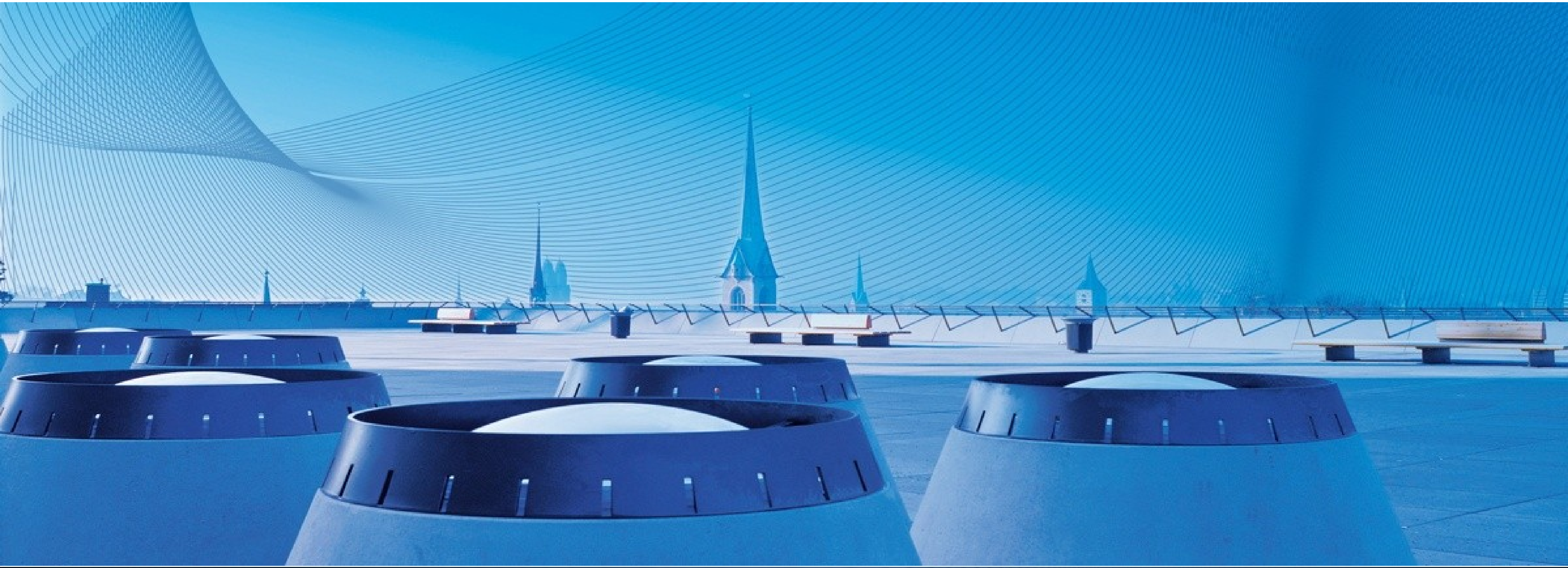
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatik
Computer Science

Introduction to formal protocol analysis

E-Voting Seminar
18 November 2010

Michael Schläpfer
(Some parts adapted from S. Mödersheim)



Formal methods in information security

Formal methods are **techniques** and **tools** based on **mathematics** and **logic** that support the **specification**, **construction** and **analysis** of hardware and software systems.

Some examples:

- Program logics (Hoare logic, dynamic logic)
- Temporal logics (LTL, CTL, TLA, μ-calculus)
- Process algebras (CCS, CSP, π-calculus, Spi-calculus)
- Abstract data types (CASL, Z)
- Development tools (Rodin/Event-B, PVS, VSE)
- Theorem provers (Isabelle, Coq, HOL, Inka)
- Model checkers (Spin, SMV, Murϕ, OFMC, Scyther)

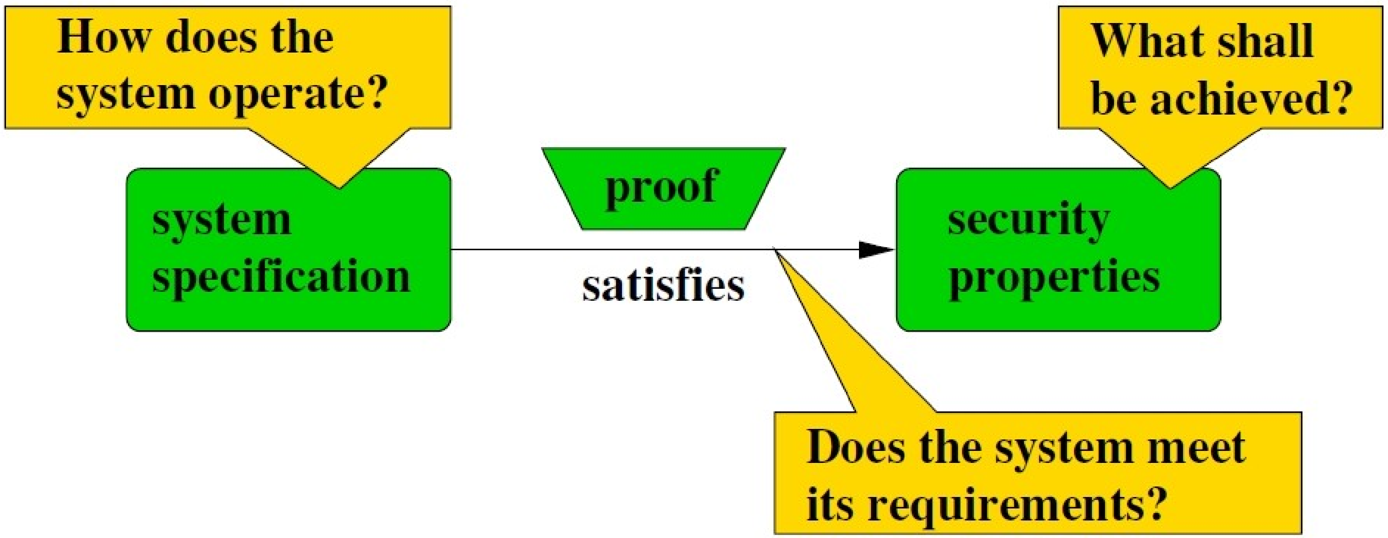
Applying formal methods:

1. Formalize the system requirements as security properties
2. Construct a formal model of the system's behavior, an abstract specification or a concrete program
3. Verify that the system satisfies the properties at the level at which the system has been modeled

Overview

1. Introduction
2. Formal languages for the specification of security protocols
3. The Dolev-Yao intruder model
4. Operational semantics of security protocols
5. Protocol goals and verification
6. Decidability of protocol security and deductive methods
7. Current research topics

Formal security models



- Formal specification with formal languages
- Semantics of languages allow for verification and validation with mathematical methods

Good crypto alone ...

Security protocols

A **protocol** consists of a set of rules (conventions) that determine the exchange of messages between two or more principals. In short, a **distributed algorithm** with emphasis on communication.

Security (or **cryptographic**) protocols use cryptographic mechanisms to achieve security objectives.

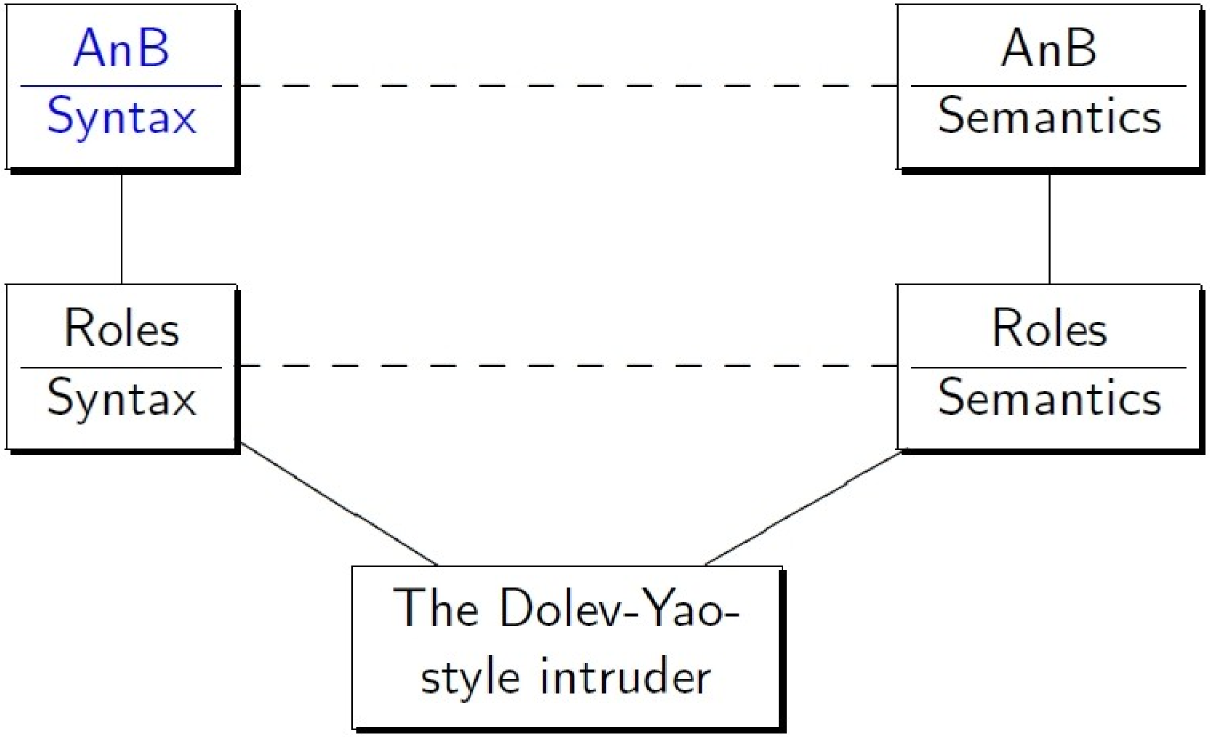
Some common security objectives:

- Entity or message authentication
- Key establishment
- Integrity
- Fair exchange
- Non-repudiation
- ...

Overview

1. Introduction
2. Formal languages for the specification of security protocols
3. The Dolev-Yao intruder model
4. Operational semantics of security protocols
5. Protocol goals and verification
6. Decidability of protocol security and deductive methods
7. Current research topics

Two formal languages



Message notation

Roles: A, B or *Alice*, *Bob*

Agents: a, b, i

Symmetric Keys: K, K_{AB}, . . . ; sk(A, S)

Symmetric Encryption: $\{|M|\}_K$

Public Keys: K, pk(A)

Private Keys: inv(K), inv(pk(A))

Asymmetric Encryption: $\{M\}_K$

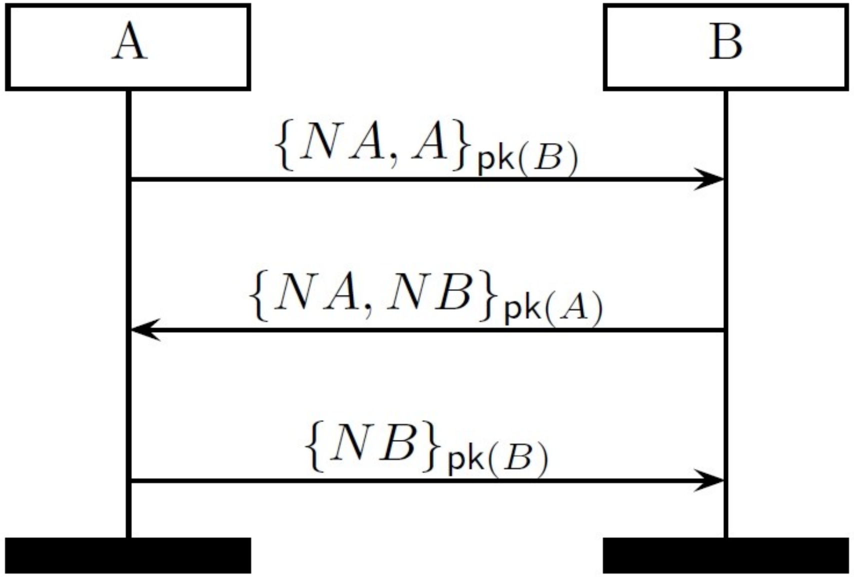
Signing: $\{M\}_{\mathsf{inv}(K)}$

Nonces: NA, $N1$ fresh data items used for challenge/response.
N.B.: sometimes subscripts are used, e.g. N_A, but it does not mean that principals can find out that N_A was generated by A.

Timestamps: T. Denote time, e.g. used for key expiration.

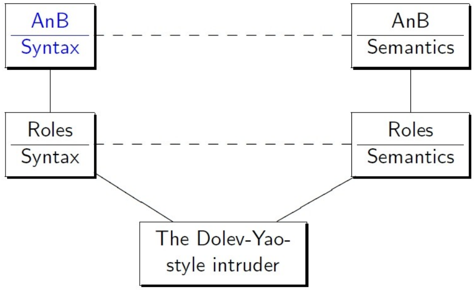
Message concatenation: M_1, M_2, M_3

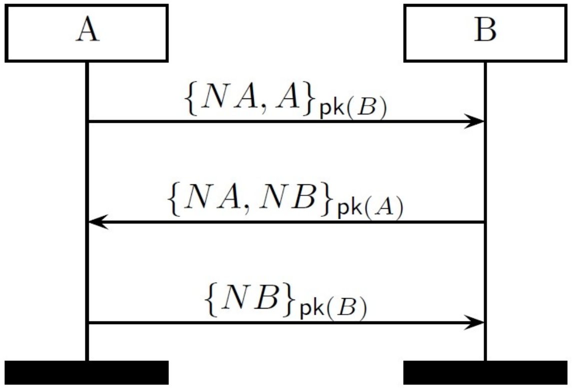
A simple protocol description language



- Message sequence chart between roles A and B
- Represents the famous Needham-Schroeder Public Key protocol (NSPK, 1978)

AnB - Syntax

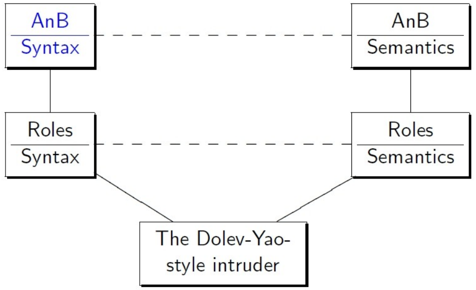




1. $A \to B : \{NA, A\}_{pk(B)}$
2. $B \to A : \{NA, NB\}_{pkA}$
3. $A \to B : \{NB\}_{pk(B)}$

- Message sequence chart between roles A and B
- Represents the famous Needham-Schroeder Public Key protocol (NSPK, 1978)

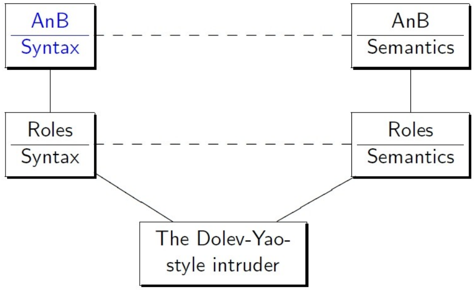
Informal correctness



1. $A \rightarrow B : \{NA, A\}_{\mathsf{pk}(B)}$ "This is Alice and I have chosen a nonce NA."
2. $B \rightarrow A : \{NA, NB\}_{\mathsf{pk}(A)}$ "Here is your nonce NA. Since I could read it, I must be Bob. I also have a challenge NB for you."
3. $A \rightarrow B : \{NB\}_{\mathsf{pk}(B)}$ "You sent me NB. Since only Alice can read this and I sent it back, I must be Alice."

- Secure?

Lowe's attack



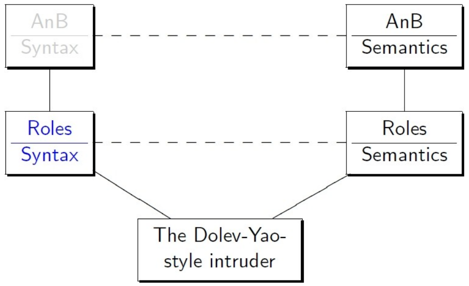
NSPK (1978)

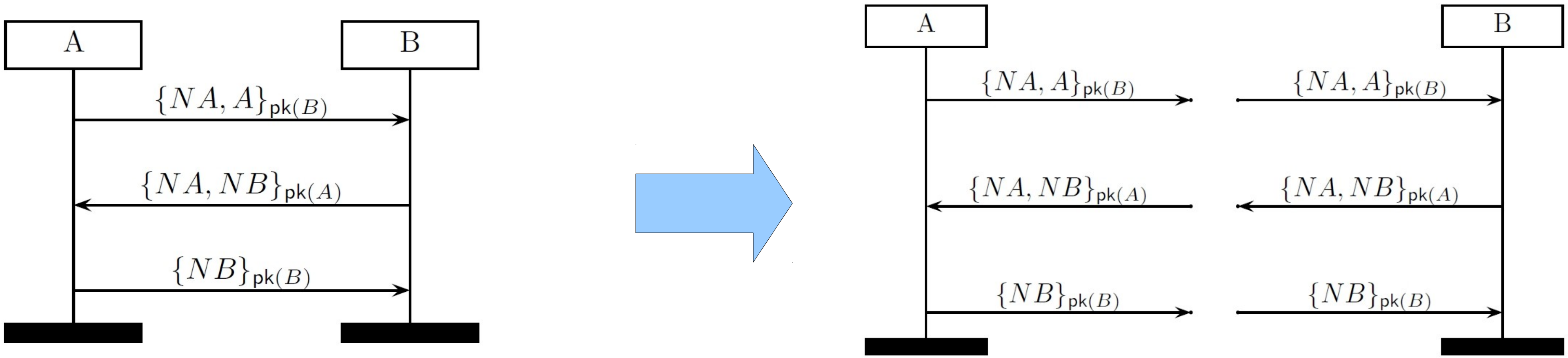
$$A \rightarrow B : \{NA, A\}_{\mathsf{pk}(B)}$$
$$B \rightarrow A : \{NA, NB\}_{\mathsf{pk}(A)}$$
$$A \rightarrow B : \{NB\}_{\mathsf{pk}(B)}$$

Attack (Lowe 1996):

$$1. \quad a \rightarrow i : \quad \{na, a\}_{\mathsf{pk}(i)}$$
$$1.' \quad i(a) \rightarrow b : \quad \{na, a\}_{\mathsf{pk}(b)}$$
$$2.' \quad b \rightarrow i(a) : \quad \{na, nb\}_{\mathsf{pk}(a)}$$
$$2. \quad i \rightarrow a : \quad \{na, nb\}_{\mathsf{pk}(a)}$$
$$3. \quad a \rightarrow i : \quad \{nb\}_{\mathsf{pk}(i)}$$
$$3.' \quad i(a) \rightarrow b : \quad \{nb\}_{\mathsf{pk}(b)}$$

Role scripts for A and B



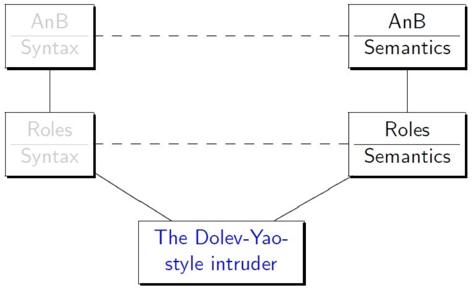


Textual:

$$\mathrm{NSPK}(A) := \mathrm{snd}(\{NA, A\}_{pk(B)}) \cdot \mathrm{rcv}(\{NA, NB\}_{pk(A)}) \cdot \mathrm{snd}(\{NB\}_{pk(B)})$$

Overview

Modelling the attacker



On the Security of Public Key Protocols (IEEE Trans. Inf. Th. 1983):

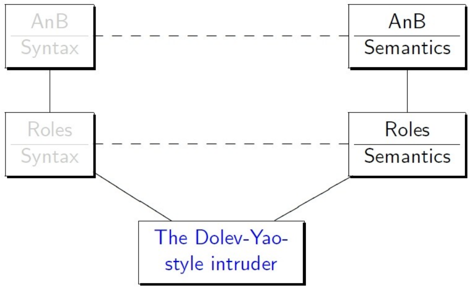
- Danny Dolev
- Andrew C. Yao

The Dolev-Yao Intruder:

- Controls the network (read, intercept, send)
- Is a legitimate user
- Can apply every publicly available information or function
- Can apply his private information and functions
- Cannot break cryptography

Following, a semi-formal overview!

Modelling the attacker



Definition

Given a set of terms M we define $\mathcal{DY}(M)$ as the least closure of M under the following rules:

$$\frac{}{m \in \mathcal{DY}(M)}\ \text{Axiom } (m \in M) \qquad \frac{s \in \mathcal{DY}(M)}{t \in \mathcal{DY}(M)}\ \text{Algebra } (s \approx t)$$

$$\frac{t_1 \in \mathcal{DY}(M) \quad \ldots \quad t_n \in \mathcal{DY}(M)}{f(t_1, \ldots, t_n) \in \mathcal{DY}(M)}\ \text{Composition } (f \in \Sigma_p)$$

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

$$\frac{\{m\}_k \in \mathcal{DY}(M) \quad \text{inv}(k) \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{DecAsym} \qquad \frac{\{m\}_{\text{inv}(k)} \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{OpenSig}$$

A simple example

Example

$M = \{\ x, \{\!|b, \exp(g,y)|\!\}_k, k, m\ \}$

$$\{\!|m|\!\}_{\exp(\exp(g,x),y)} \overset{?}{\in} \mathcal{DY}(M)$$

$$\dfrac{\dfrac{}{\{\!|b, \exp(g,y)|\!\}_k \in \mathcal{DY}(M)} \qquad \dfrac{}{k \in \mathcal{DY}(M)}}{\langle b, \exp(g,y)\rangle \in \mathcal{DY}(M)}$$

$$\dfrac{\langle b, \exp(g,y)\rangle \in \mathcal{DY}(M)}{\exp(g,y) \in \mathcal{DY}(M)}$$

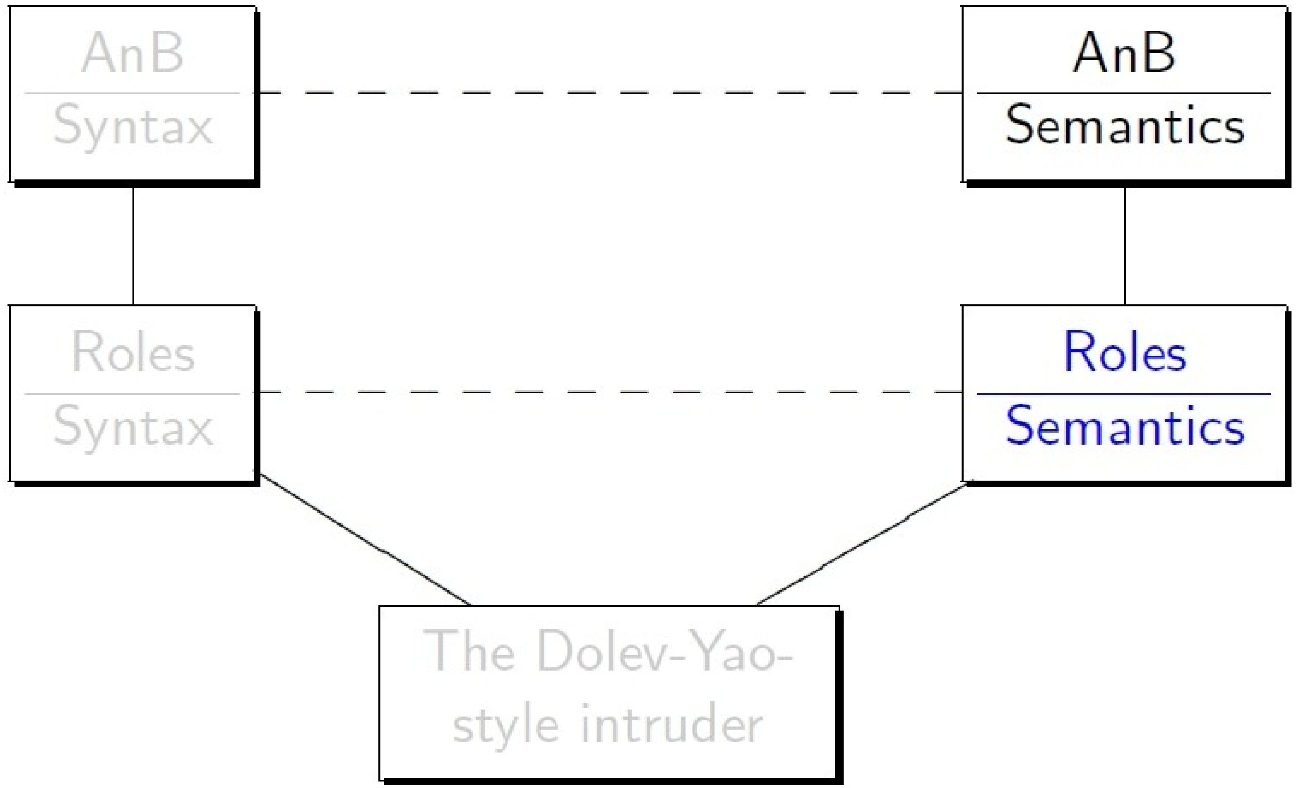
$$\dfrac{\exp(g,y) \in \mathcal{DY}(M) \qquad \dfrac{}{x \in \mathcal{DY}(M)}}{\exp(\exp(g,y),x) \in \mathcal{DY}(M)}$$

$$\dfrac{\exp(\exp(g,y),x) \in \mathcal{DY}(M)}{\exp(\exp(g,x),y) \in \mathcal{DY}(M)}$$

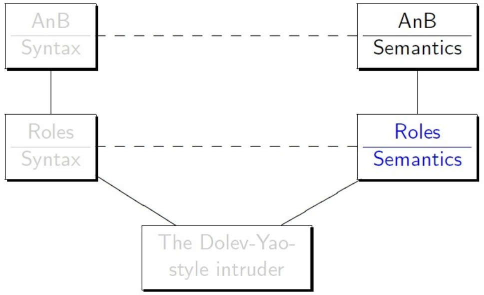
$$\dfrac{\exp(\exp(g,x),y) \in \mathcal{DY}(M) \qquad \dfrac{}{m \in \mathcal{DY}(M)}}{\{\!|m|\!\}_{\exp(\exp(g,x),y)} \in \mathcal{DY}(M)}$$

Overview

Formal semantics of the languages



Operational semantics



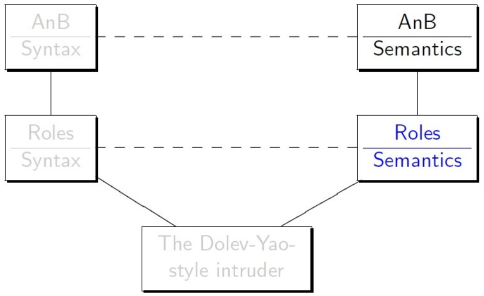
- Defined by a transition system

$$TS(P, IK_0, th_0) = (State, \rightarrow, ([], IK_0, th_0))$$

Definition (State)

- $State = Trace \times IntruderKnowledge \times Threads.$
- $Trace = (TID \times Event)^*$
- $IntruderKnowledge = \mathcal{P}(Term)$
- $Threads = TID \rightharpoonup Role$

where the trace and the intruder knowledge are ground and the threads are closed.



Operational semantics

$$TS(P, IK_0, th_0) = (State, \rightarrow, ([], IK_0, th_0))$$

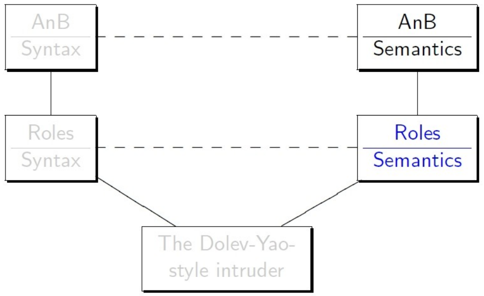
- We start from an initial state
- Roles are instantiated (free variables set)

Example (An initial state that is sufficient to find the attack against NSPK)

$$\begin{aligned} tr_0 &= [] \\ IK_0 &= \{\ a, b, i, \mathrm{pk}(a), \mathrm{pk}(b), \mathrm{pk}(i), \mathrm{inv}(\mathrm{pk}(i))\ \} \\ th_0(0) &= NSPK(A)[A \mapsto a, B \mapsto i, NA \mapsto na_0] \\ th_0(1) &= NSPK(B)[B \mapsto B, NB \mapsto nb_1] \end{aligned}$$

$$\mathrm{NSPK}(A) := \mathrm{snd}(\{NA, A\}_{pk(B)}) \cdot \mathrm{rcv}(\{NA, NB\}_{pk(A)}) \cdot \mathrm{snd}(\{NB\}_{pk(B)})$$

Operational semantics



$$TS(P, IK_0, th_0) = (State, \to, ([], IK_0, th_0))$$

- Transition relation defined by a set of deduction rules
- Signals sig will be explained later

Rules

$$\frac{th(tid) = \mathrm{snd}(t) \cdot tl}{(tr, IK, th) \to (tr \cdot (tid, \mathrm{snd}(t)), IK \cup \{t\}, th[tid \mapsto tl])}\ \mathrm{snd}$$

$$\frac{th(tid) = \mathrm{rcv}(t) \cdot tl \quad dom(\sigma) = var(t) \quad t\sigma \in \mathcal{DY}(IK)}{(tr, IK, th) \to (tr \cdot (tid, \mathrm{rcv}(t\sigma)), IK, th[tid \mapsto tl\sigma])}\ \mathrm{rcv}$$

$$\frac{th(tid) = \mathrm{sig}(sig, t) \cdot tl}{(tr, IK, th) \to (tr \cdot (tid, \mathrm{sig}(sig, t)), IK, th[tid \mapsto tl])}\ \mathrm{sig}$$

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \text{snd}(\{na_0, i\}_{\text{pk}(i)}))$	$\text{snd}(\{na_0, i\}_{\text{pk}(i)})$	$\text{rcv}(\{NA, A\}_{\text{pk}(b)})$
$(1, \text{rcv}(\{na_0, a\}_{\text{pk}(b)}))$	$\text{rcv}(\{na_0, NB\}_{\text{pk}(a)})$	$\text{snd}(\{NA, nb_1\}_{\text{pk}(A)})$
$(1, \text{snd}(\{na_0, nb_1\}_{\text{pk}(a)}))$	$\text{snd}(\{NB\}_{\text{pk}(i)})$	$\text{rcv}(\{nb_1\}_{\text{pk}(b)})$
$(0, \text{rcv}(\{na_0, nb_1\}_{\text{pk}(a)}))$		
$(0, \text{snd}(\{nb_1\}_{\text{pk}(i)}))$		
$(1, \text{rcv}(\{nb_1\}_{\text{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \mathsf{snd}(\{na_0, i\}_{\mathsf{pk}(i)}))$	$\mathsf{snd}(\{na_0, i\}_{\mathsf{pk}(i)})$	$\mathsf{rcv}(\{NA, A\}_{\mathsf{pk}(b)})$
$(1, \mathsf{rcv}(\{na_0, a\}_{\mathsf{pk}(b)}))$	$\mathsf{rcv}(\{na_0, NB\}_{\mathsf{pk}(a)})$	$\mathsf{snd}(\{NA, nb_1\}_{\mathsf{pk}(A)})$
$(1, \mathsf{snd}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$	$\mathsf{snd}(\{NB\}_{\mathsf{pk}(i)})$	$\mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)})$
$(0, \mathsf{rcv}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		
$(0, \mathsf{snd}(\{nb_1\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \mathrm{snd}(\{na_0, i\}_{\mathrm{pk}(i)}))$		$\mathrm{rcv}(\{NA, A\}_{\mathrm{pk}(b)})$
$(1, \mathrm{rcv}(\{na_0, a\}_{\mathrm{pk}(b)}))$	$\mathrm{rcv}(\{na_0, NB\}_{\mathrm{pk}(a)})$	$\mathrm{snd}(\{NA, nb_1\}_{\mathrm{pk}(A)})$
$(1, \mathrm{snd}(\{na_0, nb_1\}_{\mathrm{pk}(a)}))$	$\mathrm{snd}(\{NB\}_{\mathrm{pk}(i)})$	$\mathrm{rcv}(\{nb_1\}_{\mathrm{pk}(b)})$
$(0, \mathrm{rcv}(\{na_0, nb_1\}_{\mathrm{pk}(a)}))$		
$(0, \mathrm{snd}(\{nb_1\}_{\mathrm{pk}(i)}))$		
$(1, \mathrm{rcv}(\{nb_1\}_{\mathrm{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	*th*(0)	*th*(1)
$(0, \mathrm{snd}(\{na_0, i\}_{\mathrm{pk}(i)}))$		
$(1, \mathrm{rcv}(\{na_0, a\}_{\mathrm{pk}(b)}))$	$\mathrm{rcv}(\{na_0, NB\}_{\mathrm{pk}(a)})$	$\mathrm{snd}(\{na_0, nb_1\}_{\mathrm{pk}(a)})$
$(1, \mathrm{snd}(\{na_0, nb_1\}_{\mathrm{pk}(a)}))$	$\mathrm{snd}(\{NB\}_{\mathrm{pk}(i)})$	$\mathrm{rcv}(\{nb_1\}_{\mathrm{pk}(b)})$
$(0, \mathrm{rcv}(\{na_0, nb_1\}_{\mathrm{pk}(a)}))$		
$(0, \mathrm{snd}(\{nb_1\}_{\mathrm{pk}(i)}))$		
$(1, \mathrm{rcv}(\{nb_1\}_{\mathrm{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \mathsf{snd}(\{na_0, i\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{na_0, a\}_{\mathsf{pk}(b)}))$	$\mathsf{rcv}(\{na_0, NB\}_{\mathsf{pk}(a)})$	
$(1, \mathsf{snd}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$	$\mathsf{snd}(\{NB\}_{\mathsf{pk}(i)})$	$\mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)})$
$(0, \mathsf{rcv}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		
$(0, \mathsf{snd}(\{nb_1\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \mathsf{snd}(\{na_0, i\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{na_0, a\}_{\mathsf{pk}(b)}))$		
$(1, \mathsf{snd}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$	$\mathsf{snd}(\{nb_1\}_{\mathsf{pk}(i)})$	$\mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)})$
$(0, \mathsf{rcv}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		
$(0, \mathsf{snd}(\{nb_1\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \mathsf{snd}(\{na_0, i\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{na_0, a\}_{\mathsf{pk}(b)}))$		
$(1, \mathsf{snd}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		$\mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)})$
$(0, \mathsf{rcv}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		
$(0, \mathsf{snd}(\{nb_1\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)}))$		

Attack on NSPK

Example (NSPK Attack)

Trace	$th(0)$	$th(1)$
$(0, \mathrm{snd}(\{na_0, i\}_{\mathrm{pk}(i)}))$		
$(1, \mathrm{rcv}(\{na_0, a\}_{\mathrm{pk}(b)}))$		
$(1, \mathrm{snd}(\{na_0, nb_1\}_{\mathrm{pk}(a)}))$		
$(0, \mathrm{rcv}(\{na_0, nb_1\}_{\mathrm{pk}(a)}))$		
$(0, \mathrm{snd}(\{nb_1\}_{\mathrm{pk}(i)}))$		
$(1, \mathrm{rcv}(\{nb_1\}_{\mathrm{pk}(b)}))$		

Attack trace!

Overview

Protocol goals

Goals, the protocol should achieve:

- **Authenticate** messages, binding them to their originator
- Guarantee **secrecy** of certain items (e.g. keys)
- Sender invariance
- Anonymity
- Non-repudiation
- ...

Most common goals:

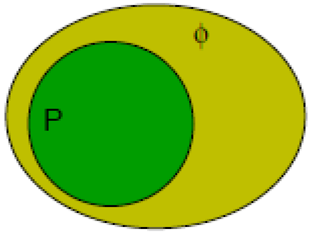
- Secrecy
- Authentication (many different forms)

Protocol properties

Properties:
- Semantics of a security protocol P is a set of traces $||P|| = traces(P)$
- Security goal / property ϕ also denotes a set of traces $||\phi||$

Correctness:
- Protocol P satisfies property ϕ, written $P \models \phi$, iff

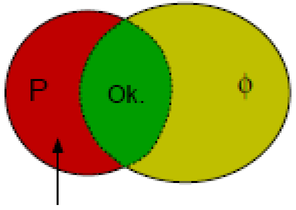
$$||P|| \subseteq ||\phi||$$



Ok, no attacks.

- Attack traces are those in

$$||P|| - ||\phi||$$



Attacks.

- Every correctness statement is either true or false

Formalizing security properties

Direct formulation:

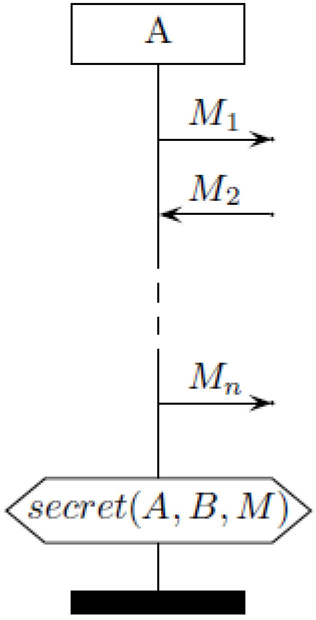
- Formulate property ϕ directly in terms of send and receive events occuring in protocol traces, i.e., as a set of (or predicate on) traces
- Drawback: Standard properties like secrecy and authentication become highly protocol-dependent, since they need to refer to the concrete protocol messagees.

Protocol instrumentation

- Insert special signal events into the protocol roles
- Possible to express properties independently of protocol
- Example:

sig(*secret, A, B, M*)

claims that *M* is a secret shared by roles *A* and *B*



Signal events

Remember:

Signal rule

$$\frac{th(tid) = \mathrm{sig}(sig, t) \cdot tl}{(tr, IK, th) \to (tr \cdot (tid, \mathrm{sig}(sig, t)), IK, th[tid \mapsto tl])}\ \mathrm{sig}$$

Properties of signal events:

- Used to record facts of claims in the protocol trace
- Since they are artificially inserted into the protocol, the intruder cannot observe or modify or generate them
- Properties formulated from the point of view of a given role, thus yielding security guarantees for that specific role

Formalizing secrecy

Example (NSPK Attack)

Trace	*th*(0)	*th*(1)
$(0, \mathsf{snd}(\{na_0, i\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{na_0, a\}_{\mathsf{pk}(b)}))$		
$(1, \mathsf{snd}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		
$(0, \mathsf{rcv}(\{na_0, nb_1\}_{\mathsf{pk}(a)}))$		
$(0, \mathsf{snd}(\{nb_1\}_{\mathsf{pk}(i)}))$		
$(1, \mathsf{rcv}(\{nb_1\}_{\mathsf{pk}(b)}))$		

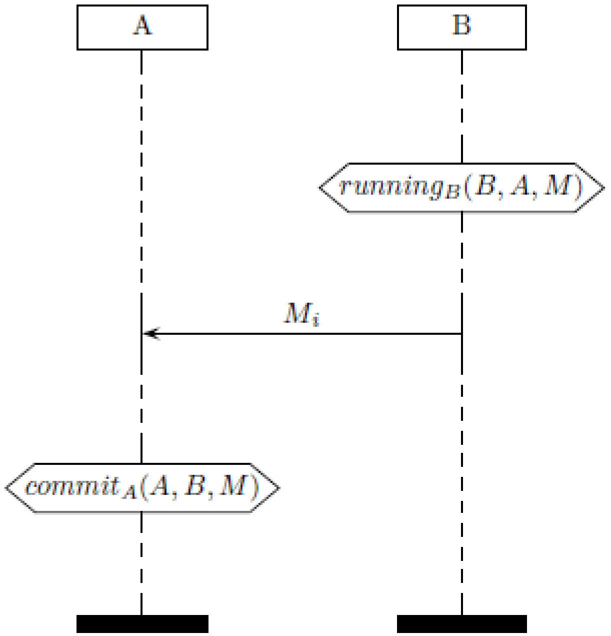
Definition (Secrecy)

The property *Secret*(*A*, *B*, *M*) consists of all traces *tr* satisfying

$$\forall tid.\ (tid, \mathsf{sig}(secret, A, B, M)) \in set(tr)\ \wedge\ B \neq i \Rightarrow M \notin \mathcal{DY}(IK(tr))$$

$$IK(tr) = \{m | \exists tid.(tid, snd(m)) \in set(tr)\}$$

Formalizing authentication



Two new signals:

- *running*
- *commit*

Different definitions:

- Aliveness
- Weak agreement
- Non-injective agreement
- Injective agreement
- ...

Example:

Definition (Non-injective agreement)

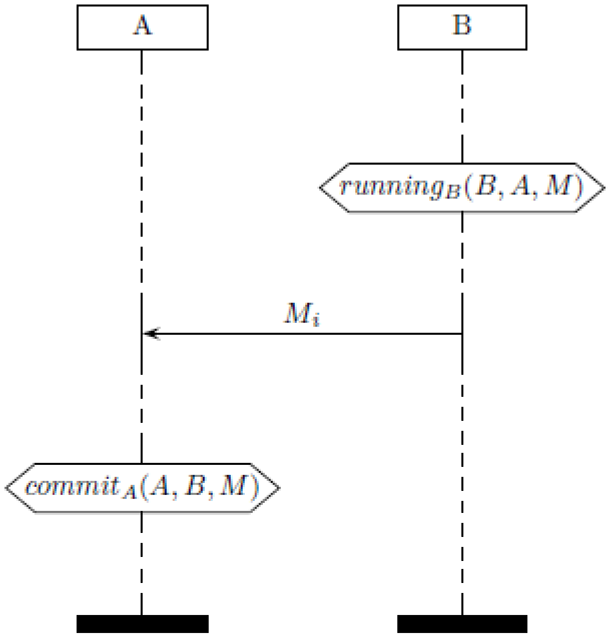
We define $tr \in Agreement_{NI}(A, B, M)$ for a trace tr by

$$\forall tid.\ (tid, \mathrm{sig}(commit_A, A, B, M)) \in set(tr) \land B \neq i$$
$$\Rightarrow \exists tid'.(tid', \mathrm{sig}(running_B, B, A, M)) \in set(tr)$$

Formalizing authentication

Two new signals:

- *running*
- *commit*



Different definitions:

- Aliveness
- Weak agreement
- Non-injective agreement
- Injective agreement
- ...

Example:

Definition (Injective agreement)

We define $tr \in Agreement(A, B, M)$ for a trace tr iff there is an injective function $g : TID \rightarrow TID$ such that

$$\forall tid.\ (tid, \mathrm{sig}(commit_A, A, B, M)) \in set(tr) \wedge B \neq i$$
$$\Rightarrow (g(tid), \mathrm{sig}(running_B, B, A, M)) \in set(tr)$$

Overview

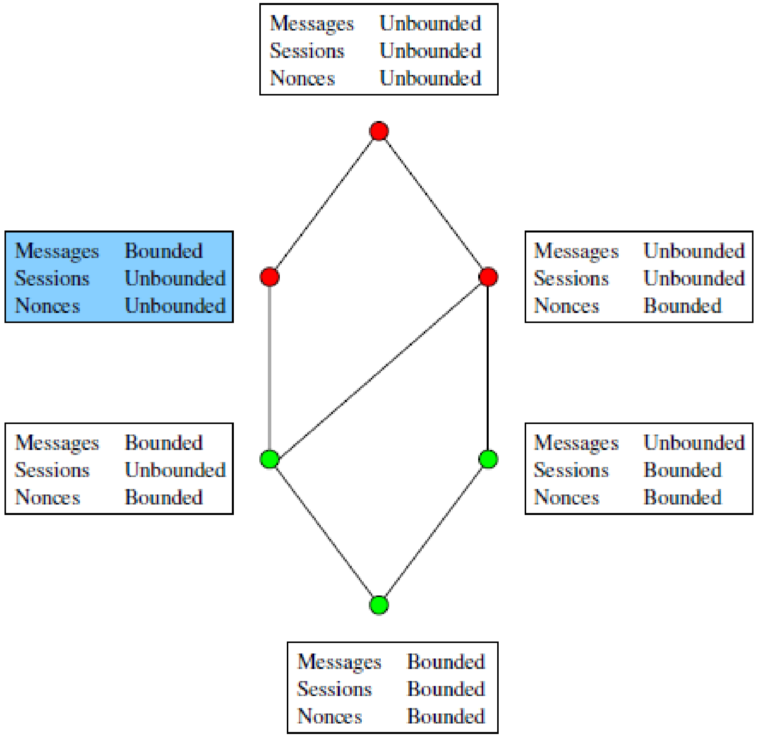
1. Introduction
2. Formal languages for the specification of security protocols
3. The Dolev-Yao intruder model
4. Operational semantics of security protocols
5. Protocol goals and verification
6. Decidability of protocol security and deductive methods
7. Current research topics

Decidability of protocol security

Algorithmic analysis:
- Fully automatic analysis
- Correctness in general undecidable

Sources of infinity:
- Messages
- Sessions
- Nonces

Solutions:
- Various kinds of abstractions
  (Not covered here!)



Deductive methods (Maybe next talk?)

Generality:

- Deductive methods can handle all infinite state spaces
- No need for finiteness bounds (e.g. on messages, nonces, sessions)
- Properties are defined over reachable states and proven by induction

Expressiveness:

- Flexible platform for experimentation
- Possibility to prove meta-results about a model

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

Drawback:

- Loss of automation, proofs generally require user interaction and profound knowledge of both, the used tools and protocol itself

Overview

Current research topics

Modeling more complex protocols:

- Modeling complex protocols is non-trivial
- Some work in progress at ETH (e.g. KERBEROS)
- Electronic voting protocols, an interesting application?

Formalizing electronic voting specific properties and goals:

- Receipt-freeness?
- Coercion-resistance?
- ...?

Open Issues:

- Secure Platform Problem (some work in progress at ETH)
- Adaptive Corruption (some work in progress at ETH)
- Side-channel attacks

Questions

?