COL876: SPECIAL TOPICS IN FORMAL METHODS

Formal verification of security protocols

Lecture 3, 3 August 2023

QUIZ!

I. Write out the proof rules for symmetric encryption and hashing. Point out which ones are the constructors and which the destructors.

II. What is the size of the following term? Also write out its subterms. $\mathsf{aenc}(\ \mathsf{pair}(\ \mathsf{aenc}(\ m,\ \mathsf{pk}(k_1)\ ),\ \mathsf{pair}(\ n_1,\ n_2\ )\ ),\ \mathsf{pk}(k_2)\ )$

BONUS: Find a derivation of *m* from the following *X*.

$$X = \{$$

$$\mathsf{aenc}(\ \mathsf{pair}(\ n,\ \mathsf{aenc}(\ k_1,\ \mathsf{pk}(k_3)\ )\ ),\ \mathsf{pk}(k_1)\ ),$$

$$\mathsf{aenc}(\ \mathsf{pair}(\ \mathsf{aenc}(\ k_3,\ \mathsf{pk}(k_2)\ ),\ k_3\ ),\ \mathsf{pk}(k_2)\ ),$$

$$\mathsf{aenc}(\ \mathsf{pair}(\ m,\ k_2\ ),\ \mathsf{pk}(k_3)\ )$$

$$\}$$

RECAP: PASSIVE INTRUDER PROBLEM

- Given an X and a t, check if $X \vdash t$ using our proof system.
- Easy to do for the system with pairing and encryption: PTIME!
- Basically models a “benign intruder”: just snoops on the channel but nothing more
- Unlikely to catch “real” bugs in the protocol due to intruder orchestrations

DOLEV-YAO INTRUDER

- Intruder I cannot break encryption, but, on the public channel, can
  - see any message sent on the channel
  - block any message from reaching the intended recipient
  - re-route any message to any principal
  - masquerade as any principal and send messages in their name
  - initiate new communication according to the protocol
  - generate messages — according to some rules

DOLEV-YAO INTRUDER

- Intruder I cannot break encryption, but, on the public channel, can
  - see any message sent on the channel ← Passive intruder problem
  - block any message from reaching the intended recipient
  - re-route any message to any principal
  - masquerade as any principal and send messages in their name
  - initiate new communication according to the protocol
  - generate messages — according to some rules

DOLEV-YAO INTRUDER

- Intruder I cannot break encryption, but, on the public channel, can
  - see any message sent on the channel
  - block any message from reaching the intended recipient
  - re-route any message to any principal
  - masquerade as any principal and send messages in their name
  - initiate new communication according to the protocol
  - generate messages — according to some rules

} Active intruder problem

ACTIVE INTRUDER PROBLEM

Given a protocol *P* and a term t, check if there is an execution of P, at the end of which, the intruder can derive t.

ACTIVE INTRUDER PROBLEM

- No explicit X; execute *P* and generate X, then check derivability.
- Have to check all possible executions, with a passive intruder problem module
- What does it even mean to execute a protocol? What does an execution look like?

EXECUTING A PROTOCOL

- Multiple sessions running in parallel.
- What all do agents need to keep track of in each session?
  - Which session they are currently involved in
  - Intended agents involved in any action by them
  - New terms generated as part of a send
  - Terms received “instead of what was sent” in a receive…

REMEMBER THIS?

- On a public network, two people share a randomly generated value m, which they want kept secret.

$$A \rightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$$
$$B \rightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

- Proving secrecy of m needs us to solve the active intruder problem

FORMALIZING EXECUTIONS

$$A \rightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$$
$$B \rightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

- Two “roles”, *init* and *resp* (described formally on next slide)
- Each parametrized by terms that are neither generated afresh, nor received. Which ones?

FORMALIZING EXECUTIONS

$$A \rightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$$

$$B \rightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

```
init(ski: skey, pkr: pkey) {
   new n: bytes;
   send(pk(ski), aenc(n,pkr));
   recv(x: bytes);
   if (adec(x,ski) ≠ n)
     error;
}
```

```
resp(skr: skey) {
   recv(k: pkey, y: bytes);
   let
      z = adec(y, skr)
   in
      send(aenc(z,k));
}
```

MORE ABOUT EXECUTIONS

- Some *instance* of each role executed by agents on the network
- Instances give meaning to parameters and variables
- Parameters: Generated by agents for sending (agent names, random etc)
- Variables: Only for received terms; given meaning by intruder!
- Man-in-the-middle attack involves *init*(A, B) and *resp*(B)
- An execution is an *interleaving* of finitely many instances of roles

MORE ABOUT EXECUTIONS

- Are all interleavings valid executions? No!

MORE ABOUT EXECUTIONS

- Are all interleavings valid executions? No!
- Honest agents should be able to construct a message to send it
- More importantly: intruder should be able to construct a message corresponding to a variable
- Constructing a message: deriving it using the proof system from their "current knowledge"

MORE ABOUT EXECUTIONS

- Needs us to check derivability at each step, but also update knowledge
- Initial knowledge state: constants, names/pubkeys of other agents, own secret key
- For every send by A
  - Check derivability from A's current local knowledge
  - Add sent message to I's knowledge state
- For every receive by A
  - Check derivability from I's current knowledge
  - Add received message to A's state

MORE ABOUT EXECUTIONS

- Each agent (and I) has a local knowledge state
- Global state: collection of these local states
- Enabled actions induce a transition with global state update
- A run of this transition system = an execution of the protocol

ACTIVE INTRUDER PROBLEM

- Passive intruder problem module is decidable
- Still need to check all possible executions though!
- Well-formedness lets us assign “sensible” values to parameters
- Unboundedly many possible values for variables though
- Unboundedly many such instances running in parallel
- Obviously undecidable

ACTIVE INTRUDER PROBLEM

- Can make it decidable by bounding one or more of these
- Bounding the number of instances is enough!
- What about parameters and variables?
- Very nifty technique by Rusinowitch and Turuani
- *Active intruder problem with boundedly many sessions in* NP [RT03]

KEY IDEAS [RT03]

- Parameters
  - Generated fresh (small: constants, names, random values), or
  - Depend on values received (variables!) earlier in the run
- Can still assign arbitrarily large values to variables
  - Crucial: Assignment done by I, to somehow violate property
  - Won't use a huge term if a small one will give same outcome
- Enough for intruder to use "relatively" small terms for variables

SEGUE: MORE INFERENCE

- Recall: inference for new messages done via a proof system
- Can have an alternative presentation
- An *equational theory* for all the functions in the term algebra
- Capture behaviour via equations, instead of proof rules

INFERENCE FOR MESSAGES

$\dfrac{X \vdash (t_1, t_2)}{X \vdash t_i}\ \textbf{split}$	$\dfrac{X \vdash t \quad X \vdash u}{X \vdash (t, u)}\ \textbf{pair}$
$\dfrac{X \vdash \mathrm{enc}(t, \mathrm{pk}(k)) \quad X \vdash k}{X \vdash t}\ \textbf{adec}$	$\dfrac{X \vdash t \quad X \vdash \mathrm{pk}(k)}{X \vdash \mathrm{enc}(t, \mathrm{pk}(k))}\ \textbf{aenc}$

$$\mathrm{fst}((t_1, t_2)) = t_1$$

$$\mathrm{snd}((t_1, t_2)) = t_2$$

$$\mathrm{adec}(\mathrm{aenc}(t, \mathrm{pk}(k)), k) = t$$

NEXT TIME

- Represent protocols as programs
- Use equational theory towards ensuring well-formedness
- One possible model for automated verification
- Different kinds of properties that can be verified