CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Formal Analysis of Real-World Security Protocols

Prof. Dr. Cas Cremers, Dr. Aleksi Peltonen
Alexander Dax, Niklas Medinger

https://cms.cispa.saarland/farwsp24/

Exercise Sheet 3

due 18.11.2024 at 14:00

Instructions:

You should submit exercises in groups of **two**. Submissions **must** include the name and matriculation number of **both** students. Please submit only once per group. Solutions must be submitted in PDF format to CMS. You should provide solutions in English. If there are additional files, they can be downloaded from the material section in CMS. **Sharing answers with other groups is strictly prohibited!**

Exercise 1: Quiz (20 points)

Determine whether the following statements are true or false. For each **false** statement, explain in one sentence why it is incorrect.

(1) You have created or joined a submission team before submitting this exercise sheet.

(2) In Tamarin, we can state arbitrary functions and equations without repercussions.

(3) In Tamarin, the Dolev-Yao adversary can learn any value marked with $.

(4) The Dolev-Yao adversary can alter, drop, and insert messages.

(5) To state a property in Tamarin, we refer to the action facts of rules.

(6) The Dolev-Yao adversary controls the network and can use all non-private functions.

(7) To state that an action A cannot occur, we write $\forall i. A@i \implies \bot$.

(8) Applying a function symbol on a fresh variable t and sending it to the network will always result in the attacker learning t.

(9) Values in Tamarin are either fresh or public.

(10) The initial state of Tamarin's MSR contains all possible fresh values.

Exercise 2: Attack Finding (30 points)

(a) (15 points) Consider the following protocol in Alice & Bob notation.

$$A \to B : aenc(K_{AB}, pk_B)$$
$$B \to A : senc(N_B, K_{AB})$$
$$A \to B : senc(sign(N_B, sk_A), K_{AB})$$

Here, K_{AB} is a fresh key that A generates. We assume that A and B know each others public keys.

In the first message, A sends B a session key K_{AB}, encrypted with B's public key. Then, B challenges A with a nonce N_B encrypted under the session key. Finally, A authenticates

against B by signing the decrypted nonce with their secret key.

Unfortunately, the protocol is not secure.

(1) Prove this by providing a counterexample showing that a network attacker can impersonate an honest user of the protocol. You can provide the attack in Alice & Bob notation or as a diagram.
(2) Now that you found the attack, propose a fix for the protocol and argue why it is effective.

(b) (15 points) Consider the following protocol in Alice & Bob notation.

$$
\begin{aligned}
&A \to B : ID_A \\
&B \to A : N_B \\
&A \to B : senc(N_B, K_{AS}) \\
&B \to S : senc(\langle ID_A, enc(N_B, K_{AS})\rangle, K_{BS}) \\
&S \to B : senc(N_B, K_{BS})
\end{aligned}
$$

Here, ID_A is the initiator's identifier. We assume that the server S shares a secret key with both a A (K_{AS}) and B (K_{BS}).

In the first message, A sends its identifier to B. B then challenges A with a nonce N_B. A encrypts this nonce with the key it shares with the server K_{AS}. Then, B forwards this ciphertext and ID_A encrypted under the key K_{BS} to the server. Finally, the server decrypts everything and sends the nonce to B encrypted under their shared key K_{BS}.

Again, the protocol is not secure.

(1) Prove this by providing a counterexample showing that a network attacker can abuse parallel sessions to impersonate an honest user of the protocol.
(2) Now that you found the attack, propose a fix for the protocol and argue why it is effective.

Exercise 3: Protocol Modeling (20 points)

In this exercise, you will model a simplified version of the WPA2 protocol[1].

WPA2 is the de-facto standard protocol for secure WiFi connections between devices like smartphones and routers. To establish this secure connection, WPA2 defines the so-called 4-way handshake between the *authenticator* (the router) and the *supplicant* (the connecting device). The goal of the 4-way handshake is to establish a shared pairwise key between the authenticator and the supplicant. They will then use this key to encrypt subsequent communication. We will now consider a simplified version of this handshake.

(a) Consider the following protocol in Alice & Bob notation.

$$
\begin{aligned}
&A \to S : N_A \\
&S \to A : N_S \\
&A \to S : \text{“}ACK\text{”}
\end{aligned}
$$

First, the authenticator computes the nonce N_A and sends it to the supplicant. Then, the supplicant computes the N_S and sends it to the authenticator. When the authenticator

[1] `https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access`

receives this message, it derives and installs a key K from both nonces via a key-derivation function (kdf). To confirm this, the authenticator sends the string "ACK" towards the supplicant. Once the supplicant receives the string, it also derives and installs the key K.

i. (10 points) Write a set of multiset rewriting rules and equational theories to model the above described protocol.

ii. (5 points (bonus)) Formulate an *exists-trace* lemma, which states that both S and A can successfully run the protocol.

iii. (5 points (bonus)) Formulate a lemma, which states that *"After S finishes the protocol, an attacker cannot derive the key K".*

(b) Our first version of the protocol is vulnerable to a *man-in-the-middle* attack that allows an attacker to compute the key K. To fix this problem, we now require that the authenticator and the supplicant start the handshake with a shared secret; the preshared master key (PMK). In practice, this is the WiFi password that we enter into our device when we first connect to a network. When they compute the session key, A and S will now incorporate the PMK (i.e., $K = kdf(PMK, N_A, N_B)$).

i. (5 points) Update your multiset rewriting rules and lemmas if needed, accordingly.

ii. (5 points (bonus)) Formulate a lemma, which states that *"Whenever S finishes a run of the protocol, A must also have finished a run of the protocol".*

(c) The previous version of our protocol, allowed an attacker to send "ACK" messages to S and thus tricking S into believing that A successfully installed the derived key K. To avoid this, we now have A compute a MAC (message authentication code) with the session key K for the "ACK" message. S now first verifies the MAC before installing the key K.

i. (5 points) Update your multiset rewriting rules and lemmas if needed accordingly.

ii. (5 points (bonus)) Now, prove the lemma that says 'Whenever S finishes a run of the protocol, A must also have finished a run of the protocol'. (An informal proof/argument is sufficient.)

Exercise 4: Execution Traces (30 points)

Note that ++ does not refer to addition of natural numbers, but an 'add' operator for multisets.

(a) (8 points) Consider the following multiset rewriting system:

Rule 1 :
[ ]
−[InitC1(‘1’)]→
[Counter1(‘1’)]

Rule 2 :
[Counter1(‘1’++‘1’++‘1’)]
−[InitC2(‘1’)]→
[Counter1(‘1’++‘1’++‘1’), Counter2(‘1’)]

Rule 3 :
[Counter1(x)]
−[Increment(x++‘1’)]→
[Counter1(x++‘1’)]

Rule 4 :
[Counter1(x), Counter2(y)]
−[Modify(x++‘1’, y++‘1’)]→
[Counter1(x++‘1’), Counter2(y++‘1’)]

Are the following actions reachable? If yes, provide the execution trace. Otherwise, explain briefly why not.

(1) Modify('1'++'1'++'1'++'1'++'1', '1'++'1'++'1')
(2) Modify(x++x, x)
(3) Modify(x, x)
(4) Modify(x, x++'1')

(b) (12 points) Now, we modify the system by adding a fresh identifier to the Counters.

Rule 1 :
[Fr(~id)]
−[InitC1(~id, '1')]→
[Counter1(~id, '1')]

Rule 2 :
[Counter1(~id, '1'++'1'++'1')]
−[InitC2(~id, '1')]→
[Counter1(~id, '1'++'1'++'1'), Counter2(~id, '1')]

Rule 3 :
[Counter1(~id, x)]
−[Increment(~id, x++'1')]→
[Counter1(~id, x++'1')]

Rule 4 :
[Counter1(~id, x), Counter2(~id, y)]
−[Modify(~id, x++'1', y++'1')]→
[Counter1(~id, x++'1'), Counter2(~id, y++'1')]

Are the following actions reachable? If yes, provide the execution trace. Otherwise, explain briefly why not.

(1) Modify(~id, x++x, x)
(2) Modify(~id, x, x)
(3) Modify(~id, x, x++'1')

(c) (10 points) Lastly, we modify Rule 4.

Rule 4 :
[Counter1(~id, x++'1'), Counter2(~id, y)]
−[Modify(~id, x, y++'1')]→
[Counter1(~id, x), Counter2(~id, y++'1')]

Are the following actions reachable? If yes, provide the execution trace. Otherwise, explain briefly why not.

(1) Modify(~id, x++x, x)
(2) Modify(~id, '1'++'1', '1'++'1')
(3) Modify(~id, '1'++'1'++'1', '1'++'1'++'1')
(4) Modify(~id, x, x)
(5) Modify(~id, x, x++'1')