CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL
Institut für Informatik · AG Software Engineering
Priv.-Doz. Henning Schnoor

Engineering Secure Software Systems Winter 2020/21
Exercise Sheet 2

issued: November 10, 2020 **due:** November 19, 2020

Exercise 2.1, simple example protocol (10 Points)

We consider the following simple authentication protocol:

- Alice sends a message M to Bob, together with her name A,
- Bob answers with a Nonce N_b,
- Alice answers with the term $\mathsf{sig}_{k_A}([M, B, N_B])$.

Please answer the following questions:

1. What are the security properties guaranteed by the protocol?
2. What is the purpose of the nonce N_B? What happens if we omit it?
3. What happens if the B is removed from Alice's last message?

Exercise 2.2, Fixing Broken Authentication Protocols (10 Points)

Consider the two authentication protocols presented in the exercise class:

a)
1. $A \to B \quad (A, \mathsf{enc}^{\mathsf{a}}_{k_B}(N_A))$
2. $B \to A \quad (B, \mathsf{enc}^{\mathsf{a}}_{k_A}(N_A))$

b)
1. $A \to B \quad (\mathsf{enc}^{\mathsf{a}}_{k_B}(N_A), \mathsf{enc}^{\mathsf{a}}_{k_B}(A))$
2. $B \to A \quad (\mathsf{enc}^{\mathsf{a}}_{k_A}(N_A, N_B), \mathsf{enc}^{\mathsf{a}}_{k_A}(B))$

Both of these protocols can be attacked with a similar attack as the Needham-Schroeder protocol or the example protocol we covered in the first exercise class. Suggest changes to the protocols that address these problems, and argue why you think your revised versions of the protocols are secure. Be as specific as possible in what "secure" means in this case.

Solution

a) The problem with the protocol is that the identity of the sender is not cryptographically secured. Informally, the protocol can be changed to

1. $A \to B \quad (\mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A))$
2. $B \to A \quad (\mathsf{enc}^{\mathsf{a}}_{k_A}(B, N_A))$

Written as formal instances, the protocol looks as follows:

Alice

1. $\epsilon \quad \to \quad \mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A)$

Bob

1. $\mathsf{enc}^{\mathsf{a}}_{k_B}(A, x) \quad \to \quad \mathsf{enc}^{\mathsf{a}}_{k_A}(B, x)$

To prove security of the protocol, we need to be precise about the intended security property. We also want to cover attacks that use different sessions between honest and dishonest participants. We assume the following setup:

- Clearly, the protocol only provides security for Alice (i.e., the initiator role), as she gets the guarantee that Bob was active during the protocol run. Bob does not obtain any guarantee. We formalize security as requiring that the adversary does not learn the value N_A.
- Alice and Bob are honest, Charlie is the adversary
- Alice starts protocol sessions A_B and A_C with both Bob and Charlie (as initiator).
- Bob starts protocol sessions B_A and B_C with both Alice and Charlie (as responder).
- (To be complete, we would also need to consider sessions where Alice is the responder and Bob is the initiator. We leave these to the reader.)
- The "secure" session is A_B, we therefore add a corresponding BREAK-rule to this instance.

We therefore get the following formalization:

A_B

1. $\epsilon \quad \rightarrow \quad \mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A)$
2. $[\mathsf{BREAK}, N_A] \quad \rightarrow \quad \mathsf{FAIL}$

A_C

1. $\epsilon \quad \rightarrow \quad \mathsf{enc}^{\mathsf{a}}_{k_C}\left(A, N'_A\right)$

B_A

1. $\mathsf{enc}^{\mathsf{a}}_{k_B}(A, x) \quad \rightarrow \quad \mathsf{enc}^{\mathsf{a}}_{k_A}(B, x)$

B_C

1. $\mathsf{enc}^{\mathsf{a}}_{k_B}\left(C, x'\right) \quad \rightarrow \quad \mathsf{enc}^{\mathsf{a}}_{k_C}\left(B, x'\right)$

At attack now is an execution order and a substitution. First, note that the instance A_C is actually not required, since the nonce N'_A does not appear in any relevant instance, and therefore the adversary can simply generate the term sent by A_C on its own. Also, we can without loss of generality assume that the adversary triggers the FAIL-action of A_B as last action in the attack. Therefore, the execution order is defined by a permutation of the three first steps of the instances A_B, B_A, and B_C. Since the adversary's goal involves opening the decryption $\mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A)$, and only the instance B_A can do this, we know that A_B must be activated before B_A. It is also clear that the instance B_C does not help the adversary, since this instance expects a term of the form $\mathsf{enc}^{\mathsf{a}}_{k_B}\left(C, x'\right)$. Since we already eliminated the instance A_C above, we know that the term that B_C waits for is generated by the adversary herself. Therefore, she also knows the plaintext and therefore does not need the instance B_C. Hence, only the instances A_B and B_A are relevant, and their order is fixed. Therefore, the execution order is

a) A_B,

b) B_A.

This also fixes the first message sent by A_B, since the adversary must deliver ϵ. To "open" the encryption, the adversary has no other option that to deliver this message to Bob, and from the resulting term she cannot learn the value N_A. Therefore, the protocol indeed guarantees security in the above sense.