Formal Methods for Information Security

Computer-aided verification of the PACE and OTR protocols

Raphael Eikenberg
reikenberg@ethz.ch

Robertas Maleckas
rmaleckas@ethz.ch

May 16, 2020

Contents

1 PACE Protocol

1.1 A simple challenge-response protocol

We have defined the theory `P1` as given in the assignment. Specifically, we implemented the protocol as follows.

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

We use a restriction in Tamarin to drop traces where two roles A and B may have set up equal keys for both directions of communication. This way, keys are forced to be uni-directional.

1.2 Mutual authentication

(a) Theory `P2a` can be described as follows.

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

When trying to prove the injective agreement lemma for A, Tamarin finds an attack where an agent a playing role A is able to complete the protocol run without actually talking to a thread playing role B. We will go into an in-depth explanation of this attack in (b).

We note that a symmetric attack is found when trying to prove injective agreement of B.

We also strengthen the key unidirectionality restriction by checking for key equality in both roles at the earliest point when both keys are known to an agent.

(b) The attack we found can be described as follows.

$$\begin{aligned} A \to A': &\quad x && (1.1) \\ A' \to E: &\quad x' && (1.2) \\ E \to A: &\quad y && (1.3) \\ A \to A': &\quad [y]_{k(A',A)} && (1.4) \\ A' \to A: &\quad [x]_{k(A,A')} && (1.5) \end{aligned}$$

When A commits after receiving message 1.4, they cannot be sure that the second nonce is properly agreed upon: the commit is (incorrectly) claimed over both nonces, while the received HMAC only guarantees that the other party agrees on y. This means an attacker can intercept and replace nonces, tricking agents into committing over values different from those of their communication party.

As we see in the attack found by Tamarin, they can even trick an agent playing role A into completing the protocol with the help of another agent also in role A. This is due to the symmetry of our protocol.

We can fix this by tagging the messages and adding *both* nonces to the HMAC. This way, a receiver can be sure that both nonces that are agreed upon are agreed upon with the other party. In addition, we extend the HMAC over the roles to prevent the aforementioned role symmetry attack. Lastly, the tags introduce asymmetry and thus prevent the attacker from tricking a party into completing the protocol with another party (possibly playing the same role!) at a different step.

Theory `P2b` can be described as follows.

$$\begin{aligned} A \to B: &\quad x \\ B \to A: &\quad y \\ A \to B: &\quad [\text{“I”}, \text{“R”}, x, y]_{k(B,A)} \\ B \to A: &\quad [\text{“R”}, \text{“I”}, x, y]_{k(A,B)} \end{aligned}$$

1.3 Introducing a session key

(a) We have extended the theory `P3a` to use session keys and only authenticate the communication partner's nonce as described in the assignment.

$$\begin{aligned} A \to B: &\quad x \\ B \to A: &\quad y \\ A \to B: &\quad [\text{“I”}, \text{“R”}, y]_{Kab} \\ B \to A: &\quad [\text{“R”}, \text{“I”}, x]_{Kab} \end{aligned}$$

(b) Since both nonces are involved in the derivation of the session key, the MAC over the agent's own nonce is redundant, i.e., the MAC still depends on both nonces through the session key.

In `P2`, this does *not* work in the same way, because an agent's own nonce is not part of the key, and hence the nonce is not associated with the unique protocol run.

Mutual agreement on both nonces, executability, and secrecy of the key lemmas all get verified successfully.

1.4 Replace the password by a nonce

We have replaced the long-term key (password) with a nonce generated by role A in theory `P4` as required by the assignment.

Theory `P4` can be described as

$$\begin{aligned} A \to B: &\quad x, [s]_{h(k(A,B))} \\ B \to A: &\quad y \\ A \to B: &\quad [\text{“I”}, \text{“R”}, y]_{Kab} \\ B \to A: &\quad [\text{“R”}, \text{“I”}, x]_{Kab} \end{aligned}$$

where $Kab = \text{kdf}(s, x, y)$.

Tamarin can prove that the same properties as for the previous protocol hold.

1.5 Introducing Diffie-Hellman: The PACE protocol

(a) We have replaced the nonces with Diffie-Hellman half-keys in theory `P5ab` as required by the assignment.

The protocol can now be described as follows.

$$\begin{aligned} A \to B: &\quad g^x, [s]_{h(K(A,B))}, p \\ B \to A: &\quad g^y \\ A \to B: &\quad [\text{“I”}, \text{“R”}, g^y]_{Kab} \\ B \to A: &\quad [\text{“R”}, \text{“I”}, g^x]_{Kab} \end{aligned}$$

(b) We added the perfect-forward-secrecy lemma and verified that Tamarin is able to prove it. The description of the protocol does not change.

(c) The secrecy of g is needed for the authentication between the two parties. If g was publicly known, the attacker could simply run a MITM attack by setting up a session key with either party. We illustrate one side of such attack with an attacker role E (Eve):

$$A \to E: \quad g^x, [s]_{h(k(A,B))}, p \tag{1.6}$$

$$E \to A: \quad g \tag{1.7}$$

$$A \to E: \quad [\text{“I”}, \text{“R”}, g^x]_{Kab} = [\text{“I”}, \text{“R”}, g^x]_{h(g^x)} \tag{1.8}$$

Message 1.6 is intercepted by Eve, so agent b does not receive it. In message 1.7, Eve supplies agent a with the (known) base. This tricks a into deriving the shared Diffie-Hellman secret $s = g^x$ (instead of g^{xy}), which is equal to their half-key sent in plaintext in message 1.6, thus known to the attacker. Message 1.8 is also blocked by Eve. The fourth message of the protocol is omitted for brevity.

As a result of this attack, a shares a session key with the attacker, contrary to its belief of sharing it with the intended party playing role B. Likewise, Eve can complete the above steps with the other role B, mounting a successful MITM attack.

(d) After removing the tags from our protocol, the injective agreement of the initiator no longer holds. Since the initiator expects a challenge right after issuing theirs, the attacker can return the same challenge. This tricks the initiator into answering its own challenge, which the attacker simply mirrors back to them in the final step.

This can illustrated be the following Alice and Bob protocol, which reflects the attack that Tamarin found.

$$\begin{aligned} A \to A: &\quad g^x, [s]_{h(K(A,A))}, p \\ A \to A: &\quad g^y \\ A \to A: &\quad [g^y]_{Kaa} \\ A \to A: &\quad [g^x]_{Kaa} \end{aligned}$$

This reflection attack is not possible with an agent in the responder role (i.e. the responder's injective agreement still holds). This is because b expects a response to their challenge before responding to the one sent by the initiator (attacker). If role B had to answer the challenge first, it would become vulnerable instead (effectively exchanging places with A).

After adding the restriction to only consider traces with different half-keys, all lemmas – including A's injective agreement with B – were provable again. In other words, the reflection attack is no longer discoverable via Tamarin if traces where an agent ends up with equal half-keys are discarded.

Theory `P5d` can be described as follows.

$$\begin{aligned} A \to B: &\quad g^x, [s]_{h(K(A,B))}, p \\ B \to A: &\quad g^y \\ A \to B: &\quad [g^y]_{Kab} \\ B \to A: &\quad [g^x]_{Kab} \end{aligned}$$

2 OTR Protocol

2.1 Modeling the original OTR Key Exchange

We modeled the protocol as follows, assuming $Sign_{sk_A}(m) = (m, signature_{sk_A}(m))$, i.e., both the message and the signature are sent.

$$A \to B: \quad g^x, \{g^x\}_{sk(A)}, pk(A) \tag{2.1}$$
$$B \to A: \quad g^y, \{g^y\}_{sk(B)}, pk(B) \tag{2.2}$$

Here, $\{m\}_k$ denotes only the signature of m using key k as a tag, not containing the actual message.

(2.1) Bob receive's a DH half-key g^x, which could only have been signed by the owner of the related private key $sk(A)$, unless compromised. Therefore, Bob can be sure that the half-key g^x must originate from Alice.

(2.2) As the initiator intending to talk to Bob, Alice will only accept a half-key g^y signed with his private key $sk(B)$, verifying it originated from Bob[1], unless compromised.

Since both parties sign their own half-key, the respective other party can verify its authenticity.

2.2 Authentication Failure

We modeled theory `OTR2` as follows.

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

Di Raimondo et al. describe "an attack in which the attacker Eve interferes between Alice and Bob in a way that both parties end computing the same key but while Alice believes that the peer to the exchange is Bob, Bob believes that the key was exchanged with Eve." This exact behavior can be observed in theory `OTR2` using Tamarin.

[1] We assume identity links to public keys are known by the participants, as "each party stores the public keys of the users he communicates with" [1, §2.1].

The attack can be illustrated as follows.

$$\begin{aligned} A \to E: &\quad g^x, \{g^x\}_{sk(A)}, pk(A) \\ E \to B: &\quad g^x, \{g^x\}_{sk(E)}, pk(E) \\ B \to E: &\quad g^y, \{g^y\}_{sk(B)}, pk(B) \\ E \to A: &\quad g^y, \{g^y\}_{sk(B)}, pk(B) \end{aligned}$$

We restricted the injective agreement lemma to only consider traces with one agent playing each of the roles A and B. However, when considering sources for the value to inject to the agent playing role B, Tamarin chose to produce a trace where the attacker generates and injects a half-key of its own (as opposed to re-signing role A's value), which was slightly different from the attack outlined in the paper. Thus, we further restricted the lemma to only consider traces where the correct *values* are agreed upon, just not necessarily between the right parties. This proved enough to mirror the described attack.

2.3 Improvement

First, we implemented the protocol described in [1] as follows.

$$\begin{aligned} A \to B: &\quad g^x, B, \{g^x, B\}_{sk(A)}, pk(A) \\ B \to A: &\quad g^y, A, \{g^y, A\}_{sk(B)}, pk(B) \end{aligned}$$

Here, A and B in the messages denote the identities of the respective agents.

We have to analyze agreement on values used for authentication (Diffie-Hellman half-keys and the full shared secret) from both points of view. For the following, assume we have an agent a in role A and agent b in role B communicating with each other.

If implemented exactly as described in the paper, Tamarin trivially disproves the lemmas for both injective and non-injective agreement on either the shared secrets or just the other party's half-keys by using multiple agents playing the same role A. When looking at the trace that Tamarin provides, we observe the following behavior.

$$\begin{aligned} A \to E: &\quad g^x, \{g^x, B\}_{sk(A)}, pk(A) \\ A' \to A: &\quad g^y, \{g^y, A\}_{sk(A')}, pk(A') \end{aligned}$$

This does not provide much insight into the protocol; to look for more complex attacks, we introduced signed role labels to prevent Tamarin from abusing multiple A roles as follows:

$$\begin{aligned} A \to B: &\quad g^x, \{\text{“I”}, g^x, B\}_{sk(A)}, pk(A) \\ B \to A: &\quad g^y, \{\text{“R”}, g^y, A\}_{sk(B)}, pk(B) \end{aligned}$$

This protocol is described in theory `OTR3_tagged` in our submission.

In this case, we learned that agent a may only agree non-injectively on b's half-key with b. The reason why a cannot agree on its own half-key and the shared key is that

a cannot correlate the messages to ensure that b's response originated from a's specific initiation (discontinuity is possible). In other words, b may be running with a different value $g^{x'}$ and thus a different shared secret $K_{a,b} = g^{x'y}$.

Moreover, the reason why the agreement cannot be injective is that an attacker may reuse half-keys to trick multiple threads of the same agent into agreeing on the same value.

This would result in non-uniqueness of the commit claim event, thus violating the injectivity property on b's half-key:

$$\begin{aligned} A_3 \to B: &\quad g^x, B, \{g^x, B\}_{sk(A)}, pk(A) \\ B \to A_1: &\quad g^y, A, \{g^y, A\}_{sk(B)}, pk(B) \\ E \to A_2: &\quad g^y, A', \{g^y, A\}_{sk(B)}, pk(B) \end{aligned}$$

Note that here, A_n denotes a thread n of an agent in role A. This means that A_1 and A_2 are two threads of a single agent a playing role A, and thus share the same private key, and identity A.

The protocol identities are handled at agent granularity, which makes this attack possible as either of the threads will accept messages meant for agent a. A "real life" scenario of this might be different processes within a machine sharing the same secrets (e.g. SSH keys).

From the view of role B, we can only agree non-injectively on the half-key of role A. This is an inherent limitation of the protocol rather than an attack: b cannot commit to an agreement with a on the shared secret, because at the time b finishes playing role B, a cannot possibly know b's half-key.

Same as with a's point of view described above, injectivity is not possible as the same half-key from a could be relayed to two (or more) agents playing the roles of B.

We note that one way of achieving injectivity would be to sign both half-keys. This would assure agents the responses are fresh as they would be associated with their own freshly generated half-keys via the shared signature.

2.4 SIGMA

We implemented the protocol SIGMA as described in [1] (SIGMA-R in the original paper [2]).

$$\begin{aligned} A \to B: &\quad g^x \\ B \to A: &\quad g^y \\ A \to B: &\quad g^y, g^x, \{g^y, g^x\}_{sk(A)}, [\text{“0”}, A]_{K_m}, pk(A) \\ B \to A: &\quad g^x, g^y, \{g^x, g^y\}_{sk(B)}, [\text{“1”}, B]_{K_m}, pk(B) \end{aligned}$$

2.4.1 Authentication

We found that injective agreement of an agent a on the shared key with b holds.

In contrast, injective agreement of an agent b on the shared key with a does not hold. This is in line with the original description stating that it "[...] provides defense to the responder's identity against active attacks and to the initiator's only against passive attacks" [2, §5.3].

When a receives b's half-key, and proceeds to reply with the signed half-keys, it cannot be sure who the nonce originated from. At this point a has no way of knowing that it may have originated from b and not the (different) intended partner. Consequently, upon receival of a's signed half-key, b too cannot be certain that a intends to communicate with b and not someone else.

Tamarin successfully finds the following counterexample to the responder's injective agreement lemma illustrating this, where a has a communication partner b' in mind, but is instead passed b's nonce by an attacker intercepting the messages.

$$\begin{aligned}
A \to B' &: \quad g^x \\
E \to B &: \quad g^x \\
B \to E &: \quad g^y \\
E \to A &: \quad g^y \\
A \to E &: \quad g^y, g^x, \{g^y, g^x\}_{sk(A)}, [\text{“0”}, A]_{K_m}, pk(A) \\
E \to B &: \quad g^y, g^x, \{g^y, g^x\}_{sk(A)}, [\text{“0”}, A]_{K_m}, pk(A) \\
B \to E &: \quad g^x, g^y, \{g^x, g^y\}_{sk(B)}, [\text{“1”}, B]_{K_m}, pk(B)
\end{aligned}$$

Without the ability to verify their identity at that point, a signs it. b receives this message and makes a commit claim on the now constructed key and the two identities, which is wrong since a believes they are running the protocol with b'. The claim cannot be moved to after a's attempt of authenticating b since the protocol finishes with a message received by a.

The attack above would not result in a successful run for a as the identity mismatch will be uncovered later in the protocol. However, it is sufficient to break the responder's injective agreement lemma as b finishes their protocol (incorrectly) claiming that a must be running with b.

We believe that this attack is not very practical. Since a signed b's nonce, their intended communication partner will refuse to authenticate a, noticing the mismatch. Thus, even if b is convinced they agree with a on the key, while the secrecy of the key is preserved, a will abort the protocol run after noticing their communication partner is not b'.

2.4.2 Secrecy

We were able to successfully verify the secrecy and perfect forward secrecy properties of the protocol. This comes at no surprise, as SIGMA is based on a basic DH key exchange, and fresh half-keys are exchanged in each protocol run.

Bibliography

[1] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Secure Off-the-Record Messaging. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES '05, page 81–89, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595932283. doi: 10.1145/1102199.1102216. URL `https://doi.org/10.1145/1102199.1102216`.

[2] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 400–425, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45146-4.