Exercise Class December 17, 2020

This is a slightly edited and re-formated transcript of the live notes taken in class.

Leftovers

Task: Woo Lam formalization

Alice:

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

Bob: need to use variable as LHS on second rule!

1. $A \rightarrow N_B$
2. $y \rightarrow \ldots$

Server:
...

Task: no unique successful attack

easiest approach

- Alice: $x \rightarrow \mathsf{FAIL}$

fails, because minimal attack is: $\sigma(x) = \epsilon$

second-easiest, formally correct solution

- Alice: $x \rightarrow \mathsf{FAIL}$
- Bob: $y \rightarrow \mathsf{FAIL}$

(cheating, because FAIL appears in two instances)

real solution

- Bob: $\epsilon \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{ABC}}(N_B)$
- Charlie: $\epsilon \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{ABC}}(N_C)$
- Alice: $\mathsf{enc}^{\mathsf{s}}_{k_{ABC}}(x) \rightarrow \mathsf{FAIL}$

minimal attacks: adv can activate Bob or Charlie

Task: parsing lemma proof

Looking for: situation in which $\sigma(x)$ contains something not derivable by the adversary. Since we proved the parsing lemma, our example must violate at least one prerequisite of the parsing lemma or the indirect assumption in the proof. We violate the indirect "$\sigma(x)$ does not match a term from the protocol" assumption, and choose a term that matches protocol rules. Example protocol:

- Alice: $\epsilon \rightarrow \mathsf{enc}^{\mathsf{s}}_{k_{AB}}(N_A, N_A, N_A)$
- Bob: $\mathsf{enc}^{\mathsf{s}}_{k_{AB}}(x) \rightarrow FAIL$

In a successful attack, $\sigma(x) = [N_A, N_A, N_A]$ must hold.

Exercise 06

Task: applying the Rusinowitch Turuani Theorem

There was a misunderstanding with the exercise task, resultung from bad wording on my part (the word "instance" here was meant as "input instance to the Rusinowitch Turuani algorithm," not as protocol instance), the following is the updated (and hopefully more clear) task description:

Exercise: applying the Rusinowitch Turuani Theorem, updated In the lecture, modelled the Needham-Schroeder protocol as an input to INSECURE such that the attack is detected. However, this required us to already specify the "correct" sessions ("Alice with Charlie, Charlie with Bob") manually. For automatic analysis, such a manual step should not be required. Can you come up with a pre-processing step that makes this manual step unnecessary?

More precisely: Can you come up with a mechanism translating a natural representation of a protocol (e.g., as the list of "intended instances" for a single session) into a protocol P such that

- P can be used as input for the Rusinowitch-Turuani algorithm for INSECURE,
- P contains all relevant protocol instances (i.e., an initiator with Alice's identity expecting to communicate with a responder with Charlie's identity, and a responder with Bob's identity expecting to communicate with an initiator with Alice's identity),
- P is formally insecure if and only if there is a successful attack on any number of sessions with any set of identities in which the original protocol is run?

Note: You do not need to make your constructions formal.

Example: Needham-Schroeder protocol with just Alice and Bob instances is not "intuitively" secure

Approach: Assume we know number of participants, then: add instances (capital letters indicate which "side" of the protocol we are modeling):

- ALICE → Bob
  - $\epsilon \rightarrow \mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A)$

    - $\mathsf{enc}^{\mathsf{a}}_{k_A}(N_A, x) \rightarrow \mathsf{enc}^{\mathsf{a}}_{k_B}(x)$
- ALICE → Bob
    - $\epsilon \rightarrow \mathsf{enc}^{\mathsf{a}}_{k_B}\left(A, N^1_A\right)$
    - $\mathsf{enc}^{\mathsf{a}}_{k_A}\left(N^1_A, y\right) \rightarrow \mathsf{enc}^{\mathsf{a}}_{k_B}\left(y\right)$
- ALICE → Bob
- ALICE → Bob
- Alice → BOB
- ALICE → “Charlie”
- “Charlie” → BOB
- “Charlie” → ALICE

questions:

- where to add BREAK/FAIL rules? (analysis of attack against responer: add BREAK/FAIL rule to responder instance, e.g., to “Charlie” → BOB rule is: Bob (receiver): $[BREAK, x, N_B] \rightarrow FAIL$
- issue with adding BREAK/FAIl rule to instance Bob/“Charlie:” After a successful, “secure” protocol run with Charlie, Charlie knows the nonces from this session and can activate the BREAK/FAIL rule.
- add BREAK/FAIL to instance where two honest identities communicate, i.e., one of the ALICE → Bob (if we analyse security for sender) or Alice → BOB (if we analyse security for receiver).
- you don’t know how many copies of each to add

The translation cannot work, because the unbounded insecurity problem is undecidable.