Formal Methods for Information Security

Project report

Mathias Woringer
mworinger@student.ethz.ch

Gian-Luca Piras
gpiras@student.ethz.ch

May 29, 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:

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

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 A. The goal of the protocol is to reach an agreement between the two roles A and B with both nonces.

a) Formally, the protocol now looks like this:

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

For the first revision of the protocol, we implemented two different protocol approaches. Firstly, we designed the protocol sequentially. This means that Alice must have received something from Bob before Alice can send something to Bob. This variant can be found in the file `PACE_2a_seq.spthy`. Furthermore, we have implemented a parallel variant,

which can be found under `PACE_2a.spthy`. To be able to carry out the attack, the sequential variant is needed.
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 *message replay attack* can take place. In the case of a message being sent from Alice to Bob, the attacker pretends to be Bob or vice versa. The scenario can be represented as follows:

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

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

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

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

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

b) In contrast to protocol `PACE_2`, the two nonces in `PACE_3a` are not MAC'ed, but instead used to generate Kab. Kab has the property that it cannot be manipulated or faked by the intruder. This ensures that the MAC'ed 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 $\mathsf{k(A, B)}$ is replaced by high-entropy password s.

$$\begin{aligned} A \rightarrow B &: x, \{s\}_{h(k(A,B)} \\ B \rightarrow A &: y \\ A \rightarrow B &: [\langle A, y \rangle]_{kdf(s,x,y)} \\ B \rightarrow A &: [\langle B, x \rangle]_{kdf(s,x,y)} \end{aligned}$$

Compare with `PACE_3a.spthy` our parallel and `PACE_3a_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:

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

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

c) The secrecy of the base is an essential part of the protocol. This is guaranteed by the encryption with s and the common password $\mathsf{k(A, B)}$. 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 basis g on a public basis, as in `PACE_5ab_pubg.spthy` one can identify new attacks on the protocol.

d) By removing the 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.

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

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