Engineering Secure Software Systems

November 24, 2020: Crypto Protocols: Security Defintion

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part I: Crypto Protocols

Overview

Overview

Executing Instances II

situation

- attacker controls scheduling, network

assumption: $\mathcal{A}$ controls everyhing we don't.

next step: message delivery

step $A \rightarrow C$ *(Charlie controlled by $\mathcal{A}$)*

- belongs to instance "A with C ($\mathcal{A}$)"
- message created by protocol instance

step $A \rightarrow B$ *(Alice impersonated by $\mathcal{A}$)*

- belongs to instance "A ($\mathcal{A}$) with B"
- messages created by adversary

questions

- which messages can $\mathcal{A}$ send?
- restricted only by cryptography!
- what does this mean concretely?
- $\rightsquigarrow$ recall Dolev-Yao closure!

Protocol Runs: Formal Definition

definition: attack

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ protocol, initial adversary knowledge I, i-th rule of $\mathcal{I}_j$ named $r_i^j \to s_i^j$. An attack on P consists of

- an execution order o for P,
- a substitution σ on the variables in P such that

$$\sigma(r^{o(k)}_{\#o(k)}) \in \mathrm{DY}\left(I \cup \left\{\sigma(s^{o(\ell)}_{\#o(\ell)}) \mid 0 \leq \ell < k\right\}\right).$$

terms

- $\sigma(r^{o(k)}_{\#o(k)})$: received in step k
- $\sigma(s^{o(k)}_{\#o(k)})$: sent in step k

simplification

- identify “protocol run” and “attack”
- protocol cannot be executed without $\mathcal{A}$ (network)

next step

definition of secure protocol / successful attack

difficulty: different goals

- authentication: Bob only accepts when he “talked” to Alice
- key exchange: adversary obtains no information about the key
- secure channel: adversary has no information about, and cannot influence, messages exchanged on the channel
- electronic voting: votes authenticated, counted correctly, “secret”
- ...

Successful Attack: Definition

variable “attack definition”

- success criterion for attack depends on protocol (goal)
- automatic analysis: security definition should be part of algorithm input
- integrate security definition into protocol: let designer specify security
- add “challenge interface” for adversary: define instances so that $\mathcal{A}$ can get constant **FAIL** if successful (**BREAK**-rules)

definition: successful attack

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ with initial adversary knowledge I. Attack (o, σ) is successful, if:

$$\mathsf{FAIL} \in \mathsf{DY}\left(I \cup \left\{\sigma(s^{o(\ell)}_{\#o(\ell)}) \mid 0 \leq \ell < |o|\right\}\right).$$

literature: **secret** instead of **FAIL**

example

application of definition to Needham-Schroeder protocol

steps

1. formalise protocol as r/s actions
2. formalise security specification: add BREAK-rules
3. find execution order for attack
4. find substitution for attack
5. check that attack is successful

Needham Schroeder Formalization with Attacker

Alice's r/s rules

$r_0^A \rightarrow s_0^A \quad \epsilon \rightarrow \mathrm{enc}^a_{k_C}(A, N_A)$

$r_1^A \rightarrow s_1^A \quad \mathrm{enc}^a_{k_A}(N_A, y) \rightarrow \mathrm{enc}^a_{k_C}(y)$

Bob's r/s rules

$r_0^B \rightarrow s_0^B \quad \mathrm{enc}^a_{k_B}(A, x) \rightarrow \mathrm{enc}^a_{k_A}(x, N_B)$

$r_1^B \rightarrow s_1^B \quad [\mathrm{BREAK}, N_B] \rightarrow \mathrm{FAIL}$

attack

execution order $o = ABAB$, substitution: $\sigma(x) = N_A$, $\sigma(y) = N_B$

actual messages

step	message sent by $\mathcal{A}$	recipient of $\mathcal{A}$ message	message received as reply
0	ϵ	A	$\mathrm{enc}^a_{k_C}(A, N_A)$
1	$\mathrm{enc}^a_{k_B}(A, N_A)$	B	$\mathrm{enc}^a_{k_A}(N_A, N_B)$
2	$\mathrm{enc}^a_{k_A}(N_A, N_B)$	A	$\mathrm{enc}^a_{k_C}(N_B)$
3	$[\mathrm{BREAK}, N_B]$	B	FAIL

Exercise

Task (Formal Representation of the Woo Lam Protocol)

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?

Exercise

Task (Otway Rees Protocol)

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

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

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.)

Successful Attack: Examples and Limits

examples

- key exchange $\rightsquigarrow$ key k
  - $\mathcal{A}$: [BREAK, k] to Alice
  - Alice replies with FAIL
- electronic voting $\rightsquigarrow$ Bob votes $vote_B$
  - $\mathcal{A}$: [BREAK, $vote_B$] to Bob
  - Bob replies with FAIL

important

BREAK-messages distinguishable from “normal” messages: FAIL value used nowhere else

issues

- “bug” in the examples?
- limits of this approach?

careful

voting example does not work like this! How can we fix this?

Exercise

Task (Security Modeling Issues: Are we Missing Something?)

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?