# Formal Methods for Information Security

**Project report**

Mathias Woringer
mworinger@student.ethz.ch

Gian-Luca Piras
gpiras@student.ethz.ch

May 26, 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. 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 MIM attack, the sequential variant is needed.

a) If we assume that there can be a network attacker who can eavesdrop on all the information exchanged. So we can assume that this protocol is vulnerable to *MIM attacks.* Since neither Alice nor Bob include any information in their messages to authenticate the message, it can be intercepted by an eavesdropper. Alice thinks she is talking to Bob. However, she is talking to Bob. Bob, on the other hand, thinks he is talking to Alice although he is talking to the attacker who is pretending to be Alice. The following attack scenario disproves the non-injective agreement on the nonces x and y for B:

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

b) The problem with the protocol in a) is the fact that neither Alice nor Bob authenticate their messages. Thus, neither Alice nor Bob can actually know whether the message they receive comes from the person they want. In order to eliminate this weakness, an identifier of the respective identity is given to the message. Thus, the property of the injective agreement now holds for both.

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

As before we have created a 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 Kab = kdf(k(A,B),x,y). The refined protocol is defined like following:

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

b) In contrast to protocol P2, the two nonces in P3 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 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 weak key k(A,B) is replaced by s.

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

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:

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

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 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.

d) Solution d

# 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