Protocols VI: Deductive Analysis Methods (Part I)

Christoph Sprenger and Sebastian Mödersheim
Department of Computer Science
ETH Zurich

FMSEC Module 7, v.2
November 2, 2009

Outline

Outline

Computer-Aided Protocol Analysis

- The operational semantics of protocols provides a basis for rigorous protocol analysis.
- Challenging since the general problem is undecidable (Module 5).
- Two possible approaches: algorithmic and deductive.

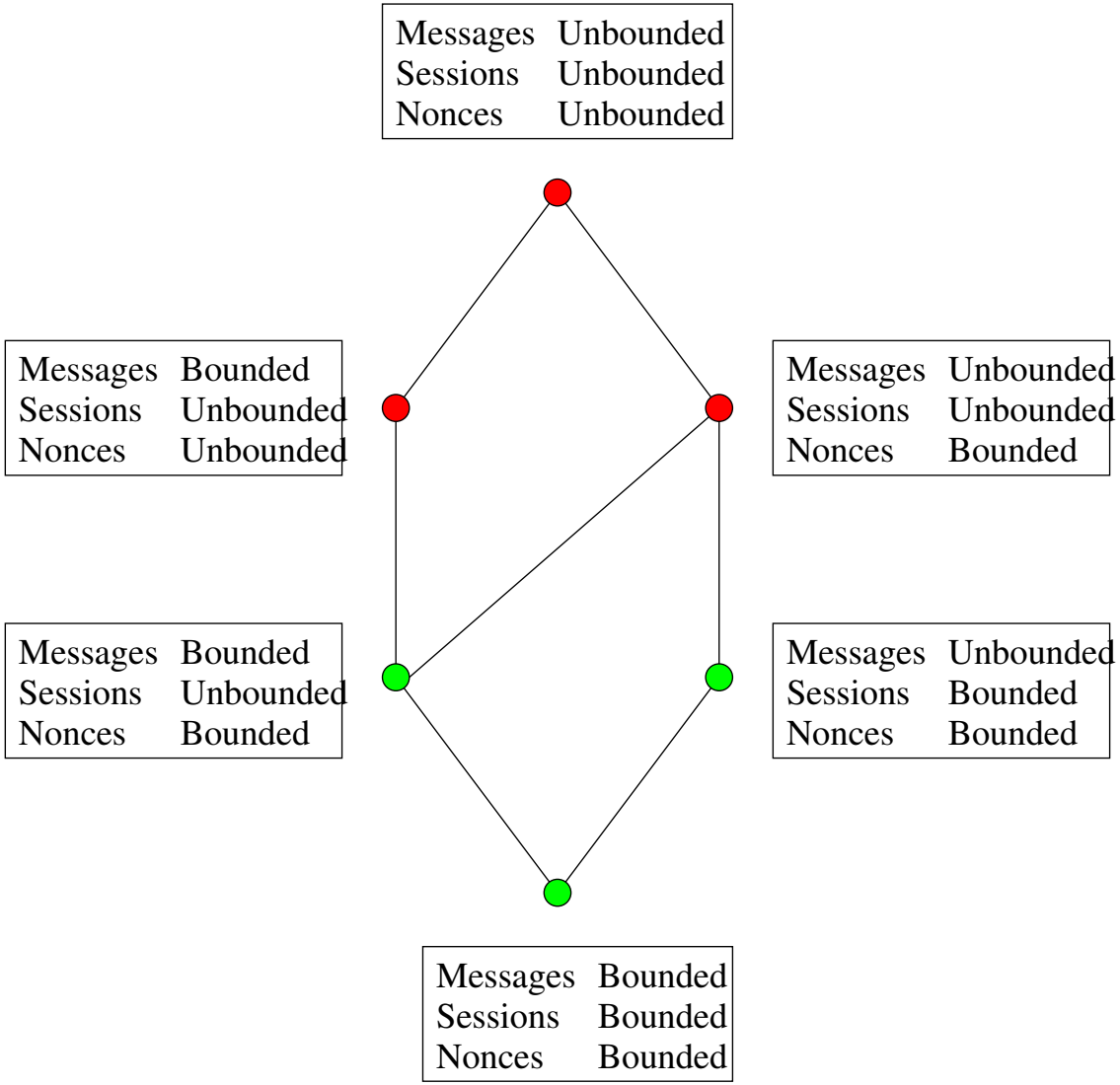
Algorithmic Analysis

- Using a decision or semi-decision procedure, aka model checker.
- Sometimes these are able to prove correctness, but not in general.
- Requires finiteness bounds if termination must be guaranteed.

Deductive Analysis

- Produce a proof of correctness using some form of induction.
- Usually supported by a semi-automatic interactive theorem prover.
- Proofs may also be done by hand, but this is extremely error-prone.

Decidability Issues



Deductive Methods

Motivations

Generality

- Deductive methods can handle all infinite state spaces.
- No need for finiteness bounds (e.g, on messages, nonces, sessions).

Expressiveness

- Flexible platform for experimentation.
- Possibility to prove meta-results about a model (vs within a model)
- Price: loss of automation, proofs generally require user interaction.

Insights

- Modeling and proving process yields insights into the problem.
- Insights may lead to simplifications of model and/or properties.
- Simplifications often foster an increased proof automation.

Deductive and Algorithmic Approaches

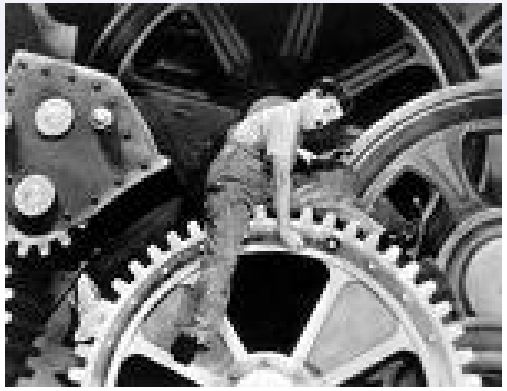
Deductive methods

- + certification: proof of protocol correctness
- + flexibility: fosters experimentation with different protocol models
- + coverage: proofs cover entire state space, even if infinite
- – user interaction with prover requires considerable expertise
- – no attacks in case of failing properties, instead extract information from analysis of failed proof state

Algorithmic methods

- + automatic: no or little user interaction
- + produce counterexample trace in case of attack
- + can sometimes prove correctness, even in infinite-state case
- – flexibility: often fixed model and set of properties
- – coverage: cannot prove correctness in general; also: finiteness restrictions needed if termination is required.

Modern Times



The formerly clear-cut border between algorithmic and deductive methods becomes more and more blurred.

Example 1: Lazy intruder

- Model checking was traditionally restricted to finite-state models.
- Symbolic methods like the lazy intruder used in OFMC can deal with infinite state spaces, e.g., due to unboundedness of messages.

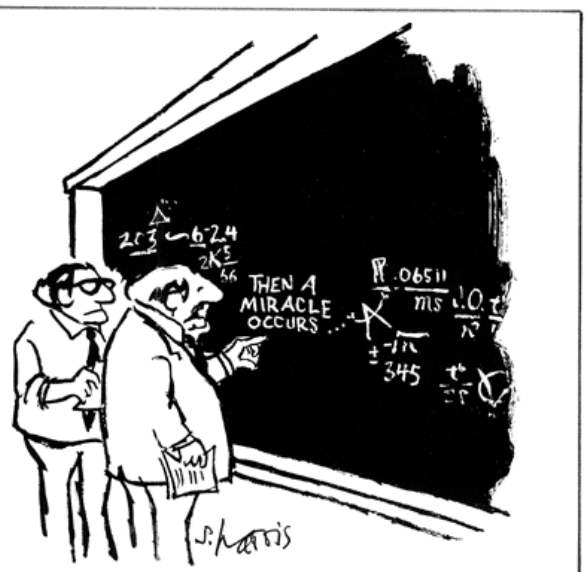
Example 2: TAPS system

- Ernie Cohen's TAPS system automatically generates a set of invariants from a protocol specification.
- These are proved by an automatic theorem prover for 1st order logic.
- This is a deductive system with little or no user interaction.

Example 3: Certifying model checkers

- There are recent prototypes of the model checkers OFMC and Scyther that generate proof certificates.
- These certificates can be verified indepedently and without user interaction by the theorem prover Isabelle/HOL.

What is a Proof?



"I think you should be more explicit here in step two."

Informally

- Purpose: convince others (and yourself) of the truth of a mathematical statement.
- A proof should be sufficiently detailed to be convincing (without hand-waving).

More Formally

- A derivation system is a set of rules of the form:

$$\frac{P_1 \quad \cdots \quad P_n}{C}$$

  where P_i are called the premises and C the conclusion of the rule.
- A rule without premises is called an axiom.
- A derivation of C is a derivation tree with C at its root and only axioms at the leaves.
- For derivation system of a logic $\mathcal{L}$: a derivation of C is called a proof of C, which promotes the statement C to a theorem of $\mathcal{L}$.

Natural Deduction Proofs

Definition (Sequent syntax and semantics)

$$H_1, \ldots, H_m \vdash Q$$

where the H_i, called the hypotheses and Q, called the consequent of the sequent, are logical formulas. The intended meaning is expressed by

$$\forall x_1 \cdots x_k.\, H_1 \wedge \cdots \wedge H_m \Rightarrow Q$$

where $x_1 \cdots x_k$ are the free variables occuring in $H_1, \ldots, H_m$ and Q.

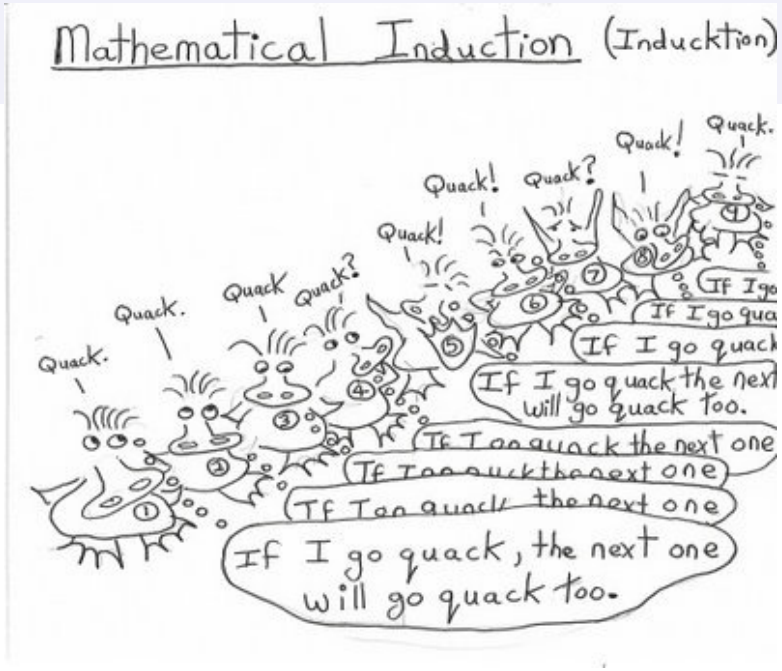
Example (Some FOL proof rules)

$$\frac{H, A \vdash B}{H \vdash A \Rightarrow B}\ (\Rightarrow\text{-I}) \qquad \frac{H \vdash A \Rightarrow B \qquad H \vdash A}{H \vdash B}\ (\Rightarrow\text{-E})$$

$$\frac{H \vdash P}{H \vdash \forall x.P}\ (\forall\text{-I}) \qquad \frac{H \vdash \forall x.P}{H \vdash P[x \mapsto t]}\ (\forall\text{-E})$$

Proviso in ($\forall$-I): x may not occur freely in H.

What is Induction?



Induction is a technique to define and reason about infinite sets of finite objects.

Examples abound!

Data types (natural numbers, lists, trees, messages), set of even numbers, less-or-equal relation, transitive closure, definition of a formal language in BNF, the set of traces of a security protocol, ...

Characteristics

- Inductive definitions can be given as a set of rules.
- The definition must be well-founded, i.e., there must be a base case.
- The definition must be monotonic; the rules must avoid negative premises involving the set (or predicate, relation) being defined.
- A derived induction rule performs induction on the defining rules (this is called rule induction, a generalization of structural induction).

Definition and Proof by Induction (1)

Example (Natural numbers)

Definition:

$$\frac{}{0 \in nat} \qquad \frac{x \in nat}{s(x) \in nat}$$

Induction rule:

$$\frac{P(0) \qquad \forall x \in nat.\ P(x) \Rightarrow P(s(x))}{\forall n \in nat.\ P(n)}$$

- The constants 0 and s are the constructors of nat.
- The induction rule states that the property P to be proved is preserved by each constructor.
- Similar definitions for other data types such as lists and binary trees.

Definition and Proof by Induction (2)

Example (An "odd" process)

Definition:

$$\frac{}{[\,] \in O} \qquad \frac{odd(x)}{[x] \in O} \qquad \frac{tr \in O \qquad x \in set(tr)}{tr \cdot (2x+1) \in O}$$

Induction rule:

$$\frac{P([\,]) \qquad \forall x.\, odd(x) \Rightarrow P([x]) \qquad \forall tr, x.\, tr \in O \wedge P(tr) \wedge x \in set(tr) \Rightarrow P(tr \cdot (2x+1))}{\forall tr \in O.\, P(tr)}$$

- The defined process is highly non-deterministic: [] has infinitely many extensions and all other traces tr have $|set(tr)|$ extensions.
- The induction rule contains one premise for each construction rule.
- Similar to list induction, but the induction is only over lists in O.

Definition and Proof by Induction (3)

Theorem (O's traces contain only odd numbers)

$\forall tr \in O.\ \forall x.\ x \in set(tr) \Rightarrow odd(x)$

Proof.

Applying the induction rule with $P(tr) = \forall x.\ x \in set(tr) \Rightarrow odd(x)$ yields:

1. $P([])$. This is trivial, since $x \in set([])$ is false.
2. $odd(x) \vdash P([x])$. By hypothesis $odd(x)$, since $set([x]) = \{x\}$.
3. $tr \in T, P(tr), x \in set(tr) \vdash P(tr \cdot (2x+1))$. First-order reasoning on the consequent using $\Rightarrow$-I and $\forall$-I and simplification of *set* yields

$$tr \in T, P(tr), x \in set(tr), y \in \{2x+1\} \cup set(tr) \vdash odd(y)$$

Union splits into two cases: (i) $y = 2x+1$ and (ii) $y \in set(tr)$.
(i) We derive $odd(2x+1)$ by arithmetical reasoning as required. (This holds for all natural numbers x.)
(ii) Follows directly by application of the induction hypothesis with $[x \mapsto y]$. $\square$

Definition and Proof by Induction (4)

Example (Reachability)

Definition:

$$\frac{s \in I}{s \in reach(I, R)} \qquad \frac{s \in reach(I, R) \quad (s, t) \in R}{t \in reach(I, R)}$$

Induction rule:

$$\frac{\forall s \in I.\ P(s) \qquad \forall s, t.\ s \in reach(I, R) \wedge P(s) \wedge (s, t) \in R \Rightarrow P(t)}{\forall s \in reach(I, R).\ P(s)}$$

- The set $reach(I, R)$ (of states) is parametrized by a set I (of initial states) and a (transition) relation R.
- Induction rule, general reading: P is preserved by each defining rule, i.e., by all different ways to construct elements in $reach(I, R)$.
- In particular: An invariant proof rule. It requires that the invariant P holds on initial states in I and is preserved by R-transitions.

Outline

Paulson's Inductive Approach

Main Ideas

Inductive Protocol Definition

- Define a protocol as the set of event traces that it produces.
- The set of protocol traces can be defined inductively: it contains the empty trace (base case) and rules determining how a given trace can be extended by adding a new event.
- There is one such rule for each message transmitted by the protocol and one rule for the intruder.

Inductive Proofs of Security Properties

- Protocol properties are expressed as predicates on traces.
- The proof that all traces of a protocol satisfy a given property proceeds by induction on the rules defining the protocol.
- Failure to prove a property either indicates the need for additional invariants, the need for a different proof idea, or points to an attack.

Paulson's Inductive Approach

Formalization in Isabelle/HOL

- Isabelle is a generic theorem prover with powerful user-configurable automatic proof tools including a simplifier and a tableau reasoner.
- Isabelle/HOL stands for Isabelle's instantiation to higher-order logic.
- Slogan: "HOL = functional programming + quantifiers".
- Isabelle/HOL supports inductive definitions and proofs. For each inductive definition an induction rule is automatically generated.
- Proof tactics are used to construct proofs. These include single-step "manual" tactics and tactics invoking the automatic proof tools.

Remarks

- Our aim is to convey the main concepts and techniques of Paulson's approach. We therefore take the freedom to deviate in several respects from Paulson (e.g., syntax, message algebra).
- Also, the fact that Paulson used HOL (e.g., instead of first-order logic) is not so important to us here.

Messages and Events

Messages

- We work with the previously defined free message algebra.
- We assume typed messages and atomic keys.
- The DY closure is replaced by *synth* and *analz*, defining message construction and analysis separately.

Events A trace is a sequence of events of the form

$$A \rightarrow B : M$$

- The agents A and B are the sender and the intended receiver of the message M.
- The event $A \rightarrow B : M$ approximately corresponds to $\mathrm{snd}(A, B, M)$ in our previous models, but here the intruder does not learn A and B.
- In Paulson's (original) model no explicit receive events appear in the trace. (These were added later, see, e.g., Bella's book/PhD thesis.)

Intruder Deduction

Definition (Composition - *synth*)

$$\frac{t \in H}{t \in synth(H)}\ \text{Inject}$$

$$\frac{t_1 \in synth(M) \quad \cdots \quad t_n \in synth(H)}{f(t_1, \ldots, t_n) \in synth(M)}\ \text{Composition } (f \in \Sigma_p)$$

Definition (Decomposition - *analz*)

$$\frac{t \in H}{t \in analz(H)}\ \text{Inject}$$

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

$$\frac{\{m\}_k \in analz(H) \quad \text{inv}(k) \in analz(H)}{m \in analz(H)}\ \text{DecAsym} \qquad \frac{\{\!|m|\!\}_{\text{inv}(k)} \in analz(H)}{m \in analz(H)}\ \text{OpenSig}$$

Similar to $\mathcal{DY}_C(\cdot)$ and $\mathcal{DY}_D(\cdot)$ from Module 5, but without composed keys. Hence, there is no dependency of *analz*$(\cdot)$ on *synth*$(\cdot)$.

Some Properties of *analz*

Theorem (General properties)

- $analz(\emptyset) = \emptyset$ *(emptiness)*
- $H \subseteq analz(H)$ *(expansion)*
- $analz(analz(H)) = analz(H)$ *(closure)*
- $G \subseteq H \vdash analz(G) \subseteq analz(H)$ *(monotonicity)*
- $analz(G) \cup analz(H) \subseteq analz(G \cup H)$ *(union)*

Theorem (Message insertion lemmas)

- $analz(\{N\} \cup H) = \{N\} \cup analz(H)$ *(for nonce N; analz-nonce)*
- $analz(\{\langle m_1, m_2 \rangle\} \cup H) = \{\langle m_1, m_2 \rangle\} \cup analz(\{m_1, m_2\} \cup H)$
- $analz(\{\{M\}_K\} \cup H) =$
  $\begin{cases} \{\ \{M\}_K\ \} \cup analz(\{M\} \cup H) & \textit{if } \mathrm{inv}(K) \in analz(H) \\ \{\ \{M\}_K\ \} \cup analz(H) & \textit{otherwise} \end{cases}$ *(analz-crypt)*

Many of these results are declared to the automatic proof tools of Isabelle and then applied automatically (e.g., rewriting of equations by simplifier).

Inductive Definition of Protocols

Protocol Events

Example (NSL protocol, 2nd message: $B \to A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}$)

The inductive definition of the NSL protocol defines the set of traces *nsl*. Here is the rule for the second message:

$$\frac{tr \in nsl \qquad A' \to B\colon \{N_A, A\}_{\mathrm{pk}(B)} \in set(tr) \qquad N_B \notin used(tr)}{tr \cdot (B \to A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}) \in nsl}\ \text{nsl2}$$

- 2nd premise: This premise acts as an implicit receive event. Note: the sender is A' instead of the expected A, since the sender is not authenticated. In particular, we may have $A' = \mathrm{i}$!
- 3rd premise: $used(tr)$ denotes the set of nonces occurring on trace tr, hence the premise requires that N_B is a fresh nonce.
- The conclusion allows us to extend the trace tr with the event $B \to A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}$, sending the 2nd protocol message.

Note: An arbitrary number of replies with different nonces might be sent for a given $A' \to B\colon \{N_A, A\}_{\mathrm{pk}(B)} \in set(tr)$. The premises do not avoid this.

Inductive Definition of Protocols

Intruder

Definition (Fake rule)

The definition of each protocol *proto* contains the intruder rule:

$$\frac{tr \in proto \qquad X \in synth(analz(IK(tr))))}{tr \cdot (\mathrm{i} \rightarrow B\colon X) \in proto}\ \text{fake}$$

Remarks

- The current intruder knowledge is defined by

$$IK(tr) = IK_0 \cup \{M \mid A \rightarrow B\colon M\}$$

- This rule is part of each protocol definition, but is the same for all protocols (substitute actual name for *proto*).

Inductive Definition of Protocols

Example (NSL protocol)

Here is the inductive definition of the entire NSL protocol.

$$\frac{}{[] \in nsl}\ \text{empty}$$

$$\frac{tr \in nsl \qquad N_A \notin used(tr)}{tr \cdot (A \rightarrow B\colon \{N_A, A\}_{\mathrm{pk}(B)}) \in nsl}\ \text{nsl1}$$

$$\frac{tr \in nsl \qquad A' \rightarrow B\colon \{N_A, A\}_{\mathrm{pk}(B)} \in set(tr) \qquad N_B \notin used(tr)}{tr \cdot (B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}) \in nsl}\ \text{nsl2}$$

$$\frac{tr \in nsl \qquad \begin{array}{c}(A \rightarrow B\colon \{N_A, A\}_{\mathrm{pk}(B)}) \in set(tr)\\ (B' \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}) \in set(tr)\end{array}}{tr \cdot (A \rightarrow B\colon \{N_B\}_{\mathrm{pk}(B)}) \in nsl}\ \text{nsl3}$$

$$\frac{tr \in nsl \qquad X \in synth(analz(IK(tr)))}{tr \cdot (\mathrm{i} \rightarrow B\colon X) \in nsl}\ \text{fake}$$

Inductive Proofs of Protocol Properties

Overview

Expressing properties

- Statement that a property P holds for all traces of a protocol *proto*:

$$H \vdash \forall tr \in proto.\ P(tr)$$

- H is a list of hypotheses (e.g., $A \neq i, B \neq i$).

Inductive Proofs

- The inductive definition of the protocol traces gives rises to a corresponding induction rule.
- The induction rule contains one case for each defining rule, stating that the extension of a trace using the rule preserves the property.
- This rule is automatically generated (and proved) by Isabelle/HOL as a result of the definition.

Specification of Properties

Note: Definitions are protocol-dependent (since not based on signals).

Example (Secrecy of N_B in NSL, from B's perspective)

$$A \neq i, B \neq i$$
$$\vdash \forall tr \in nsl.$$
$$B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr) \Rightarrow N_B \notin analz(IK(tr)))$$

- We use $analz(\cdot)$ to specify secrecy (ok for atomic secrets).

Example (B authenticates A in NSL)

$$A \neq i, B \neq i$$
$$\vdash \forall tr \in nsl.$$
$$B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr) \wedge A' \rightarrow B\colon \{N_B\}_{\mathrm{pk}(B)} \in set(tr)$$
$$\Rightarrow A \rightarrow B\colon \{N_B\}_{\mathrm{pk}(B)} \in set(tr)$$

- If B has sent nsl2 and received nsl3 from some A' then A sent nsl3.
- Corresponds to non-injective agreement for B with A on $\{N_B\}_{\mathrm{pk}(B)}$; injective agreement is not achievable in this formalization of NSL.

Protocol Induction Rules

Example (NSL protocol, 2nd message)

Definition of nsl2:

$$\frac{tr \in nsl \qquad A' \rightarrow B\colon \{N_A, A\}_{\mathrm{pk}(B)} \in set(tr) \qquad N_B \notin used(tr)}{tr \cdot (B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}) \in nsl}\ \text{nsl2}$$

Corresponding premise of induction rule:

$$\begin{array}{l}\forall A, A', B, N_A, N_B, tr.\\ \quad tr \in nsl \wedge P(tr) \ \wedge\ A' \rightarrow B\colon \{N_A, A\}_{\mathrm{pk}(B)} \in set(tr) \ \wedge\ N_B \notin used(tr)\\ \quad \Rightarrow P(tr \cdot (B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}))\end{array}$$

This premise requires that we show:

- if the premises of rule nsl2 hold for tr (and A, A', B, N_A, N_B), and
- tr satisfies the property P (induction hypothesis),
- then the P also holds for tr extended with $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}$.

Protocol Induction Rules

Example (NSL induction rule)

(empty) $P([])$

(nsl1) $\forall A, B, N_A, tr.\ tr \in nsl \wedge P(tr) \wedge N_A \notin used(tr)$
$\Rightarrow P(tr \cdot (A \rightarrow B\colon \{N_A, A\}_{pk(B)}))$

(nsl2) $\forall A, A', B, N_A, N_B, tr.\ tr \in nsl \wedge P(tr) \wedge$
$A' \rightarrow B\colon \{N_A, A\}_{pk(B)} \in set(tr) \wedge N_B \notin used(tr)$
$\Rightarrow P(tr \cdot (B \rightarrow A\colon \{N_A, N_B, B\}_{pk(A)}))$

(nsl3) $\forall A, B, B', N_A, N_B, tr.\ tr \in nsl \wedge P(tr) \wedge$
$(A \rightarrow B\colon \{N_A, A\}_{pk(B)}) \in set(tr) \wedge$
$(B' \rightarrow A\colon \{N_A, N_B, B\}_{pk(A)}) \in set(tr)$
$\Rightarrow P(tr \cdot (A \rightarrow B\colon \{N_B\}_{pk(B)}) \in nsl)$

(fake) $\forall B, X, tr.\ tr \in nsl \wedge P(tr) \wedge X \in synth(analz(IK(tr)))$
$\Rightarrow P(tr \cdot (\mathrm{i} \rightarrow B\colon X))$

$\vdash$

$\forall tr \in nsl.\ P(tr)$

Reasoning about Protocol Steps

Summary of (typical) lemmas used in nsl2 case of secrecy proof

Messages (free algebra, typed!) and events

- Constructor injectiveness, e.g., $\{M\}_K = \{M'\}_{K'} \Rightarrow M = M' \wedge K = K'$.
- Constructor distinctness, e.g., $\langle N_A, A\rangle \neq \{N_D\}_{\mathsf{pk}(D)}$.
- Lemmas for rewriting $\mathit{analz}(\{m\} \cup H)$ for various forms of m.
- Simplifying equations for $\mathit{set}(\mathit{tr} \cdot e)$ and $\mathit{IK}(\mathit{tr} \cdot e)$.
- Events and messages: $A \rightarrow B\colon m \in \mathit{set}(\mathit{tr}) \Rightarrow m \in \mathit{parts}(\mathit{IK}(\mathit{tr}))$
  Here, $\mathit{parts}(H)$ closes the message set H under submessages.

These are general results that are applicable in most protocol proofs.

Auxiliary invariants of NSL

- secrecy of long-term keys: $\mathsf{inv}(\mathsf{pk}(D)) \in \mathit{analz}(\mathit{IK}(\mathit{tr})) \Leftrightarrow D = i$.
- unicity of N_B in nsl2: assuming N_B is secret we have
  $\{N_A, N_B, B\}_{\mathsf{pk}(A)} \wedge \{N'_A, N_B, B'\}_{\mathsf{pk}(A')} \Rightarrow A = A' \wedge B = B' \wedge N_A = N'_A$.

These are invariants of NSL, proved prior to the proof of secrecy of N_B.

Proof automation: Many lemmas are declared to be applied automatically by Isabelle/HOL. The proof script for the entire secrecy proof fits on two lines!

Proof of Secrecy of N_B in NSL

Example (Secrecy of N_B for B – initial goal)

$$A \neq i, B \neq i \vdash \forall tr \in nsl.\ Secret(A, B, N_A, N_B, tr)$$

where

$$Secret(A, B, N_A, N_B, tr) \equiv$$
$$B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr) \Rightarrow N_B \notin analz(IK(tr))$$

- The "original" theorem statement is

  $$\forall A, B, N_A, N_B, tr.\ A \neq i \wedge B \neq i \wedge tr \in nsl \Rightarrow Secret(A, B, N_A, N_B, tr)$$

  The version above is equivalent, but prepared for induction.
- Goal: Simplify the proof by making the induction hypothesis only as general as required for the proof to work.
- **Next:** apply NSL induction rule; we will consider the case for nsl3.

Proof of Secrecy of N_B in NSL (nsl3.1)

Example (Secrecy of N_B – induction step for nsl3)

H1. $A \neq i, B \neq i, tr \in nsl$,
H2. $Secret(A, B, N_A, N_B, tr)$,
H3. $C \rightarrow D\colon \{N_C, C\}_{\mathrm{pk}(D)} \in set(tr)$,
H4. $D' \rightarrow C\colon \{N_C, N_D, D\}_{\mathrm{pk}(C)} \in set(tr)$
$\vdash$
$Secret(A, B, N_A, N_B, tr \cdot (C \rightarrow D\colon \{N_D\}_{\mathrm{pk}(D)})$

- Above: subgoal for (nsl3) produced by the application of the nsl induction rule.
- **Next:** unfold the definition of *Secret*.

Proof of Secrecy of N_B in NSL (nsl3.2)

Example (Secrecy of N_B – after unfolding *Secret*)

H1. $A \neq i, B \neq i, tr \in nsl,$
H2. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr) \Rightarrow N_B \notin analz(IK(tr)),$
H3. $C \rightarrow D\colon \{N_C, C\}_{\mathrm{pk}(D)} \in set(tr),$
H4. $D' \rightarrow C\colon \{N_C, N_D, D\}_{\mathrm{pk}(C)} \in set(tr)$
$\vdash$
$B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr \cdot (C \rightarrow D\colon \{N_D\}_{\mathrm{pk}(D)}))$
$\Rightarrow N_B \notin analz(IK(tr \cdot (C \rightarrow D\colon \{N_D\}_{\mathrm{pk}(D)})))$

Next: Propositional reasoning ($\Rightarrow$-intro/elim, $\neg$-intro) yields two cases:

1. $(B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)}) \notin set(tr)$
   Message not yet sent, solved by contradiction. Done.
2. $N_B \notin analz(IK(tr))$
   Nonce N_B secret; continued on next slides.

Proof of Secrecy of N_B in NSL (nsl3.4)

Example (Secrecy of N_B – case 2: $N_B \notin analz(IK(tr))$)

H1. $A \neq i, B \neq i, tr \in nsl$,
H2. $N_B \notin analz(IK(tr))$,
H3. $C \rightarrow D\colon \{N_C, C\}_{pk(D)} \in set(tr)$,
H4. $D' \rightarrow C\colon \{N_C, N_D, D\}_{pk(C)} \in set(tr)$
H5. $B \rightarrow A\colon \{N_A, N_B, B\}_{pk(A)} \in set(tr \cdot (C \rightarrow D\colon \{N_D\}_{pk(D)}))$
H6. $N_B \in analz(IK(tr \cdot (C \rightarrow D\colon \{N_D\}_{pk(D)})))$
$\vdash \bot$

- **Next in H5:** rewrite equation $set(tr \cdot e) = \{e\} \cup set(tr)$ and then simplify $f \in set(tr \cdot e)$ to $f \in set(tr)$, since $e \neq f$; i.e., H5 becomes

$$B \rightarrow A\colon \{N_A, N_B, B\}_{pk(A)} \in set(tr)$$

- **Next in H6:** rewrite equation $IK(tr \cdot (A \rightarrow B\colon M)) = \{M\} \cup IK(tr)$.

Proof of Secrecy of N_B in NSL (nsl3.5)

Example (Secrecy of N_B – after simplification)

H1. $A \neq i, B \neq i, tr \in nsl$,
H2. $N_B \notin analz(IK(tr))$,
H3. $C \rightarrow D\colon \{N_C, C\}_{\mathrm{pk}(D)} \in set(tr)$,
H4. $D' \rightarrow C\colon \{N_C, N_D, D\}_{\mathrm{pk}(C)} \in set(tr)$
H5. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr)$
H6. $N_B \in analz(\{\ \{N_D\}_{\mathrm{pk}(D)}\ \} \cup IK(tr))$
$\vdash \bot$

- Below we need the long-term key secrecy invariant stating that $\mathrm{inv}(\mathrm{pk}(D)) \in analz(IK(tr)) \Leftrightarrow D = i$ and the fact that $N_B \neq \{N_D\}_{\mathrm{pk}(D)}$.
- **Next in H6:** using the lemma (analz-crypt) yields two cases:
  1. $\mathrm{inv}(\mathrm{pk}(D)) \notin analz(IK(tr)) \ \wedge\ N_B \in \{\{N_D\}_{\mathrm{pk}(D)}\} \cup analz(IK(tr))$
     $\Leftrightarrow\ D \neq i \wedge N_B \in analz(IK(tr))$ Contradiction with H2. Done.
  2. $\mathrm{inv}(\mathrm{pk}(D)) \in analz(IK(tr)) \ \wedge\ N_B \in \{\{N_D\}_{\mathrm{pk}(D)}\} \cup analz(\{N_D\} \cup IK(tr))$
     $\Leftrightarrow\ D = i \wedge N_B \in analz(\{N_D\} \cup IK(tr)))$ Continued on next slides.

Proof of Secrecy of N_B in NSL (nsl3.6)

Example (Secrecy of N_B – case $D = i$)

H1. $A \neq i, B \neq i, tr \in nsl$,
H2. $N_B \notin analz(IK(tr))$,
H3. $C \rightarrow D\colon \{N_C, C\}_{\mathrm{pk}(D)} \in set(tr)$,
H4. $D' \rightarrow C\colon \{N_C, N_D, D\}_{\mathrm{pk}(C)} \in set(tr)$
H5. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr)$
H6. $D = i,\ N_B \in analz(\{N_D\} \cup IK(tr))$
$\vdash \bot$

- **Next on H6:** transform to $N_B \in \{N_D\} \cup analz(IK(tr))$ using lemma (analz-nonce) and then split into the two cases:
  1. $N_D = N_B$ — Rewritten and continuted on next slide.
  2. $N_B \in analz(IK(tr))$ — Done: contradiction with H2.
- **Next on H4 and H5:** Transform events using lemma: $A \rightarrow B\colon m \in set(tr) \Rightarrow m \in parts(IK(tr))$.

Proof of Secrecy of N_B in NSL (nsl3.7)

Example (Secrecy of N_B – case $D = i$)

$$
\begin{array}{ll}
\text{H1.} & A \neq i, B \neq i, tr \in nsl, \\
\text{H2.} & N_B \notin analz(IK(tr)), \\
\text{H3.} & C \rightarrow D\colon \{N_C, C\}_{\mathsf{pk}(D)} \in set(tr), \\
\text{H4.} & \{N_C, N_B, D\}_{\mathsf{pk}(C)} \in parts(IK(tr)) \\
\text{H5.} & \{N_A, N_B, B\}_{\mathsf{pk}(A)} \in parts(IK(tr)) \\
\text{H6.} & D = i, N_D = N_B \\
 & \vdash \bot
\end{array}
$$

- **Next:** we apply the following unicity invariant to H2, H4 and H5:

$$
\begin{array}{l}
\{N_A, N_B, B\}_{\mathsf{pk}(A)} \in parts(IK(tr)) \wedge \\
\{N_C, N_B, D\}_{\mathsf{pk}(C)} \in parts(IK(tr)) \ \wedge \ N_B \notin analz(IK(tr)) \\
\Rightarrow A = C \wedge B = D \wedge N_A = N_C
\end{array}
$$

  It states that N_B uniquely determines the other fields of nsl2.
- Done, since we have a contradiction: $B \neq i$ in H1 and $B = i$ in H6.

Failed NSPK Secrecy Proof

Example (Secrecy of N_B in NSPK - failing proof goal)

H1. $A \neq i, B \neq i, tr \in nsl$
H2. $N_B \notin analz(IK(tr))$
H3. $C \rightarrow D\colon \{N_C, C\}_{\mathrm{pk}(D)} \in set(tr)$
H4. $D' \rightarrow C\colon \{N_C, N_B\}_{\mathrm{pk}(C)} \in set(tr)$
H5. $B \rightarrow A\colon \{N_A, N_B\}_{\mathrm{pk}(A)} \in set(tr)$
H6. $D = i, N_D = N_B$
$\vdash \bot$

- The unicity invariant for N_B in NSPK reads:

$$\{N_A, N_B\}_{\mathrm{pk}(A)} \in parts(IK(tr)) \wedge$$
$$\{N_C, N_B\}_{\mathrm{pk}(C)} \in parts(IK(tr)) \wedge N_B \notin analz(IK(tr))$$
$$\Rightarrow A = C \wedge N_A = N_C$$

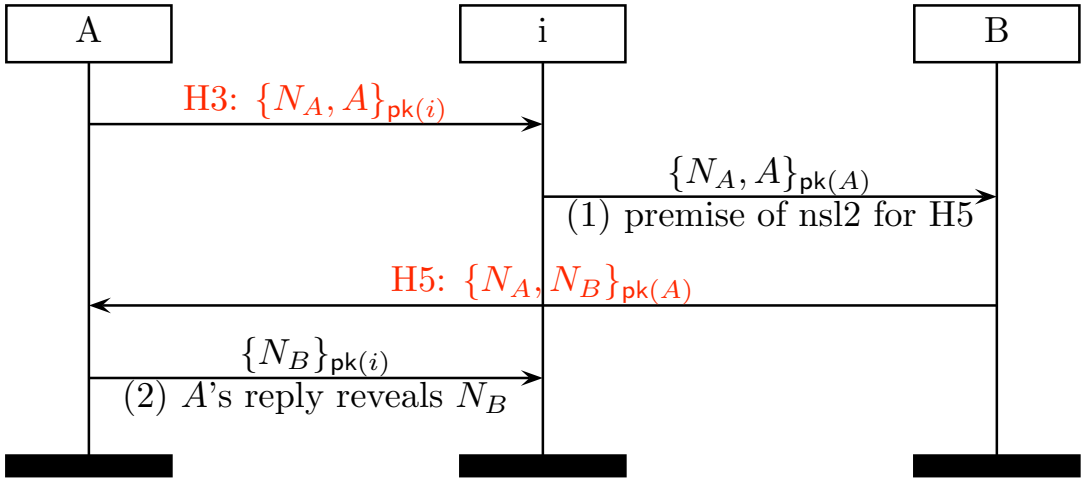
- Here we cannot conclude $B = D$!! The proof state above represents the man-in-the-middle attack on NSPK!

Failed NSPK Secrecy Proof

Example (Secrecy of N_B in NSPK - failing proof goal; H4, H6 deleted)

H1. $A \neq i, B \neq i, tr \in nsl$
H2. $N_B \notin analz(IK(tr))$
H3. $A \rightarrow i\colon \{N_A, A\}_{\mathsf{pk}(i)} \in set(tr)$
H5. $B \rightarrow A\colon \{N_A, N_B\}_{\mathsf{pk}(A)} \in set(tr)$
$\vdash \bot$

msc Relating the failed proof state with the attack on NSPK



Reasoning about the Intruder

Theorem (Inserting an intruder-generated message under *analz*)

$$X \in synth(analz(H)) \wedge m \in analz(\{X\} \cup H)$$
$$\Rightarrow m \in synth(analz(H))$$

- **Intuition:** If the intruder can deduce m by adding a self-generated message X to H, then he can also directly generate m from H.
- Typically, m is a message occuring in a security property (e.g., a nonce that is intended to remain secret), H is the intruder knowledge $IK(tr)$ and X is the message (fake)'d by the intruder.
- Intuition, in other words: sending a self-generated message X does not help the intruder to extend his knowledge.

Proof of Secrecy of N_B in NSL (fake.1)

Example (Secrecy of N_B – induction step for Fake)

H1. $A \neq i, B \neq i, tr \in nsl,$
H2. $Secret(A, B, N_A, N_B, tr),$
H3. $X \in synth(analz(IK(tr)))$
$\vdash$
$Secret(A, B, N_A, N_B, tr \cdot (i \rightarrow C : X)$

- Above: subgoal for (fake) produced by the application of the NSL induction rule.
- **Next:** Unfold the definition of *Secret*.

Proof of Secrecy of N_B in NSL (fake.2)

Example (Secrecy of N_B – after unfolding *Secret*)

H1. $A \neq i, B \neq i, tr \in nsl,$
H2. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr) \Rightarrow N_B \notin analz(IK(tr)),$
H3. $X \in synth(analz(IK(tr))),$
$\vdash$
$B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr \cdot (i \rightarrow C\colon X))$
$\Rightarrow N_B \notin analz(IK(tr \cdot (i \rightarrow C\colon X)))$

Next: Simplify *set* and *IK* in conclusion and propositional reasoning.

Proof of Secrecy of N_B in NSL (fake.3)

Example (Secrecy of N_B – after simplification)

H1. $A \neq i, B \neq i, tr \in nsl$,
H2. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr) \Rightarrow N_B \notin analz(IK(tr))$,
H3. $X \in synth(analz(IK(tr)))$,
H4. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in \{i \rightarrow C\colon X\} \cup set(tr)$
H5. $N_B \in analz(\{X\} \cup IK(tr))$
$\vdash \bot$

Next: split union in H4, yields two cases:

1. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} = i \rightarrow C\colon X$
   Hence $B = i$, contradicting H1. Done.
2. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr)$
   Apply modus ponens with H2, continued on next slides.

Proof of Secrecy of N_B in NSL (fake.4)

Example (Secrecy of N_B – after splitting H4)

H1. $A \neq i, B \neq i, tr \in nsl,$
H2. $N_B \notin analz(IK(tr)),$
H3. $X \in synth(analz(IK(tr)))$
H4. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathsf{pk}(A)} \in set(tr)$
H5. $N_B \in analz(\{X\} \cup IK(tr))$
$\vdash \bot$

Next: apply the following lemma to H3 and H5 (modus ponens in hyps)

$$X \in synth(analz(H)) \wedge m \in analz(\{X\} \cup H)$$
$$\Rightarrow m \in synth(analz(H))$$

Proof of Secrecy of N_B in NSL (fake.5)

Example (Secrecy of N_B – after lemma application)

H1. $A \neq i, B \neq i, tr \in nsl,$
H2. $N_B \notin analz(IK(tr)),$
H3. $X \in synth(analz(IK(tr))),$
H4. $B \rightarrow A\colon \{N_A, N_B, B\}_{\mathrm{pk}(A)} \in set(tr)$
H5. $N_B \in analz(\{X\} \cup IK(tr))$
H6. $N_B \in synth(analz(IK(tr)))$
$\vdash \bot$

Next: Now we can finish the proof branch by applying the lemma

$$N \in synth(H) \Rightarrow N \in H$$

to H6 (the intruder cannot generate fresh nonces himself), which yields a contradiction with H2. Done.

Inductive Method for Reasoning about Protocols

Summary

Inductive definition of security protocols

- We can directly define the traces produced by a security protocol as an inductive set.
- There is one rule for the empty trace (base case) and one (uniform) rule for the intruder.
- There is one rule for sending each protocol message

Inductive proof of security properties

- Each inductive definition has an associated induction rule with base and induction cases corresponding to those in the definition.
- Define security property (e.g., secrecy, authentication) as a predicate on traces and show by induction that it holds for all protocol traces.
- We often need to strengthen the Induction hypothesis by using auxiliary properties such as unicity and secrecy lemmas.
- Some general lemmas about intruder deduction needed to reason about the (fake) rule.

Inductive Method for Reasoning about Protocols

Comparison and Discussion

Protocol definitions

- The sending of each protocol message is specified by a rule of the inductive definition; there are neither role scripts nor threads.
- There is no check whether a reply was already sent. This leads to an over-approximation of a more realistic model including such checks.
- Such checks can be added, but require negative premises, which are potentially difficult to reason about.

Protocol properties

- Properties are defined in a protocol-dependent way. No signals are used as hooks to express properties.
- In particular, the definition of authentication properties appears very ad-hoc. Some definitions seem difficult to understand and justify (why did you define it this and not that way?).
- As a result of the over-approximation, it is not possible to prove strong authentication properties (injective agreement).

Bibliography

- Larry C. Paulson. *The inductive approach to verifying cryptographic protocols*. J. Computer Security 6 (1998), 85–128.
- Giampaolo Bella. *Formal Correctness of Security Protocols*. Springer-Verlag, 2007. Based on author's PhD thesis, University of Cambridge (2000).
- Paulson, Bella et al., Various conference and journal papers. See http://www.cl.cam.ac.uk/ lp15/papers/protocols.html
- Ernie Cohen. *First-order Verification of Cryptographic Protocols*. Journal of Computer Security 11(2): 189-216 (2003)