02244 Logic for Security
Security Protocols
Symbolic Analysis: The Lazy Intruder

Sebastian Mödersheim

February 16, 2026

Dolev-Yao Closure: Summary

Dolev-Yao rules

$$\frac{}{M \vdash m} \text{ if } m \in M \text{ (Axiom)}$$

$$\frac{M \vdash m_1 \quad \ldots \quad M \vdash m_n}{M \vdash f(m_1, \ldots, m_n)} \text{ if } f/n \in \Sigma_p \text{ (Compose)}$$

$$\frac{M \vdash \langle m_1, m_2 \rangle}{M \vdash m_i} \text{ (Proj}_i\text{)} \qquad \frac{M \vdash \{\!|m|\!\}_k \quad M \vdash k}{M \vdash m} \text{ (DecSym)}$$

$$\frac{M \vdash \{m\}_k \quad M \vdash \mathrm{inv}(k)}{M \vdash m} \text{ (DecAsym)} \qquad \frac{M \vdash \{m\}_{\mathrm{inv}(k)}}{M \vdash m} \text{ (OpenSig)}$$

The compose rule is for all public functions Σ_p,
including $\{\!| \cdot |\!\}_{\cdot}$ $\quad \{\cdot\}_{\cdot}$ $\quad \langle \cdot, \cdot \rangle$

Automation

Goal: design (in pseudocode) a decision procedure for Dolev-Yao:

- Given a finite set M of messages (the intruder knowledge)
- and given a message m (the goal)
- Output whether $M \vdash m$ holds.
  - ★ additionally, in the positive case, give the proof.

Automating Dolev-Yao

Step 1: Composition only

Consider first the following simpler problem:

- $M \vdash_c m$ are those deductions where the intruder does not apply any analysis steps ("composition only"):

Composition Only

$$\frac{}{M \vdash_c m} \text{ if } m \in M \text{ (Axiom)}$$

$$\frac{M \vdash_c m_1 \quad \ldots \quad M \vdash_c m_n}{M \vdash_c f(m_1, \ldots, m_n)} \text{ if } f \in \Sigma_p \text{ (Compose)}$$

Example

$M = \{k_1, k_2, \{|m|\}_{h(k_1,k_2)}\}$ where $h \in \Sigma_p$

- $M \vdash_c h(k_1, k_2)$
- $M \nvdash_c m$
- $M \vdash m$

Negative Question

Can we thus prove also statements of the form $M \nvdash m$

. . . that a m cannot be derived from M?

Example

$$M = \{\, k_1, \{\!|m_1|\!\}_{k_1}, m_2, \{\!|m_3|\!\}_{k_2} \,\} \nvdash m_3$$

Negative Question

Can we thus prove also statements of the form $M \nvdash m$

. . . that a m cannot be derived from M?

Example

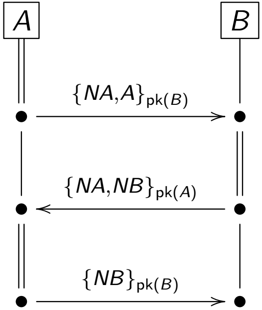
$$M = \{\, k_1, \{\!|m_1|\!\}_{k_1}, m_2, \{\!|m_3|\!\}_{k_2} \,\} \nvdash m_3$$

- Yes, due to completeness when our algorithm answers "no", we know there is no derivation for m.

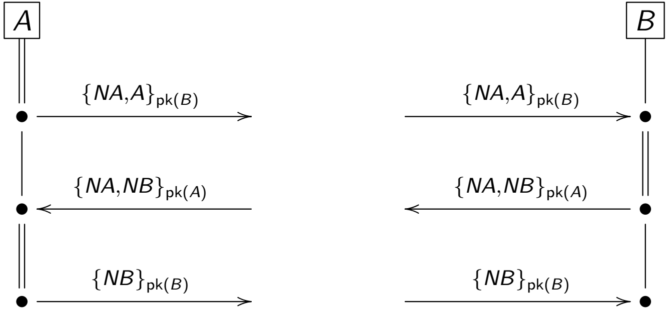
Needham-Schroeder Public-Key Protocol [1978]

```
Protocol: NSPK

Types: Agent A,B;
       Number NA,NB;
       Function pk,h

Knowledge: A: A,pk(A),inv(pk(A)),B,pk(B),h;
           B: B,pk(B),inv(pk(B)),A,pk(A),h
Actions:
A->B: {NA,A}(pk(B))  # A generates NA
B->A: {NA,NB}(pk(A)) # B generates NB
A->B: {NB}(pk(B))

Goals:
h(NA,NB) secret between A,B
```

NSPK as A Message Sequence Chart

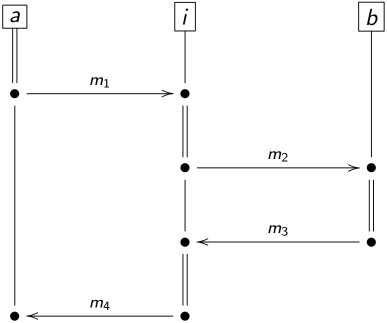


NSPK as Roles / Strands



- For each Role of the protocol, a program that sends and receives messages (over possibly insecure network)
- Strand: concrete execution of a role: all variables (here A, B, NA, NB) instantiated with concrete values
  - ★ or a prefix thereof (an agent might not finish)

Attacks

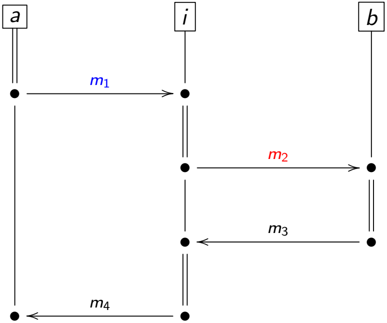


An attack is a strand space where the following conditions are met:

- Messages sent by honest agents are received by i
- Messages received by honest agents are sent by i who can compose the message from the messages he has received so far.

- The successful completion violates a goal of the protocol.

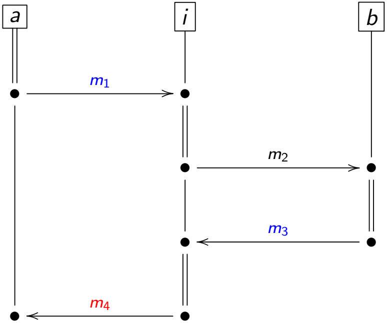
Attacks



An attack is a strand space where the following conditions are met:

- Messages sent by honest agents are received by i
- Messages received by honest agents are sent by i who can compose the message from the messages he has received so far.
  - ★ In the example: $\{m_1\} \vdash m_2$
- The successful completion violates a goal of the protocol.

Attacks



An attack is a strand space where the following conditions are met:

- Messages sent by honest agents are received by i
- Messages received by honest agents are sent by i who can compose the message from the messages he has received so far.
  - ★ In the example: $\{m_1\} \vdash m_2$ and $\{m_1, m_3\} \vdash m_4$.
- The successful completion violates a goal of the protocol.

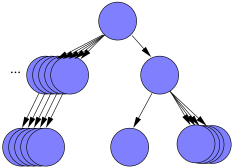
Infinite State Space

Problem 1: at any time, any number of people can run the protocol in parallel. (Think of TLS...)

- For now we bound the number of sessions: only finitely many strands of honest agents
- Later: how to verify for unbounded sessions

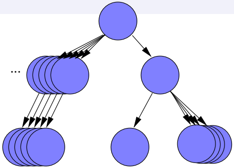
Problem 2: at any time the intruder has an inifinite choice of message they can construct and send to an agent.

- We will solve this problem with a constraint approach: the lazy intruder.



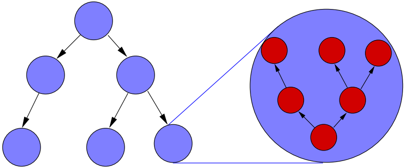
Lazy Intruder: Overview

Even for bounded sessions we have an infinite tree of reachable states, i.e., how the intruder can interact with honest agents.



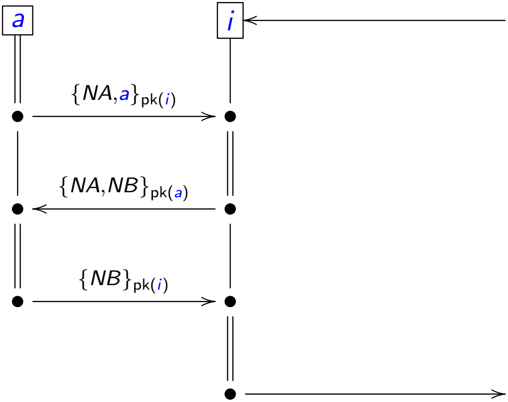
Idea: symbolic states

- Each state is an attack scenario: a sequence of interactions with honest agents, leaving undetermined what exactly the intruder sends.



- Then each state is a constraint solving problem: "Can the intruder generate all messages that the attack scenario has?"
- This will be a backward search: we start at the complete attack scenario and try to see how the intruder could have constructed this.

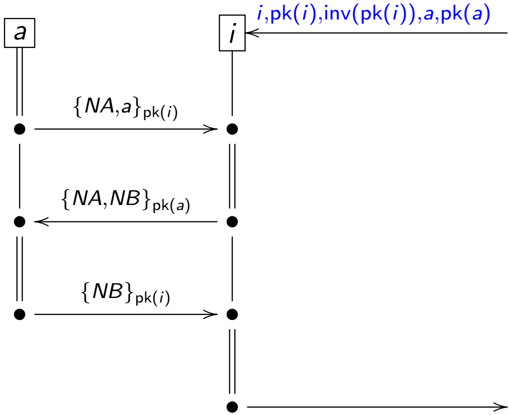
Example: Can the Intruder Play a Protocol Role?



Example:

- NSPK with $A = a$ and $B = i$

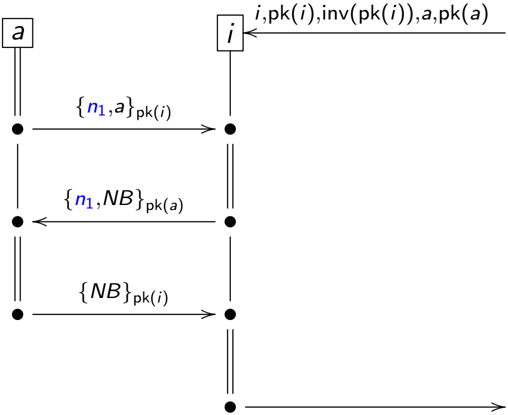
Example: Can the Intruder Play a Protocol Role?



Example:

- NSPK with $A = a$ and $B = i$
- i needs corrsp. knowledge of role B: $i, \mathrm{pk}(i), \mathrm{inv}(\mathrm{pk}(i)), a, \mathrm{pk}(a)$

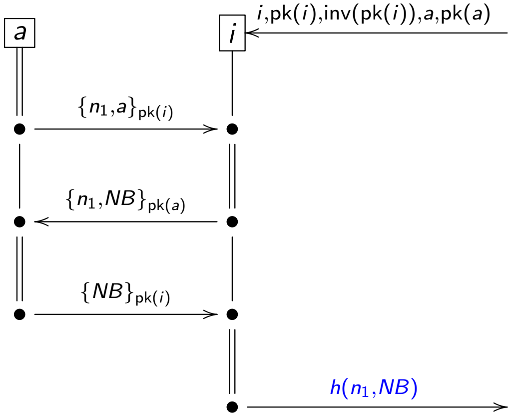
Example: Can the Intruder Play a Protocol Role?



Example:

- NSPK with $A = a$ and $B = i$
- i needs corrsp. knowledge of role B: $i, \mathrm{pk}(i), \mathrm{inv}(\mathrm{pk}(i)), a, \mathrm{pk}(a)$
- a uses a fresh $NA = n_1$.

Example: Can the Intruder Play a Protocol Role?



Example:

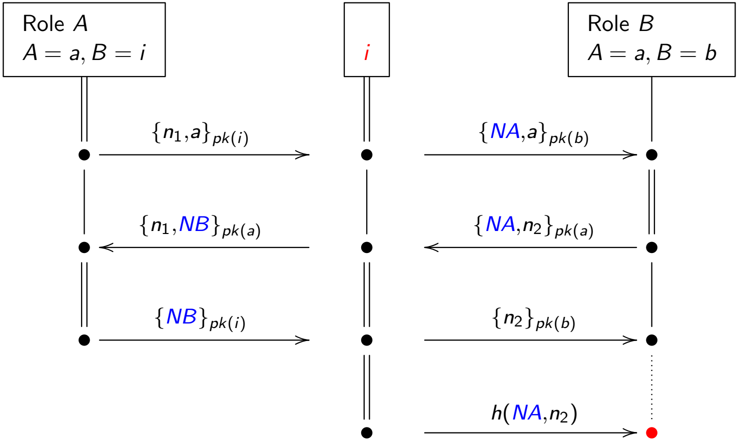
- NSPK with $A = a$ and $B = i$
- i needs corrsp. knowledge of role B: $i, \mathrm{pk}(i), \mathrm{inv}(\mathrm{pk}(i)), a, \mathrm{pk}(a)$
- a uses a fresh $NA = n_1$.
- Afterwards, i should be able to construct the shared key $h(NA, NB) = h(n_1, NB)$

The Lazy Intruder Attacking

- So far, we have only looked at how the intruder can play a role of the protocol under his real name.
  - We have with this a check that the role is even executable: the initial knowledge is sufficient to perform all steps of a "normal" protocol execution.
- We want to use this technique for checking an attack instead.
- For the NSPK we consider the following attack scenario:
  - a in role A wants to talk to i in role B
    - She uses fresh n_1 as NA in her strand
  - b in role B is willing to talk to a in role A;
    - He uses fresh n_2 as NB in his strand
- We want to show that $b : B$ can be attacked: the intruder can find out the session key that $b : B$ believes to have with $a : A$.

The Lazy Intruder Attacking

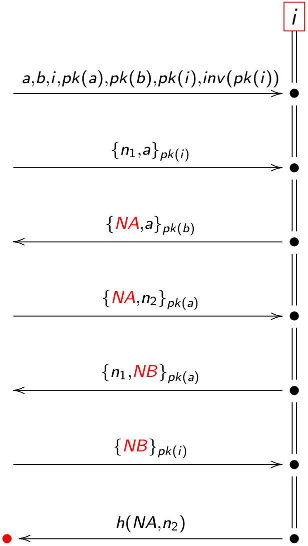
Scenario



The challenge labeled •:

- Can the intruder produce the secret session key $h(NA, n2)$?

Choose and Check



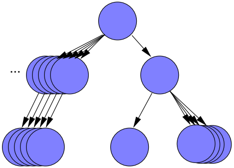
- For simplicity we display here only outgoing and incoming messages of the intruder.

Needham-Schroeder-Lowe [1996]

```
Protocol: NSL

Types: Agent A,B;
       Number NA,NB;
       Function pk,h

Knowledge: A: A,pk(A),inv(pk(A)),B,pk(B),h;
           B: B,pk(B),inv(pk(B)),A,pk(A),h
Actions:
A->B: {NA,A}(pk(B))
B->A: {NA,NB,B}(pk(A))
      # Inserted B's name into the message
A->B: {NB}(pk(B))

Goals:
h(NA,NB) secret between A,B
```

Lazy Intruder: Summary

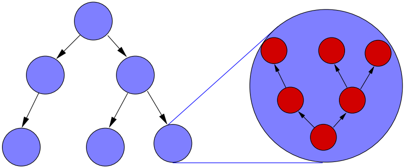
- Without the lazy approach, we would get an infinite search tree because the intruder has often an infinite choice of messages to send.



- We avoid this by using the lazy intruder:

  Layer 1: a symbolic search tree

  Layer 2: constraint solving



Lazy Intruder: Summary

Constraint Solving

Go step by step through the constraint.

- For incoming messages: can a decryption rule be applied?
  If so, add the decrypted message also as an incoming message
  - When the key contains variables this can be handled by adding the key as an outgoing message, i.e., require that the intruder can produce it.
- For outgoing messages
  - If it is a variable: be lazy for now.
    - If it gets replaced, you need to come back here.
  - Otherwise: check all following possibilities (backtracking!):
    - Compose: can a compose rule be applied?
    - Axiom: can the axiom rule be applied?
      This may require instantiating variables

Lazy Intruder: Summary

Theorem (Rusinowitch & Turuani 2001)

Protocol insecurity for a bounded number of sessions is NP-complete.

Proof Sketch.

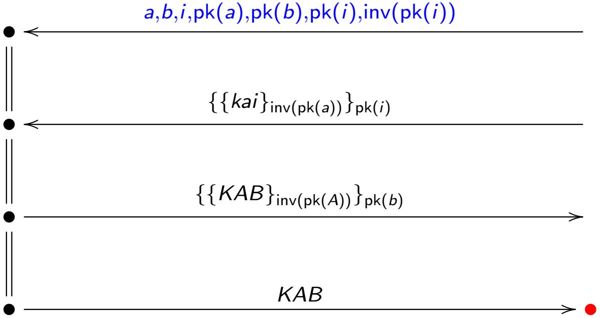
In NP: Guess a symbolic attack trace for the given strands and a sequence of reduction steps for the resulting constraints. Check that this sequence of reduction steps solves the constraint.

NP-hard: Polynomial reduction for boolean formulae to security protocols such that formula satisfiable iff protocol has an attack. □

Relevant Research Papers

- David Basin, Sebastian Mödersheim, and Luca Viganò. *OFMC: A symbolic model checker for security protocols.* International Journal of Information Security, 4(3), 2005.
- Gavin Lowe. *Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR*. Software Concepts Tools, 17(3), 1996.
- Jonathan K. Millen and Vitaly Shmatikov. *Constraint solving for bounded-process cryptographic protocol analysis*. Computer and Communications Security, 2001,
- Roger Needham and Michael Schroeder. *Using Encryption for Authentication in Large Networks of Computers.* Communications of the ACM, 21(12), 1978.
- Michaël Rusinowitch and Mathieu Turuani. *Protocol Insecurity with Finite Number of Sessions is NP-complete.* Computer Security Foundations Workshop, 2001.

Exercise



Exercise: show that this constraint has both

- a solution where $A = i$ (i.e., a normal execution)
- a solution where $A = a$ (i.e., an attack)