University of Stuttgart
Institute of
Information Security

Security and Privacy
Summer Term 2018
Prof. Dr. Ralf Küsters

Homework 0

Submission into the *red SEC mailbox* in front of Room 2.041 until April 24, 2018, 14:00.

General Notes

- If you encounter difficulties, you SHOULD[1] ask the teaching assistants (see ILIAS for contact information).
- To solve the homework, you SHOULD form teams of 3 people.
- Your team size MUST NOT exceed 3 people.
- You MUST submit your homework on paper (one submission per team).
- You are free to choose whether you write your solutions in German or in English.
- If your submission contains multiple sheets, you MUST staple them.
- Each sheet of your submission MUST include all team member's names and matriculation numbers.
- If you do not adhere to these rules, you risk losing points.

Problem 1: Needham-Schroeder Protocol (4 points)

The Needham-Schroeder-Lowe (NSL) protocol shown in the lecture was proposed to prevent the attack on the Needham-Schroeder protocol. While the NSL protocol can be proven secure, its security strongly depends on fresh nonces, i.e., for every session of a protocol new nonces have to be chosen.

a) Briefly describe why the attack on the Needham-Schroeder protocol fails for the NSL protocol.

b) Find an attack on the NSL protocol that is possible if Alice uses the same nonce N_A during two different sessions. Describe your attack by giving the message flow during the attack. Which security properties are violated by your attack?

Problem 2: Another attack (4 points)

The goal of the following protocol is mutual authentication, i.e., if both parties complete the protocol successfully, it should be the case that they actually talked to each other. The protocol assumes that both parties, A and B, share a secret key k. As usual, N_A and N_B denote nonces generated by A and B, respectively.

$$\begin{array}{lrcl} 1. & A \to B & : & \{N_A\}_k^s \\ 2. & B \to A & : & \{N_B\}_k^s, N_A \\ 3. & A \to B & : & N_B \end{array}$$

Find an attack on this protocol and give its message flow. Why is authentication violated by your attack?

[1] SHOULD, MUST, and MUST NOT are used as defined in RFC2119.

Problem 3: Woo and Lam Mutual Authentication Protocol (4 points)

Another protocol meant for mutual authentication and key exchange is the Woo and Lam protocol. Unlike the NSL protocol, this protocol is run between three parties: Two users A and B that want to authenticate each other and establish a key, and a server S that already shares symmetric encryption keys K_{AS} with A and K_{BS} with B. In a nutshell, the server generates a new symmetric key K_{AB} and sends it to both A and B using the already established keys. To be more precise, the protocol is defined as follows:

1. $A \to B : A, N_A$
2. $B \to A : B, N_B$
3. $A \to B : \{A, B, N_A, N_B\}^s_{K_{AS}}$
4. $B \to S : \{A, B, N_A, N_B\}^s_{K_{AS}}$, $\{A, B, N_A, N_B\}^s_{K_{BS}}$
5. $S \to B : \{B, N_A, N_B, K_{AB}\}^s_{K_{AS}}$, $\{A, N_A, N_B, K_{AB}\}^s_{K_{BS}}$
6. $B \to A : \{B, N_A, N_B, K_{AB}\}^s_{K_{AS}}$, $\{N_A, N_B\}^s_{K_{AB}}$
7. $A \to B : \{N_B\}^s_{K_{AB}}$

Show that the Woo and Lam protocol is insecure. That is, show that an adversary can run two protocol sessions with B where the adversary plays the roles of A and S without knowing the secret symmetric keys K_{AS} and K_{BS} such that at least one of these sessions runs successfully. As a result, B thinks he has established a session key with A, although A (and S) has never been involved. Describe your attack by writing down the message flow.

Hints

- While the adversary does not know the keys K_{AS} and K_{BS}, he can (ab)use B to encrypt messages of a specific format with K_{BS}.
- Note that B cannot "see" whether messages that are encrypted by K_{AS} have a specific format. To him, those encrypted messages look like random bits. Thus, the adversary can send random bits instead.
- The attack assumes that it is not possible to distinguish a key K_{AB} and a nonce as both are just some arbitrary random bit strings. This allows the adversary to let B encrypt a specific message in the fourth step, which will be accepted by another session of B in the fifth step.