Formal Methods for Information Security

Project report

Mathias Woringer
mworinger@student.ethz.ch

Gian-Luca Piras
gpiras@student.ethz.ch

May 31, 2021

1 The PACE protocol

The referenced files are the theory files, each of which contains an executability lemma for the described protocols as well as additional lemmas that map the security properties required in the task.

1.1 A simple challenge-response protocol

The following simple challenge-response protocol was formalized in Tamarin:

$$A \to B : x$$
$$B \to A : [x]_{k(a,b)}$$

The MAC function was realised by an user-defined function with two parameters in Tamarin. The details can be found in theory file `PACE_1.spthy`.

1.2 Mutual authentication

The challenge-response protocol is extended in this subtask in that now it is not just Alice who sends a nonce x to Bob. Bob now also generates and sends a nonce y, which is sent to Alice. The goal of the protocol is to reach an agreement between the two agents A and B with both nonces.

a) Formally, the protocol now looks like this:

$$A \to B : x$$
$$B \to A : y$$
$$A \to B : [y]_{k(B,A)}$$
$$B \to A : [x]_{k(A,B)}$$

For the first revision of the protocol, we implemented two different protocol approaches. Firstly, we designed the protocol as described (i.e. A sends x to B and B sends y to A, etc), which can be found under `PACE_2a.spthy`, and we noticed in the trace that these steps are happening in parallel. We then implemented the protocol sequentially,

i.e. B must receive something from A before sending something back. This variant can be found in the file `PACE_2a_seq.spthy`. And we use this variant to explain the trace shown in the non-injective agreement lemmas.
The problem in the protocol is that, taking into account a Dolev-Yao intruder, the non-injective agreement can be disproved here for both nonces x and y for both Alice and Bob. This is because neither Alice nor Bob can know for sure whether the message received was actually sent by the other or by an intruder. Therefore, in this case, a *man-in-the-middle attack* can take place as shown in the non-injective agreement lemmas in `PACE_2a_seq.spthy`.
Here, the *man-in-the-middle* establishes communication with Alice and Bob. Alice and Bob think they are communicating with each other and send their nonces to the intruder. The attacker can now send the nonce back 1:1 or even send back any nonce. Alice and Bob now assume that they are communicating with each other, as there is no identifier in the exchanged messages. The returned nonce is now used to send the made message. Thus, the protocol can be carried out successfully without Alice or Bob noticing that they have not communicated with each other.

The trace found by tamarin-prover for the non-injective agreement initiator lemma can be described as follows: The intruder is able to duplicate the agent A including its keys $k(A, B)$ and $k(B, A)$ and the protocol looks approximately like this:

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

(*) Actually it is the intruder which sends z to A without using the duplicate agent.

(**) In the trace, the intruder lets B send the message directly to A. We changed the description slightly to describe the *man-in-the-middle attack*.

For the non-injective agreement responder lemma, the trace can be described as follows: The intruder intercepts the nonce x from A and sends its own nonce z to B. Afterwards B communicates directly wit A, and when B receives back its nonce y MACed with the correct key, it concludes (wrongly) that both nonces y and z are authenticated.

b) We have solved the problem in protocol 2(a) by adding the identifiers 'roleA' and 'roleB' to the participant's nonces on the MACed message. Thus, the property of the injective agreement now holds for both.

$$A \to B : x$$
$$B \to A : y$$
$$A \to B : [\langle' roleA', x, y\rangle]_{k(b,a)}$$
$$B \to A : [\langle' roleB', x, y\rangle]_{k(a,b)}$$

As before we have created a parallel protocol (see theory file `PACE_2b.spthy`) and a parallel protocol (see theory file `PACE_2b_seq.spthy`).

1.3 Introducing a session key

a) With the second refinement we can provide and hold mutual injective agreement on the nonces x and y and therefore on $\mathsf{Kab} = \mathsf{kdf}(\mathsf{k}(\mathsf{A},\mathsf{B}),\mathsf{x},\mathsf{y})$. The refined protocol is defined like following:

$$A \to B : x$$
$$B \to A : y$$
$$A \to B : [\langle' roleA', y\rangle]_{kdf(k(A,B),x,y)}$$
$$B \to A : [\langle' roleB', x\rangle]_{kdf(k(A,B),x,y)}$$

b) In contrast to protocol `PACE_2b`, the two nonces in `PACE_3a` are not MACed, but instead used to generate Kab. Kab has the property that it cannot be manipulated or faked by the intruder. This ensures that the MACed message was only sent by a participant with the correct role. In this way, protocol `PACE_3a.spthy` ensures consistency across both nonces and roles.

1.4 Replace the password by a nonce

The refined protocol is very similar to the previous protocol. Only the low-entropy key $\mathsf{k}(\mathsf{A},\mathsf{B})$ is replaced by high-entropy nonce s.

$$A \to B : x, \{|s|\}_{h(k(A,B)}$$
$$B \to A : y$$
$$A \to B : [\langle' roleA', y\rangle]_{kdf(s,x,y)}$$
$$B \to A : [\langle' roleB', x\rangle]_{kdf(s,x,y)}$$

Compare with `PACE_4.spthy` our parallel and `PACE_4_seq.spthy` our sequential solution. All three security properties from the previous expansion stages are retained

1.5 Introducing Diffie-Hellman: The PACE protocol

a) The nonces x and y were replaced by Diffie-Hellman keys g^x and g^y. The refined protocol looks now like following:

$$
\begin{aligned}
&A \to B : g^x, p, \{|s|\}_{h(k(A,B)} \\
&B \to A : g^y \\
&A \to B : [\langle A, g^y \rangle]_{h(g^{xy})} \\
&B \to A : [\langle B, g^x \rangle]_{h(g^{xy})}
\end{aligned}
$$

b) Perfect forward secrecy is achieved for g^{xy}. See `PACE_5ab.spthy`.

c) The secrecy of the generator g is an essential part of the protocol. This is guaranteed by the encryption with s and the common password $\mathsf{k}(\mathsf{A}, \mathsf{B})$. Otherwise, a Dolev-Yao attacker would be able to infiltrate as a participant who knows the shared secret. In this way, the protocol would continue to run successfully. However, it would not be guaranteed that A would talk to B or vice versa. By defining this generator g as a public value, as in `PACE_5ab_pubg.spthy` one can identify new attacks on the protocol.

d) By removing the role tags, the last two messages become unifiable (see `PACE_5_unify.spthy`). In this way, an attack on the security or authenticity can be made by an attacker using the parameters of A, for example, to send them back to A and thus pretend to be B. This is called a *reflection attack*

$$
\begin{aligned}
&A \to B : g^x, p, \{|s|\}_{h(k(A,B)} \\
&Adv[B] \to A : g^x \\
&A \to B : [g^x]_{h(g^{xx})} \\
&Adv[B] \to A : [g^x]_{h(g^{xx})}
\end{aligned}
$$

By adding the restriction that g^x and g^y must be different, the attack shown above cannot be made (see `PACE_5d.spthy`).

2 The Off-the-Record Messaging Protocol

2.1 Modeling the original OTR Key Exchange

2.2 Authentication Failure

2.3 Improvement

2.4 SIGMA