Protocol Design Notes

Applied Information Security — IT University of Copenhagen

October 4, 2020

A protocol defines a sequence of *steps* between several *actors* in a system. Actors are the principals in the system, e.g., users, servers, desktops, laptops, card readers, tokens, etc. In order for these actors to interact they can: *send messages* or *run local programs*. This document describes the syntax and provides guidelines for designing protocols. The document concludes with a full-fledge example and some remarks.

1 Protocol Design Syntax

Every step in the protocol must be explicitly enumerated.

```
1. step_1
2. step_2
3. step_3
   ...
```

The enumeration denotes the execution order (in this case, `step_1`, `step_2`, `step_3`). Each step must be defined in either of these two forms:

- `Sender -> Receiver: message`
  This represents a step where an actor `Sender` sends a message to an actor `Receiver`. The content of the message is the value in the variable `message`.
- `Actor: program`
  This represents a step where `Actor` runs the local program `program`. Programs are specified using pseudo-code, for example,
  - **`if`**–**`then`**–**`else`** statements.
  - `Sender -> Receiver: message` (see above).
  - Variable assignments `x := v`; we can add conditions on `v`, for instance, *with* `v = 3` or *with* `v > 42`.
  - An English statement indicating some effect, e.g., *the user is authenticated*, or *show a red light*.

– Simply a value, e.g., a string (*"Enter password"*).

This is not an exhaustive list. Other statements may be used as long as no ambiguity arises in their meaning.

If the program consists of more than one statement, you can separate them with line breaks and indent them at the same level. In this case, you do not need to enumerate the steps in the program. For example,

```
1. A: x := 1
      y := 2
      A -> B: x+y
```

2 Definitions and Enrollment

Definitions It is recommended to begin the definition of a protocol by explicitly describing:

- The actors in the system.
- Variables and values (information) that each device stores.
- The functions (computation) that each device can perform.
- Assumptions on communication channels.
- ...

This helps clarifying, for instance, whether certain value was stored at enrollment in the device, or it needs to be sent by another actor.

Enrollment It is also recommended to describe the enrollment process. This information may help clarify why certain information is only accessible to certain actors, and increase our confidence on the validity of the information.

3 ITU Room Access Authentication Protocol — Gym

Here we show an example to illustrate how to apply the guidelines in the previous sections. We model the authentication protocol of the room access authentication system installed at ITU. Disclaimer: this is not based on the real implementation of the authentication system. Figure **??** shows the complete definition of the protocol.

```
# ITU Room Access Authentication Protocol

## Definitions and Enrollment

1. Consider a user (U), a card reader (R) and an authentication system
   (S).

2. At enrollment, U is given a plastic card containing her user id
   (uid). Also, U chooses a pin that is stored in S.

3. S uses the function db_pins(uid) to retreive the pin of the
   user with user id uid.

4. If U decides to pay for the gym, then the tuple <uid,GYM> is added
   to the room access dataset (RoomAccess).

5. Card readers include a hard-coded variable (room) indicating the
   room they are installed in.

## Authentication Protocol

1. U -> R: uid      // by showing the card nearby the card reader
2. R -> S: <uid, room>
3. S: if <uid,room> ∈ RoomAccess and room = GYM
      then S -> R: res with res = pin_required
      else S -> R: res with res = not_registered
4. R: if res = pin_required
      then R -> U: Show blinking orange light      // meaning "enter pin"
      else R -> U: Show red light
5. U -> R: upin
6. R -> S: upin
7. S: if upin = db_pins(uid)
      then S -> R: Show green light and open door
      else S -> R: Show red light
```

Figure 1: ITU Room Access Authentication Protocol

Remarks

As you might have noticed, we have made some somehow arbitrary choices in defining this protocol. This was done on purpose, to illustrate that the flexibility of the protocol language.

- We have modeled explicitly the database RoomAccess whereas we have implicitly model a pins database. In doing so, we can use standard set

theoretic operations (set inclusion $\in$, union $\cup$, intersection $\cap$, ...) to handle access to the RoomAccess database. On the contrary, for the pin database, we simply define one function (`db_pin(uid)`), which is the only interface to the database. This choice requires explicitly describing this function before defining the protocol. Both options are equally valid.

- The intermediate values (`pin_required` and `not_registered`) and variables (`res` and `upin`) are not mentioned in the definition and enrollment section. This is because these values and variables are generated on-the-fly as the protocol is executed, therefore it is clear who provides them, where they are stored, and who has access to them.
- We have not mentioned the color code interface from the card reader in the definitions section. This is because is not very relevant for the correctness protocol, it is just an effect of certain operations. That said, it is not incorrect to have defined this interface.
- We have added extra explanations as comments on the protocol. These are unnecessary, but lacking a formal description might clarify the interaction between the user and the card reader. Alternatively, one might write a small description of the protocol before defining it.

We emphasize that you should make choices that prime clarity and aim to decrease ambiguities and enhance readability.