Formal Methods for Information Security Project: Formal verification of the PACE and OTR protocols

Authors:
Alexander Khalil Arwadi
Yunxin Sun

June 3, 2022

1 PACE Protocol

1.1 A simple challenge-response protocol

Theory `P1` implements the following protocol:

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

To make sure that k(A,B) is unidirectional, we added a restriction in Tamarin that drops traces where k(A,B) is equal to k(B,A). We modeled the MAC function by defining an equation in Tamarin that takes the message m and a key k and symmetrically encrypts m under key k. We then proved that A injectively agrees with B on nonce x by using lemma `injectiveagreementINITIATOR` since A is the initiator in the protocol.

1.2 Mutual authentication

(a) Theory `P2a` implements the following protocol:

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

To make sure that the keys are unidirectional, we added the same restriction used in `P1`. When trying to prove injective agreement of each role with the other role on both nonces x and y, Tamarin finds attacks to both lemmas. We will go over an in-depth explanation of these attacks in the next part.

(b) Assume A and A' are two agents in role A and E is the attacker. The attack found by Tamarin can be described as follows:

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

This trace includes two attacks. First, as we can see, the two agents in role A do not agree on nonces x and y. In fact, the attacker can intercept and replace nonces, tricking agents into committing to values different than the one chosen by the other party. For example, E sends y' to A instead of y that is chosen by A'. The protocol ends with A committing to y' instead of y and A' committing to x. The second attack here is a reflection attack. As we can see, the protocol runs between two agents in role A instead of one in role A and another in role B. This is due to the symmetry of the protocol.
To fix the first issue, we propose to include both nonces x and y in the HMAC. This way both agents commit to both nonces at the same time. To fix the second issue, we propose to include tags "I" and "R" in the HMAC. The tags will introduce asymmetry in the HMAC and will counter the reflection attack. The mutual agreement on both nonces gets verified successfully.

Theory `P2b` implements the following protocol:

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

1.3 Introducing a session key

(a) Theory `P3a` implements the following protocol:

$$\begin{aligned}
&A \rightarrow B : x\\
&B \rightarrow A : y\\
&A \rightarrow B : ["I", "R", y]_{Kab}\\
&B \rightarrow A : ["R", "I", x]_{Kab}
\end{aligned}$$

Key `Kab` is a session key that is equal to kdf(k(A, B), x, y), meaning that it is the output of a key derivation function where the input is the long-term key and the nonces x and y. Note that we still include the tags in the HMAC to overcome the reflection attack. The mutual agreement on both nonces gets verified successfully as well as the secrecy lemma.

(b) In this protocol, it is enough to include each role's own nonce in the agreement despite that nonce not being maced because the session key depends on both nonces x and y, meaning that the HMAC still indirectly depends on the values of x and y.

1.4 Replace the password by a nonce

Theory `P4` implements the following protocol:

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

Key `Kab` is a session key that is equal to kdf(s, x, y), meaning that it is the output of a key derivation function where the input is a high entropy nonce s generated by A and the nonces x and y. The mutual agreement on both nonces gets verified successfully as well as the secrecy lemma.

1.5 Introducing Diffie-Hellman: The PACE protocol

(a) Theory `P5ab` implements the following protocol:

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

The nonces x and y are replaced by Diffie-Hellman half-keys. Generator g is equal to map(s, p) where p is a public domain parameter. The mutual agreement on both half-keys gets verified successfully as well as the secrecy lemma.

(b) We added the perfect forward secrecy lemma and Tamarin successfully verifies it.

(c) If g was publicly know, then each agent would not need to calculate it using encrypted nonce s. Everyone knows g including the attacker which lets him run a man in the middle attack and set up a session key with either party. We will illustrate the attack in the following, where E is considered the attacker and A is an agent in role A:

$$\begin{aligned}
&A \to E : g^x, [s]_{h(k(A,B))}, p\\
&E \to A : g\\
&A \to E : [\text{"}I\text{"}, \text{"}R\text{"}, g]_{Kab}\\
&E \to A : [\text{"}R\text{"}, \text{"}I\text{"}, g^x]_{Kab}
\end{aligned}$$

Here E intercepts the messages sent by A and won't send them to B. Instead, E sends g to A as a half-key. Therefore the session key `Kab` will be equal to $h(g^x)$ which can be computed by E easily. At the end of the protocol, A shares a session key with E instead of B. Likewise, E implements the same attack on the side of agent B, and therefore successfully mounts a man in the middle attack.

(d) If $h(g^x)$ is equal to $h(g^y)$ and the tags are removed, the following reflection attack on authentication can be mounted:

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

In this attack, E chooses to mirror back the same challenge used by A. E will wait for the encryption of the challenge using the session key and will also mirror this message to A. The protocol successfully terminates between A and E, when in reality A thinks it is communicating with B.

To overcome this attack, we added a restriction on the set of traces which eliminates the ones where $h(g^x)$ is equal to $h(g^y)$. This is shown in theory `P5d` which implements the following protocol:

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

2 The Off-the-Record Messaging Protocol

2.1 Modeling the original OTR Key Exchange

Theory `OTR1` implements the following protocol:

$$\begin{aligned}&A \rightarrow B : g^x, Sign_{skA}(g^x), vkA\\&B \rightarrow A : g^y, Sign_{skB}(g^y), vkB\end{aligned}$$

Our model implements the protocol as described in [1]. Note that we send the Diffie-Hellman half-key with the signature and verification key. When A sends to B its half-key, it signs it with its own secret key so that B can verify the signature with the corresponding verification key. Same thing happens when B sends its half-key to A. After the verification of the signatures, both parties compute their shared secret value g^{xy} and erase the Diffie-Hellman exponents. Finally, the executability lemma gets verified.

2.2 Authentication Failure

Theory `OTR2` implements the following protocol:

$$\begin{aligned}&A \rightarrow B : g^x, Sign_{skA}(g^x), vkA\\&B \rightarrow A : g^y, Sign_{skB}(g^y), vkB\end{aligned}$$

In this part of the project, we needed to demonstrate the authentication failure that was discussed in section 3.1 of [1]. The attack consists of an attacker Eve who interferes between Alice and Bob in a way that both parties end up computing the same key but while Alice believes that the key was exchanged with Bob and Bob believes that the key was exchanged with Eve. To find this specific trace, we restricted the Tamarin to only consider traces where we have a single agent in role A and in role B. Moreover, we restricted the adversary to use values that are agreed upon and not come up with a new half-key. This was enough show that injective agreement doesn't hold because of the reason described in the paper. The attack can be modeled as follows:

$$\begin{aligned} &A \rightarrow E : g^x, Sign_{skA}(g^x), vkA \\ &E \rightarrow B : g^x, Sign_{skE}(g^x), vkE \\ &B \rightarrow E : g^y, Sign_{skB}(g^y), vkB \\ &E \rightarrow A : g^y, Sign_{skB}(g^y), vkB \end{aligned}$$

2.3 Improvement

Theory `OTR3` implements the following protocol:

$$\begin{aligned} &A \rightarrow B : g^x, Sign_{skA}('I', g^x, B), vkA \\ &B \rightarrow A : g^y, Sign_{skB}('R', g^y, A), vkB \end{aligned}$$

Our model implements the protocol as described in [1]. However, in order to prevent reflection attacks that were found by Tamarin, we added the roles 'I' and 'R' in the signatures. Moreover, we restricted Tamarin to consider only traces where we have one agent in role A and in role B. Injective and non-injective agreement on the key gets verified from both point of views of the initiator and the responder. In fact, by adding the identity of the other party in the signature, we are sure that an attacker Eve won't be able to relay the signature from B to A with role E in the signature. This means that the attack that was seen in the previous section cannot be achieved anymore by the attacker.

2.4 SIGMA

Theory `OTR4` implements the following protocol:

$$\begin{aligned} &A \rightarrow B : g^x \\ &B \rightarrow A : g^y \\ &A \rightarrow B :' A', Sign_{skA}(g^y, g^x), ['0',' A']_{Kab}, vkA \\ &B \rightarrow A :' B', Sign_{skB}(g^x, g^y), ['1',' B']_{Kab}, vkB \end{aligned}$$

In this part of the project we tried to prove all standard lemmas. We restricted Tamarin to consider only traces where we have one agent in role A and in role B. We found that injective and non-injective agreement on the key holds with respect to the initiator A. However, they do not hold with respect to the responder B. In fact, the half-keys are sent in the clear and there is no proof showing who is the sender. A man in the middle can clearly manipulate the messages and trick the parties. When A sends its half-key to B, an attacker E can relay the half-key to another party B'. B' will send back its half-key to E who relays it to A. A will sign the half-keys and send it to B, but E will relay it to B'. B' will sign the half-keys and send it to E. At the end of this attack, B will think that they share a key with A, while A doesn't. This attack doesn't work with respect to the initiator A, because the last message of the protocol cannot be created by E in a way that makes A commit. Finally, we were able to verify the secrecy and forward secrecy lemmas for this protocol. This was predictable since the protocol is based on a Diffie-Hellman key exchange with fresh half-keys.

References

[1] R. G. Mario Di Raimondo and H. Krawczyk, “Secure off-the-record messaging,” *WPES*, p. 81–89, 2005.