Project Report: Formal Methods for Information Security

Sabina Fischlin & Wanja Chresta

May 30, 2018

1 PACE Protocol

In this first part we describe our results regarding the modelling of the PACE protocol. For each of the theories in this section we have modelled the setup of the symmetric keys by using a fresh value, which we have used as a parameter for the `!LTK` fact. The actors have two separate (uni-directional) symmetric shared keys $K(A, B)$ and $k(B, A)$ (in the theories denoted by $ltkIR$ and $ltkRI$) in their initial state. As we have seen in the lectures we have specified a reveal rule which outputs the key and reveals both A and B.
To model a MAC function we have added a user-defined function $mac/2$.

Additionally we have added a reasonable restriction that the long-term keys must be unique in order to rule out some otherwise possible attacks. As instructed in the project description we have also implemented an executability lemma, which we have proven for each of the theories.

1.1 Assignment 1.1, Theory P1

As given in the assignment description we have modelled the following protocol:

$$\begin{aligned} A \rightarrow B &\;:\; x \\ A \leftarrow B &\;:\; [x]_{k(A,B)} \end{aligned}$$

We have added *attacker_can_be_I* and *attacker_can_be_R* to the immutable lemmas to ensure (and prove) that the intruder can act as a regular protocol participant.

Results

We have proven *non-injective* and *injective agreement* for the initiator. Obviously, for the responder they have been disproven. We have also been able to verify the additional lemmas regarding the intruder acting as a participant.

1.2 Assignment 1.2, Theory P2

1.2.1 Section (a), Theory P2a

In the first part we interleave two instances of the P1 protocol with each other as described in the assignment in the following way:

$$A \rightarrow B \;:\; x$$

$$\begin{aligned} A \leftarrow B &\ :\ y \\ A \rightarrow B &\ :\ [y]_{k(A,B)} \\ A \leftarrow B &\ :\ [x]_{k(B,A)} \end{aligned}$$

Results

We have found attacks on both the *non-injective* and the *injective agreement* for the initiator and responder (described in the next section).

1.2.2 Section (b), Theory P2b

In the attacks in section (a) the intruder was able to inject values for x and y and therefore authentication was violated for both initiator and responder in different ways. By injecting a value for y, the initiator A was able to complete the run thinking it was talking to B, but never having done so. In fact B was never in the role of the responder, but a second initiator:

$$\begin{aligned} \leftarrow B(I) &\ :\ y(dropped) \\ A(I) \rightarrow B(I) &\ :\ x \\ A(I) \leftarrow &\ :\ y'(injected) \\ A(I) \rightarrow &\ :\ [y']_{k(A,B)}(dropped) \\ A(I) \leftarrow B(I) &\ :\ [x']_{k(B,A)} \end{aligned}$$

A here commits to the value y', which B never even received.

In the attack which violates the *non-injective agreement* for the responder, the intruder can make the responder commit to the injected value x' in the following way:

$$\begin{aligned} \leftarrow A(I) &\ :\ x(dropped) \\ B(R) \leftarrow &\ :\ x'(injected) \\ B(R) \rightarrow A(I) &\ :\ y \\ B(R) \leftarrow A(I) &\ :\ [y]_{k(A,B)} \\ B(R) \rightarrow &\ :\ [x']_{k(B,A)}(dropped) \end{aligned}$$

In order to prevent these attacks we have included both nonces in the maced messages and have also added tags:

$$\begin{aligned} A \rightarrow B &\ :\ x \\ A \leftarrow B &\ :\ y \\ A \rightarrow B &\ :\ ["I", x, y]_{k(A,B)} \\ A \leftarrow B &\ :\ ["R", x, y]_{k(B,A)} \end{aligned}$$

Results

With this modification we have been able to prove *non-injective* and *injective agreement* for both participants on nonces x and y.

1.3 Assignment 1.3, Theory P3

1.3.1 Section (a), Theory P3a

As instructed in the assignment description we have added the user-defined function *kdf*/3 and have modelled the protocol in the following way:

$$\begin{aligned} A \rightarrow B &\ :\ x \\ A \leftarrow B &\ :\ y \\ A \rightarrow B &\ :\ [\text{"}I\text{"}, y]_{sIR} \\ A \leftarrow B &\ :\ [\text{"}R\text{"}, x]_{sIR} \end{aligned}$$

where sIR denotes the session key derived from $K(A, B)$ and the nonces x and y. Note that we have kept the tags in the maced messages, but each participant only macs its own nonce.

Results

We have proven *non-injective* and *injective agreement* for both participants on x, y and sIR as well as *secrecy* for this model (however not *perfect-forward-secrecy*).

1.3.2 Section (b)

Unlike in the P2 protocol, information about both nonces is part of the session key sIR. In other words the derivation of the session key sIR directly relies on both parties agreeing on the same nonces (including their own). Therefore including the role's own nonce in the maced message is no longer necessary. However the tags are still necessary (a reflection attack is otherwise possible). (It might even be possible to remove x from the maced message and still reach an agreement, however we did not check this.)

1.4 Assignment 1.4, Theory P4

Again we have used the function *kdf*/3 and have made use of the builtins *hashing* and *symmetric-encryption*. We have modelled the protocol below (as before sIR stands for the session key, this time derived from the nonces s, x and y):

$$\begin{aligned} A \rightarrow B &\ :\ x, \{s\}_{h(k(A,B))} \\ A \leftarrow B &\ :\ y \\ A \rightarrow B &\ :\ [\text{"}I\text{"}, y]_{sIR} \\ A \leftarrow B &\ :\ [\text{"}R\text{"}, x]_{sIR} \end{aligned}$$

Results

We have been able to prove the same lemmas as for the protocol P3a.

1.5 Assignment 1.5, Theory P5

For this part of the PACE protocol we have added the user-defined function *map*/2 to derive g. We have also made use of the *diffie-hellman* builtin.

1.5.1 Section (a)

In this section we have used almost the same model as in P4 with the exception of sIR being derived in a different way and the addition of the public domain parameter p being appended in plaintext to the first message, i.e.:

$$\begin{array}{rcl} A \rightarrow B & : & x, \{s\}_{h(k(A,B))}, p \\ A \leftarrow B & : & y \\ A \rightarrow B & : & [\text{"}I\text{"}, y]_{sIR} \\ A \leftarrow B & : & [\text{"}R\text{"}, x]_{sIR} \end{array}$$

Perfect-forward-secrecy could not be proven for this protocol. This is due to the fact that the reveal of an agent's long-term key will make it possible to derive s and therefore the generator g. Since the nonces x and y have been made public, the session key sIR is no longer secret.

1.5.2 Section (b)

In order to prove *perfect-forward-secrecy* we have adjusted the model in the following way:

$$\begin{array}{rcl} A \rightarrow B & : & g^x, \{s\}_{h(k(A,B))}, p \\ A \leftarrow B & : & g^y \\ A \rightarrow B & : & [\text{"}I\text{"}, g^y]_{sIR} \\ A \leftarrow B & : & [\text{"}R\text{"}, g^x]_{sIR} \end{array}$$

Perfect-forward-secrecy now holds since x and y are never sent and therefore the generator g being made public has no effect on the secrecy of past session keys.

Also all other given lemmas (i.e. *non-injective* and *injective agreement* for both participants on sIR and the half-keys g^x and g^y, and *secrecy* including *perfect-forward-secrecy*) have been proven for this protocol.
(Note that the file *5ab.spthy* only includes this adjusted protocol)

1.5.3 Section (c)

The protocol relies on the secrecy of g since g^x and g^y are sent as plaintext. If g were known this could result in a *Man in the Middle attack*, similar to the active attack on the original Diffie-Hellman protocol. By keeping g secret we forego the necessity to authenticate g^x and g^y.

However, as mentioned above, if g is later revealed we are still guaranteed *perfect-forward-secrecy* since x and y are never sent. (Note that the secrecy is trivially a must for the protocol described in section (a).)

1.5.4 Section (d)

As instructed we have removed the tags in the third and fourth message resulting in the protocol:

$$\begin{array}{rcl} A \rightarrow B & : & g^x, \{s\}_{h(K(A,B))}, p \\ A \leftarrow B & : & g^y \\ A \rightarrow B & : & [g^y]_{sIR} \\ A \leftarrow B & : & [g^x]_{sIR} \end{array}$$

Results

As expected the modification resulted in a reflection attack (included as *P5d-proof_prefix.spthy*). As suggested in the assignment, we have added the restriction *NotEq_MustHold*, after which all the lemmas could again be proven.

2 Off-the-Record Messaging Protocol

In this section we describe our results regarding the *Off-the-Record Messaging Protocol* as described in the provided reference [1].

We have made use of the builtins *asymmetric-encryption*, *signing* and *diffie-hellman* in order to realise the requirements. Registration of the public key and its reveal have been modelled analogously to what we have seen in the lectures. As before we have also implemented and proven our own executability lemma.

2.1 Assignment 2.1, Theory OTR1

We have modelled the protocol as described in Section 2.1 of [1]:

$$\begin{aligned} A \rightarrow B &\quad:\quad \{g^x\}_{ltkA}, pk(ltkA) \\ A \leftarrow B &\quad:\quad \{g^y\}_{ltkB}, pk(ltkB) \end{aligned}$$

where *ltkA* is the long-term (secret) key of A and *pk*(*ltkA*) its corresponding public key. x and y are secret fresh values and g is a not necessarily fresh public value (as in the Diffie-Hellman protocol). This corresponds directly to the protocol specified in [1].

2.2 Assignment 2.2, Theory OTR2

In the theory OTR2 we have specified the lemma *findAttack*, with which we were able to find (and prove) the specified attack trace from Section 3.1 of [1]. (We have also added some restrictions in order to reach this trace.)

2.3 Assignment 2.3, Theory OTR3

As instructed we model the following improvement of OTR1 from Section 3.1 in [1]:

$$\begin{aligned} A \rightarrow B &\quad:\quad g^x, \{g^x, B\}_{ltkA}, pk(ltkA) \\ A \leftarrow B &\quad:\quad g^y, \{g^y, A\}_{ltkB}, pk(ltkB) \end{aligned}$$

Analysis

We first look at the point of view of the **Initiator**. We were able to find the following attack violating the *injective agreement.* In this attack an actor can be tricked into taking the responder's role while he thinks he is in the role *I*: First A wants to talk to B in the role of the initiator:

$$A(I) \rightarrow \quad:\quad g^x, \{g^x, B\}_{ltkA}, pk(ltkA)$$

Next B wants to talk to A, also in the role of the initiator:

$$\leftarrow B(I) \quad : \quad g^y, \{g^y, A\}_{ltkB}, pk(ltkB)$$

Then A receives the initiating request from B and interprets it as its response (since the two are indistinguishable):

$$A(I) \leftarrow \quad : \quad g^y, \{g^y, A\}_{ltkB}, pk(ltkB)$$

A finishes the run thinking it has finished key-establishment with B (and now has a channel with B), but B was never in the role of the responder. This could even happen by accident if both A and B want to initiate a connection at the same time and send their first messages before the message of the other participant has arrived.
We have selected this attack in the proof file.

We now look at the point of view of the **Responder**. The problem in this case is that for the responder, injectivity does not hold trivially: if both the initiator and the responder want to agree on $K = g^{xy}$, then the initiator has to receive g^y before he can commit to K. However, this can only happen after the responder has committed to K. The attacker can therefore just block the second message (i.e. the response) and the initiator will never commit to K. This obviously breaks the injective agreement for the responder. This attack trace looks as follows:

$$\begin{array}{rcl} A(I) \rightarrow B(R) & : & g^x, \{g^x, B\}_{ltkA}, pk(ltkA) \\ B(R) \rightarrow & : & g^y, \{g^y, A\}_{ltkB}, pk(ltkB) \end{array}$$

In this case B commits to $K = g^{xy}$, but A can never commit.

2.4 Assignment 2.4, Theory OTR4

We have modelled the improvement of OTR3 as specified in Section 4 of [1]:

$$\begin{array}{rcl} A \rightarrow B & : & g^x \\ A \leftarrow B & : & g^y \\ A \rightarrow B & : & A, \{g^x, g^y\}_{ltkA}, [\text{"0"}, A]_{h(g^{xy})}, pk(ltkA) \\ A \leftarrow B & : & B, \{g^x, g^y\}_{ltkB}, [\text{"1"}, B]_{h(g^{xy})}, pk(ltkB) \end{array}$$

Note that in OTR4 we have added our own user-defined functions $mac/2$ and $hash/1$ as in Part 1.

Analysis

We have looked at this protocol with regards two separate cases: one including the possibility of agents talking to themselves and one where we excluded this possibility by adding a restriction *NotEqMustHold*. The reason behind this is so that we could find more interesting attacks beyond reflection attacks.

For the first case we have been able to break the *non-injective agreement* for both the initiator and the responder (proofs stored as *OTR4-proof.spthy*).

In the latter case, i.e. where we excluded agents talking to themselves, we were able to prove *secrecy*, *PFS* and *injective agreement* for the initiator.

However, we found the same attack as in the first case for the responder (proofs stored as *OTR4-proof-noteq.spthy*). We will now describe the attacks we found in both cases in more detail.

Breaking the non-injective agreement for the initiator when A may be B. We have found the following attack trace:

$$\begin{aligned}
A(I) \rightarrow &\quad : \quad g^x \\
A(I) \leftarrow &\quad : \quad g \\
A(I) \rightarrow &\quad : \quad A, \{g^x, g\}_{ltkA}, [\text{"0"}, A]_{h(g^x)}, pk(ltkA) \\
A(I) \leftarrow &\quad : \quad B, \{g^x, g\}_{ltkA}, [\text{"1"}, A]_{h(g^x)}, pk(ltkA)
\end{aligned}$$

A will in this case commit to $K = g^x$ without any agent ever being in the role of the responder. It should be noted that this also breaks secrecy for K, since $K = g^x$ which is made public (i.e. A has now established a revealed key to talk to itself).

Breaking the non-injective agreement for the receiver (this attack is possible in both cases). We have found the following attack trace where A tries to initiate a connection with C, but B will interpret this as a A initiating a connection with him instead:

$$\begin{aligned}
A(I) \rightarrow &\quad : \quad g^x(meantforC) \\
\rightarrow B(R) &\quad : \quad g^x \\
\leftarrow B(R) &\quad : \quad g^y \\
A(I) \leftarrow &\quad : \quad g^y \\
A(I) \rightarrow &\quad : \quad A, \{g^x, g^y\}_{ltkA}, [\text{"0"}, A]_{h(g^{xy})}, pk(ltkA)(meantforC) \\
\rightarrow B(R) &\quad : \quad A, \{g^x, g^y\}_{ltkA}, [\text{"0"}, A]_{h(g^{xy})}, pk(ltkA) \\
\leftarrow B(R) &\quad : \quad B, \{g^x, g^y\}_{ltkB}, [\text{"1"}, B]_{h(g^{xy})}, pk(ltkB)
\end{aligned}$$

B has committed to $K = g^{xy}$ as a shared secret with A, even though A has never run the protocol with B, which violates the *non-injective agreement* for the responder. This attack is possible, since none of the messages contain their intended recipient.

References

[1] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Secure Off-the-Record Messaging. In *WPES*, pages 81-89, 2005.