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:

$$\begin{aligned} &1. \quad A \rightarrow B : x \\ &2. \quad B \rightarrow A : [x]_{k(a,b)} \end{aligned}$$

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 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:

$$\begin{aligned} &1. \quad A \rightarrow B : x \\ &2. \quad B \rightarrow A : y \\ &3. \quad A \rightarrow B : [y]_{k(B,A)} \\ &4. \quad B \rightarrow A : [x]_{k(A,B)} \end{aligned}$$

For the first revision of the protocol, we implemented two different protocol approaches. Firstly, we designed the protocol 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:

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

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.

[*] 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*.

b) We have solved the problem in protocol `PACE_2a` 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.

1. $A \rightarrow B : x$
2. $B \rightarrow A : y$
3. $A \rightarrow B : [\langle 'roleA', x, y \rangle]_{k(b,a)}$
4. $B \rightarrow A : [\langle 'roleB', x, y \rangle]_{k(a,b)}$

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:

1. $A \rightarrow B : x$
2. $B \rightarrow A : y$
3. $A \rightarrow B : [\langle 'roleA', y \rangle]_{kdf(k(A,B),x,y)}$
4. $B \rightarrow A : [\langle 'roleB', x \rangle]_{kdf(k(A,B),x,y)}$

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.

1. $A \rightarrow B : x, \{|s|\}_{h(k(A,B)}$
2. $B \rightarrow A : y$
3. $A \rightarrow B : [\langle 'roleA', y \rangle]_{kdf(s,x,y)}$
4. $B \rightarrow A : [\langle 'roleB', x \rangle]_{kdf(s,x,y)}$

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:

$$\begin{aligned}
&1. \quad A \to B : g^x, p, \{|s|\}_{h(k(A,B))}\\
&2. \quad B \to A : g^y\\
&3. \quad A \to B : [\langle A, g^y \rangle]_{h(g^{xy})}\\
&4. \quad B \to A : [\langle B, g^x \rangle]_{h(g^{xy})}
\end{aligned}$$

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 $\mathsf{k(A,B)}$. 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 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

In the given paper, the key-exchange protocol is described as follows:

> In order to add authentication to the basic DH protocol, OTR uses public keys and digital signatures. Specifically, each party A in the OTR network has a pair of secret/public keys (sk_A,vk_A) for a digital signature scheme (implemented using either DSA or RSA signatures).
>
> In a signature scheme, the secret key sk_A is used to create valid signatures by the owner of the key such that no other party can create valid signatures. At the same time, anyone, using the public key vk_A, can verify the validity of any given signature. Note that the public key is associated to the identity of a specific party. OTR adopts a simple and non-hierarchical approach to the distribution of public keys, where each party stores the public keys of the users he communicates with. When first entered users are prompted to verify validity of the public key via fingerprint recognition, much like in SSH [26].
>
> The OTR authenticated key-exchange phase requires that each party signs its Diffie-Hellman value. The public key is sent with the first message:
>
> $$A \to B : Sign_{sk_A(g^x)}, vk_A$$
>
> $$A \leftarrow B : Sign_{sk_b(g^y)}, vk_B$$
>
> If the public key vk_A is already stored by B and it is associated with the identity of A , this assures B that g^x comes from A and vice versa (in the absence of pre-stored public keys the protocol could use PK certificates). After the verification of the signatures, both parties compute their shared secret value g^{xy} and erase the DH exponents.

In the protocol theory, we use the builtins "signing" and "diffie-hellman", which include the necessary functions and equations for the protocol. We then use the Key-generation rule to generate key-pairs sk_x (signing key) and pk_x (verification key). Then we use Initialisation rules to create the agents A and B so each agent knows its partner's pk (A know pk_B and B knows pk_A) from beginning on (as specified in the above description).
We then implement the protocol as follows:

$$1. \quad A \to B : g^x, sign\{g^x\}_{sk(A)}, pk(A)$$

$$2. \quad B \to A : g^y, sign\{g^y\}_{sk(B)}, pk(B)$$

Where g is a fixed constant to represent the generator, x and y are fresh values for the ephemeral keys and $sk(x)$ and $pk(x)$ are the key-pairs generated for each agent.
In order to model the executable lemma, we add *Finish* statements at the end of the protocol for each agent; we also make sure that a message received by A corresponds to a message sent by B, that only one agent A and one agent B are created and finally that agent A and agent B are different.
See `OTR1.spthy`

2.2 Authentication Failure

The *man-in-the-middle-attack* presented in the paper looks as follows:

$$A \to E[B] : g^x, Sign_{sk_A}(g^x), vk_A$$

$$E \to B : g^x, Sign_{sk_E}(g^x), vk_E$$
$$E \leftarrow B : g^y, Sign_{sk_B}(g^y), vk_B$$
$$A \leftarrow E[B] : g^y, Sign_{sk_B}(g_y), vk_B$$

The adversary E impersonates B for A (hence the notation $E[B]$ for the communication between A and E) and then forwards g^x to B but signed with E's key. B then replies to E by sending g^y signed by B's key and E forwards this message to A.
We were able to reproduce a similar attack in tamarin by restricting the creation of only one agent A and max. two agents B (one representing the man in the middle E), but this attack is not quite the one described previously. In this attack, because the agent B is only able to send a fresh g^y (where y is fresh), $E[B]$ cannot forward g^x to B, but sends its own g^z to B; in the same manner $E[B]$ cannot forward B's reply to A and so the attack in tamarin looks as follows:

$$A \to E[B] : g^x, sign\{g^x\}_{sk(A)}, pk(A)$$
$$E \to B : g^z, sign\{g^z\}_{sk(E)}, pk(E)$$
$$B \to A : g^y, sign\{g^y\}_{sk(B)}, pk(B)$$

So at the end A and B won't be able to communicate because the resulting key is different for A and B (g^{xy} for A and g^{zy} for B).
See `OTR2.spthy`

2.3 Improvement

We implement the improvement as described in the OTR paper in section 3.1 as follows:

1. $A \to B : g^x, sign\{g^x, B\}_{sk(A)}, pk(A)$
2. $B \to A : g^y, sign\{g^y, A\}_{sk(B)}, pk(B)$

and the analysis of the new protocol with the standard non-injective agreements (both on the initiator as well as on the responder side) proves unsatisfied, i.e. a trace was found for both lemmas. The previous non-injective agreement lemma now succeeds, i.e. no trace was found. It is included in the theory as *NonInjectiveAgreementInitiatorOld.*
As per tamarin-prover, an adversary is able to generate many key-pairs and create many agents A and B, which will simulate the various steps in the protocol, because there is no real link between the agent and its role, e.g. a malicious agent A can simulate being in the responder role and the same is valid for a malicious agent B in the initiator role.
See `OTR3.spthy`.

But by adding the role in the agent's signature, e.g. by replacing agent A's $sign\{g^x, B\}_{sk(A)}$ by $sign\{I, g^x, B\}_{sk(A)}$(and the same for agent B), the non-injective agreement lemmas for both initiator and recipient now succeed. And the secrecy and forward secrecy lemmas also succeed. Unfortunately the injective agreement lemmas for both roles do fail as the specified protocol doesn't prevent an agent A to communicate with two agent B and vice versa.
See `OTR3_B.spthy`.
As an experiment, we modified the signed Diffie-Hellman theory seen in the exercise session to include the authentication lemmas (non-injective and injective) and there too the non injective agreement lemmas succeed but the injective ones fail.
See *ex33_SDH_mod.spthy.*

2.4 SIGMA

In order to model the SIGMA protocol as described in the OTR paper in section 4.1, we add the builtin *hashing* and a function *mac(k, m).* We then model the protocol as follows:

1. $A \rightarrow B : g^x$
2. $B \rightarrow A : g^y$
3. $A \rightarrow B : A, sign\{g^y, g^x\}_{sk(A)}, MAC_{h(g^{xy})}('I', A), pk(A)$
4. $B \rightarrow A : B, sign\{g^x, g^y\}_{sk(B)}, MAC_{h(g^{xy})}('R', B), pk(B)$

The analysis of the protocol shows that the following lemma succeed:

- non-injective agreement initiator,
- injective agreement initiator,
- secrecy,
- forward secrecy.

But the non-injective agreement responder fails. In the trace, one can see that the adversary can fool an honest agent B by creating a malicious agent A which intercepts the messages send by the honest agent A and also takes the responder role to fool the honest agent A.
See `OTR4.spthy`
Contrary to the `OTR3_B` improvements, adding the role in the agent's signature doesn't resolve the issue; there is still a trace found for the non-injective agreement responder lemma. Our guess is that because the first message from agent A to agent B is not authenticated, it will be always possible for an adversary to create some doubt on the identity of the sender of the first message. In `OTR3_B`, the first message was signed by the agent.
See `OTR4_B.spthy`