Formal verification of security protocols

Lecture 5, 10 August 2023
APPLIED-PI CALCULUS: GRAMMAR

\[ P, Q := \begin{align*}
\text{plain process} & \quad \circ \\
\text{null process} & \quad P \mid Q \\
\text{parallel composition} & \quad !P \\
\text{replication} & \quad \nu n.P \\
\text{name restriction} & \quad \text{if } t_1 = t_2 \text{ then } P \text{ else } Q \\
\text{conditional branching} & \quad \text{in}(c, x).P \\
\text{receive action} & \quad \text{out}(c, t).P \\
\text{send action} & \quad \text{let } x = t \text{ in } P \\
\text{let binding} & \quad \end{align*} \]
FORMALIZING EXECUTIONS

\[A \to B : A, \text{enc}(m, \text{pk}(B))\]
\[B \to A : \text{enc}(m, \text{pk}(A))\]

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

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

- $P_i(ski, pkr) \triangleq \nu n. \text{out}(c, \text{aenc}(n, pkr)). \text{in}(c, x). \text{if}(\text{adec}(x, ski) = n) \text{then SUCCESS}$

- $P_r(skr) \triangleq \text{in}(c, y). \text{let } pka = \text{fst}(y) \text{in. let } z = \text{adec}(y, skr) \text{in. out}(c, \text{aenc}(z, pka))$

- Allow the intruder to supply the other party the initiator talks to

- Allow the same agent to play either role; allow unboundedly many honest agents

- Can write this out more succinctly as follows:

  $$P_r \triangleq !\nu sk. ( !\text{in}(c, x_{pk}). P_i(sk, x_{pk}) \mid !P_r(sk) \mid \text{out}(c, pk(sk)) )$$
n should be secret to the initiator and the responder

Is there any session where the name established between the initiator and responder in that session can be deduced by the intruder?

\[ Q_i(n, ski, pkr) \triangleq \text{out}(c, \text{aenc}(n, pkr)). \text{in}(c, x). \text{if}(\text{adec}(x, ski) = n) \text{ then SUCCESS, and} \]
\[ P_i(ski, pkr) \triangleq \nu n. Q_i(n, ski, pkr) \]

\[ P^n(ski, pkr) = \nu n. (Q_i(n, ski, pkr) \mid (\text{in}(c, x). \text{if } x = n \text{ then event } \text{leak}(n) \text{ else } 0) \]

\[ \text{Pr} \triangleq \text{in}(c, x_{pk}). P^n(s_i, x_{pk}) \mid !\nu sk. (!\text{in}(c, x_{pk}). P_i(sk, x_{pk}) \mid !P_r(sk) \mid \text{out}(c, pk(sk))) \]
EXAMPLE 2: NS PUBLIC KEY

\[ A \rightarrow B : \text{enc}((A, n_a), \text{pk}(B)) \]
\[ B \rightarrow A : \text{enc}((n_a, n_b), \text{pk}(A)) \]
\[ A \rightarrow B : \text{enc}(n_b, \text{pk}(B)) \]
EXAMPLE 2: NS PUBLIC KEY

\[ A \rightarrow B : \text{enc}((A, n_a), \text{pk}(B)) \]
\[ B \rightarrow A : \text{enc}((n_a, n_b), \text{pk}(A)) \]
\[ A \rightarrow B : \text{enc}(n_b, \text{pk}(B)) \]

\begin{verbatim}
init(ski: skey, pkr: pkey) {
    new na: bytes;
    send(aenc((pk(ski), na), pkr));
    recv(x: bytes);
    let z = adec(x, ski) in
        if (fst(z) \neq na) error
        else send(aenc(snd(z), pkr));
}
\end{verbatim}

\begin{verbatim}
resp(pki: pkey, skr: skey) {
    recv(y: bytes);
    let (k, na) = adec(y, skr) in
        new nb: bytes;
        send(aenc((na, nb), k));
        recv(z: bytes);
        if (adec(z, skr) \neq nb) error;
}
\end{verbatim}
NS PUBLIC KEY

- Proposed by Roger Needham and Michael Schroeder in 1978.

- Requirement: At the end of an execution, the two agents should agree on the identity of their respective correspondent.

- Is there an attack?
NS PUBLIC KEY

- Proposed by Roger Needham and Michael Schroeder in 1978.
- Requirement: At the end of an execution, the two agents should agree on the identity of their respective correspondent.
- Is there an attack?
  - Yes! Found by Gavin Lowe in 1995. Different flavour of MitM.
CORRESPONDENCE: FORMALIZED

- \(e_0(\vec{t}_0) \triangleright e_1(\vec{t}_1)\) denotes the following correspondence: “if \(e_1(\vec{t}_1)\) occurred in a run, then \(e_0(\vec{t}_0)\) occurred earlier”

- A reduction sequence \(P_0 \xrightarrow{\gamma_1} P_1 \cdots \xrightarrow{\gamma_n} P_n\) satisfies a correspondence \(e_0(\vec{t}_0) \triangleright e_1(\vec{t}_1)\) iff for any \(\sigma\),

  whenever \(e_1(\vec{t}_1 \sigma)\) occurs in some \(P_i\), there is a \(j \leq i\) such that

  \(e_0(\vec{t}_0 \sigma)\) occurs in \(P_j\)

- A process \(P\) satisfies a correspondence property iff all reduction sequences starting from \(P\) satisfy it.
Consider a really simple voting protocol

Voter V encrypts their vote $v$ in C’s public key and sends it

C decrypts it and counts the vote

**Anonymity:** nobody but C should be able to find a link between V’s name and their vote

Is there an attack? What does it mean to find a link?
ANONYMITY

- Consider a situation where only V has voted, nobody else
- The intruder sees a single term going by on the channel
- The term is $\text{aenc}(v, \text{pk}(A))$
- Can the intruder make any judgements based on this one term?
ANONYMITY

- Assume the set of candidates is \( \{v_0, v_1\} \) (both public constants)

- What would differ between a situation where \( V \) voted for \( v_0 \) versus one where they voted for \( v_1 \)?
ANONYMITY

- Assume the set of candidates is \( \{v_0, v_1\} \) (both public constants)

- What would differ between a situation where V voted for \( v_0 \) versus one where they voted for \( v_1 \)?

- Frames \( \sigma_0 = [x \mapsto aenc(v_0, pk(A))] \) and \( \sigma_1 = [x \mapsto aenc(v_1, pk(A))] \)

- What recipe tells these frames apart?