
  signature_sent_by_agent (all-traces): verified (5 steps)

==============================================================================
```

You can also inspect the file and results in interactive mode by running:
`tamarin-prover interactive .`
(Note the full stop!)