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

Engineering Secure Software Systems Winter 2020/21
Exercise Sheet 4

issued: November 24, 2020 **due:** December 3, 2020

Exercise 4.1, Formal Representation of the Woo Lam Protocol (10 Points)

Study the authentication protocol by Woo and Lam (see slide 17 of the lecture from November 10).

1. Specify the protocol as sequence of receive/send actions, once in the intended execution between Alice and Bob, and once in a form that allows to model the attack introduced in the lecture.
2. Specify the attack on the protocol formally.
3. How can we modify the protocol in order to prevent this attack?

Solution

1. The following is the protocol as r/s actions in the intended setting. We therefore have three instances: Alice, Bob, and the server.

   Alice
   1. $\epsilon \rightarrow A$
   2. $x \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{AS}}(x)$

   Bob
   1. $A \rightarrow N_B$
   2. $y \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(A, y)$

   Server
   1. $\mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left(A, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}(z)\right) \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(z)$

2. The protocol setup containing all instances that are present in the attack presented in the lecture is as follows:

   - C is controlled by the adversary, hence the initial adversary knowledge is $I = \{k_{CS}, \mathsf{BREAK}\}$
   - There are two protocol instances modeling sessions by honest Bob. Bob acts as responder in both of these instances (i.e., runs the "Bob" program from above). In one instance, Bob expects messages from Alice (we denote this instance with $\text{Bob}_{\text{Alice}}$), in the second instance (denoted with $\text{Bob}_{\text{Charlie}}$) he expects messages from Charlie.
   - Similarly, there are two instances modeling the two sessions of the server. In both of these, the server expects Bob in the responder role of the protocol. In one instance (denoted $\text{Server}_{\text{Alice}}$), the server expects Alice as the initiator, in the second one (denoted $\text{Server}_{\text{Charlie}}$), he expects Charlie.
   - We do not model a protocol instance containing Alice's rules, since in the attack scenario, no honest participant performs the initiator role.

   The attack is against the instance $\text{Bob}_{\text{Alice}}$, since Alice is assumed to be honest (the adversary knowledge does not contain any of Alice's keys): If the protocol run completes, then Bob assumes that Alice in fact participated in the protocol run. Therefore, the FAIL-rule must also be part of the instance $\text{Bob}_{\text{Alice}}$: When the instance $\text{Bob}_{\text{Alice}}$ receives the final message, the protocol is broken.

$\text{Bob}_{\text{Alice}}$

1. $A \rightarrow N_B$
2. $y \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(A, y)$
3. $\mathsf{enc}^{\mathsf{s}}_{k_{BS}}(N_B) \rightarrow \mathsf{FAIL}$

$\text{Bob}_{\text{Charlie}}$

1. $C \rightarrow N'_B$
2. $y' \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(C, y')$

$\text{Server}_{\text{Alice}}$

1. $\mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left(A, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}(z)\right) \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(z)$

$\text{Server}_{\text{Charlie}}$

1. $\mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left(C, \mathsf{enc}^{\mathsf{s}}_{k_{CS}}(z')\right) \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{BS}}(z')$

Note that the different instances of Bob use different nonces and variables. Also note that our modeling does not include the message containing the "trash" symbol from the lecture slides, since this clearly cannot be expressed in the formal model (adding a rule involving a trash symbol to the server's program would be very unnatural and would also require a case distinction in the server's modeling). However, modeling this message is not required, since the adversary can simply leave out the corresponding receive/send action from the server instance $\text{Server}_{\text{Alice}}$.

3. The attack on the protocol consists of the execution order and the substitution:
   - The execution order is as follows (the last step is for triggering the FAIL-rule):
     a) $\text{Bob}_{\text{Alice}}$
     b) $\text{Bob}_{\text{Charlie}}$
     c) $\text{Bob}_{\text{Alice}}$
     d) $\text{Bob}_{\text{Charlie}}$
     e) $\text{Server}_{\text{Charlie}}$
     f) $\text{Bob}_{\text{Alice}}$
4. The substitution is easily obtained from the above formalization of the instances and the attack as presented in the lecture:
   - $\sigma(y) = \sigma(y') = \mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)$,
   - $\sigma(z') = N_B$.
5. To demonstrate that Charlie can derive the messages he sends, we list the actually appearing terms in the receive/send actions that are performed by the attack—the terms are obtained by simply replacing the variables appearing in the instances above with their values assigned by σ

step / instance	receive		send
1. $\text{Bob}_{\text{Alice}}$	A	$\rightarrow$	N_B
1. $\text{Bob}_{\text{Charlie}}$	C	$\rightarrow$	N'_B
2. $\text{Bob}_{\text{Alice}}$	$\mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)$	$\rightarrow$	$\mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left(A, \mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)\right)$
2. $\text{Bob}_{\text{Charlie}}$	$\mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)$	$\rightarrow$	$\mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left(C, \mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)\right)$
1. $\text{Server}_{\text{Charlie}}$	$\mathsf{enc}^{\mathsf{s}}_{k_{BS}}\left(C, \mathsf{enc}^{\mathsf{s}}_{k_{CS}}(N_B)\right)$	$\rightarrow$	$\mathsf{enc}^{\mathsf{s}}_{k_{BS}}(N_B)$
3. $\text{Bob}_{\text{Alice}}$	$\mathsf{enc}^{\mathsf{s}}_{k_{BS}}(N_B)$	$\rightarrow$	FAIL

Exercise 4.2, Otway Rees Protocol (10 Points)

Consider the following protocol (Otway-Rees-Protocol):

1. $A \to B \quad [M, A, B, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}([N_a, M, A, B])]$
2. $B \to S \quad [M, A, B, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}([N_a, M, A, B]), \mathsf{enc}^{\mathsf{s}}_{k_{BS}}([N_b, M, A, B])]$
3. $S \to B \quad [M, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}([N_a, k_{AB}]), \mathsf{enc}^{\mathsf{s}}_{k_{BS}}([N_b, k_{AB}])]$
4. $B \to A \quad [M, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}([N_a, k_{AB}])]$
5. $A \to B \quad \mathsf{enc}^{\mathsf{s}}_{k_{AB}}(\mathsf{FAIL})$

1. Why are the subterms M, A, and B in the second message sent both encrypted and as plaintext?
2. Why is the nonce N_b encrypted in message 2?
3. Is the protocol secure? (You do not need to give a formal proof of security or insecurity.)

Solution The protocol is insecure, since the adversary can fool A into accepting (M, A, B) as the key. This works as follows:

- In Alice's last step, she expects the message $[M, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}([N_a, k_{AB}])]$.
- To perform the attack, the adversary simply needs to deliver the message $[M, \mathsf{enc}^{\mathsf{s}}_{k_{AS}}([N_a, (M, A, B)])]$ instead.
- The adversary learns the relevant (encrypted) part of this message in the second step of the protocol.

Note that this attack relies on the fact that the principals do not perform type-checking, i.e., Alice is willing to accept a complex term as a key where maybe a nonce would be more natural. Such an attack can therefore be circumvented in a real implementation, however this defense cannot be expressed in our model. (One can of course extend our model with type-checking, however this is not straight-forward when we e.g., discuss type-checking of ciphertexts.)

Exercise 4.3, Security Modeling Issues: Are we Missing Something? (10 Points)

In the lecture, we defined security of a protocol as, essentially, unreachability of a state in which the adversary learns the constant FAIL. However, this FAIL-constant obviously does not have a correspondance in a real implementation of a protocol. In particular, the rules releasing the FAIL-constant are removed from the protocol in a real implementation. As a consequence, a potential security proof of a protocol in our formal model treats a different protocol than the protocol running in a real implementation.

Are there cases where this difference results in an insecure protocol that can be proven secure in our formal model? If this is the case, how can we circumvent this issue?

Solution This can in fact happen, as an example consider the following protocol: The protocol goal is for Alice and Bob to exchange a secret nonce N_A, the security goal is thus for the adversary to not learn N_A. We assume that there is a PKI in place.

Alice's protocol instance is as follows:

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

Bob does not perform any actions in this protocol run, and therefore does not need to be specified. Clearly, assuming that the adversary does not know $\hat{k}_B$, the original protocol is secure, since the adversary can never learn N_A. However, if we remove the second rule, the protocol is completely insecure with respect to the security goal of protecting N_A. A natural way to address this issue is to assume that the BREAK-rules only appear at the end of a protocol instance. In this case, as long as we restrict ourselves to reachability conditions, a secure protocol remains secure when removing the BREAK-rules.