

Exercise Sheet 2

Out: Thu, Nov 5, 2009

Due: Tue, Nov 17, 2009, 10am

A request: If possible, please use white paper (without lines or grid) for writing your solution. I find that the grid distracts from the text and makes the solution harder to read. Also, please leave a large margin for corrections.

Problem 1: Automating the protocol analysis

In exercise sheet 1, problem 2, we have analyzed a simple example protocol by hand. This time, your task is to make an automated analysis:

- Write an input file for the SPASS theorem prover to automatically prove the security of the protocol from exercise sheet 1, problem 2.
- To check that your SPASS input makes sense, try out what happens when the hash function is insecure, i.e., when we additionally have the deduction rule $\frac{S \vdash \text{hash}(x)}{S \vdash x}$.

The necessary changes to the SPASS input should be minimal.

Hint: You find a very short tutorial for SPASS on <http://www.spass-prover.org/tutorial.html>. You do not need to install SPASS on your computer, there is a web-interface that allows to execute SPASS input files: <http://www.spass-prover.org/WebSPASS>.

Hint: Do not try to model the deduction relation in its full generality (i.e., as a relation between sets of messages and messages). This will be unnecessarily difficult because you would have to model sets. Instead, define a predicate `knows1` such that `knows1(x)` holds iff $S_1 \vdash x$, and similarly `knows2`. Furthermore, it is OK to do a slight overapproximation if necessary.

Hint: If you do not manage to write a SPASS input file for the whole protocol analysis, you might try to break it down and first let SPASS show that $S_2 \not\vdash N_{secret}$ assuming that $K' = K_2$. This would not be the full solution, but it will give you part of the points.

Problem 2: Computationally unsound protocols

When proving computational soundness of passive secrecy properties, various conditions on the passive secrecy properties have to hold. In the lecture, we have already seen that we need to exclude key-cycles (Definition 24 in the lecture notes). During the proof, we will meet additional conditions. For example, freshness of randomness requires that any

two different ciphertexts use different randomness. In the following two problems, you are required to design a computational implementation such that certain protocols are not computationally sound.

In both problems, you will need the definition of IND-CPA security which we give here for reference:

Definition 1 (IND-CPA) An encryption scheme (K, E, D) consisting of probabilistic polynomial-time algorithms K (key-generation), E (encryption), and D (decryption) is IND-CPA secure if for any polynomial-time oracle machine Adv , there is a negligible function μ such that for all k we have $|\text{Pr}_0(k) - \text{Pr}_1(k)| \leq \mu(k)$. Here

$$\text{Pr}_i(k) := \Pr[b = 1 : \text{key} \leftarrow K(1^k), b \leftarrow \text{Adv}^{\mathcal{E}_i(\text{key}, \cdot)}(1^k)].$$

and $\mathcal{E}_0(\text{key}, \cdot)$ is an oracle that returns $E(1^k, \text{key}, m)$ when queried with a message m , and $\mathcal{E}_1(\text{key}, \cdot)$ is an oracle that returns $E(1^k, \text{key}, 0^{|m|})$ when queried with a message m . (The relevant difference to IND-OT-CPA as defined in Definition 1 in the lecture notes is that in the present definition, the adversary may request the encryptions of many messages, while in the IND-OT-CPA definition, he may request only one.)

We say a computational implementation A is IND-CPA secure if (K, E, D) is IND-CPA secure where (i) all A_N for $N \in \mathbf{N}$ are identical, (ii) $K(1^k) := A_N(1^k)$ for some $N \in \mathbf{N}$, (iii) $E(1^k, \text{key}, m) := A_{\text{enc}}(1^k, \text{key}, m, A_N(1^k))$ for some $N \in \mathbf{N}$. (There is no need to specify D because it does not occur in Definition 1.)

- (a) **Key-cycles.** Let $\varphi := (N_{\text{secret}}, (\text{enc}(N_{\text{secret}}, N_{\text{secret}}, R)))$ with $N_{\text{secret}}, R \in \mathbf{N}^P$. (The most trivial key-cycle.) Design a computational implementation A that is IND-CPA secure such that φ does not hold computationally. (You do not need to prove that A is IND-CPA secure.)

Hint: Assume an encryption scheme (K', E', D') that is IND-CPA secure and modify it to get another, still IND-CPA secure scheme (K, E, D) which you use for your computational implementation.

- (b) **Fresh randomness.** Let $\varphi_1 := (N_{\text{secret}}, (\text{enc}(K_1, N_{\text{secret}}, R), \text{enc}(K_2, N_{\text{secret}}, R)))$ with $N_{\text{secret}}, K_1, K_2, R \in \mathbf{N}^P$. Design a computational implementation A that is IND-CPA secure such that φ_1 does not hold computationally. (You do not need to prove that A is IND-CPA secure.)

Alternatively, if you prefer, you can do the same for $\varphi_2 := (N_{\text{secret}}, (\text{enc}(K_1, N_{\text{secret}}, R), \text{enc}(K_1, N, R), N))$ with $N_{\text{secret}}, K_1, N, R \in \mathbf{N}^P$ instead of φ_1 .

Hint: Same hint as in part (a). Think of one-time-pads (which are, however, not IND-CPA secure). Furthermore, if you happen to need randomness that is longer than a nonce, use a pseudorandom generator.