Symbolic model checking of security protocols

Hauptseminar Perlen der Informatik

Felix Weninger

TU München

August 1, 2006
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
</tr>
<tr>
<td>Challenges</td>
</tr>
<tr>
<td>Modelling a state transition system</td>
</tr>
<tr>
<td><strong>Ground models</strong></td>
</tr>
<tr>
<td>Basic definitions</td>
</tr>
<tr>
<td>Protocol specification</td>
</tr>
<tr>
<td>State reachability</td>
</tr>
<tr>
<td><strong>Lazy models</strong></td>
</tr>
<tr>
<td>Lazy states</td>
</tr>
<tr>
<td>Constraint reduction</td>
</tr>
<tr>
<td><strong>Conclusion</strong></td>
</tr>
</tbody>
</table>
Security protocols

- Interaction of **honest agents** (Alice, Bob, merchant, ...) and an **intruder** (Eve)
- Exchange of **messages** over a **network**
- Use of **cryptography**
- Security of protocols = Achievement of defined **goals** (e.g. message secrecy)
Model checking of security protocols

- Formalization as state transition system:
  - states represent knowledge of agents
  - transitions represent message exchange, according to protocol rules
- Infinity of messages can be generated
- Security problem is *undecidable* (∼ co-re)
- “State explosion“

Possible ideas:

- Introduction of variables
- Lazy evaluation

→ symbolic model checking
Agents

**Honest agents:**
- Behaviour follows predefined rules
- Cannot break cryptography, e.g. determine the inverse of a key (perfect cryptography assumption)

**Intruder:**
- Can pose as any honest agent
- Enhances knowledge by intercepting messages
- Maintains knowledge across transitions (monotonicity)
- Actively generates and sends messages generated from his knowledge
- Cannot break cryptography either
Introduction

Challenges

Modelling a state transition system

Ground models

Basic definitions

Protocol specification

State reachability

Lazy models

Lazy states

Constraint reduction

Conclusion
Messages

- Use of context-free grammar for formalization of protocols
- Grammar = Input format of our model checking software

Paradigm: “Everything is a message“ – untyped language to be able to detect type flaw attacks!

\[
\begin{align*}
Msg & ::= \text{AtomicMsg} | \text{ComposedMsg} \\
\text{AtomicMsg} & ::= C | V \\
\text{ComposedMsg} & ::= \langle \text{Msg}, \text{Msg} \rangle | \text{Msg}(\text{Msg}) | \\
& \quad \text{Msg}|_{\text{Msg}} | \text{Msg} \parallel_{\text{Msg}} | \text{Msg}^{-1}
\end{align*}
\]

with \( C \) and \( V \) being arbitrary countable sets of constant and variable identifiers.
Intruder capabilities

**Dolev–Yao intruder:** For set of messages $M$, $\mathcal{DY}(M)$ is the closure of $M$ under

- **message generation:** composition, encryption, function application
- **message analysis:** decomposition, decryption (requires key)

**Example 1**

\[
\begin{align*}
\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{m_1 | m_2 \in \mathcal{DY}(M)} & \quad G_{\text{acrypt}} \\
\frac{m_1 | m_2 \in \mathcal{DY}(M) \quad m_2^{-1} \in \mathcal{DY}(M)}{m_1 \in \mathcal{DY}(M)} & \quad A_{\text{acrypt}} \\
\frac{m_1 | m_2^{-1} \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{m_1 \in \mathcal{DY}(M)} & \quad A^{-1}_{\text{acrypt}}
\end{align*}
\]
Ground terms and substitutions

Definition 2
A term is ground iff it contains no variables.

Definition 3
A substitution $\sigma$ is a mapping $V \rightarrow L(Msg)$. We denote by $\text{dom}(\sigma)$ the set of variables that are substituted. A substitution is ground iff $\sigma(v)$ is ground for every $v \in \text{dom}(\sigma)$. We denote by $t\sigma := \sigma(t)$ the application of $\sigma$ to an arbitrary term $t$. 
Introduction

Challenges

Modelling a state transition system

**Ground models**

- Basic definitions

**Protocol specification**

- State reachability

**Lazy models**

- Lazy states

- Constraint reduction

Conclusion
States

States are nonempty finite sets of positive facts.

\[
\text{State} ::= \text{StateFact}. \text{StateFact}^* \\
\text{StateFact} ::= \text{agentState}(\text{Msg}) | \text{PosFact} \\
\text{PosFact} ::= \text{intKnows}(\text{Msg}) | \text{seen}(\text{Msg}, \text{Msg}) | \ldots
\]
Transitions

\[ \text{agentState}(B, 1) \]
Transitions

agentState(B, 2, M₁)
Transitions

Network

agentState(B, 2, M1)
Transitions

agentState(\(B, 2, M_1\))

3 steps in 1 transition!
Transitions

\[ \text{Intruder} = \text{Network!} \]
Transitions
Transitions

A

DY(IK)

IK

M₁

B

M₁

Intruder

Protocol specification
Transitions

Diagram showing transitions involving symbols and processes labeled as $M_1$, $IK$, $DY(IK)$, and $M'_1$. The diagram illustrates the interaction between A and B in the context of intruder models.
Transitions

\[
DY(IK) \\
M_1' \\
IK \\
M_1 \\
M_1 \\
M_1'
\]

A \rightarrow B \rightarrow A
Transitions
Transitions

Diagram showing the interaction between A, B, and the Intruder model. The transition DY(IK) is highlighted, indicating a specific interaction or state change in the protocol specification.
Step compression

- 3 steps in 1 transition:
  - honest agent receives message from intruder
  - honest agent changes state (≈ message processing)
  - honest agent sends message to intruder

- Intruder knowledge “grows“ by the outgoing message (set semantics!)

- No need to specify senders and recipients

→ Simpler rule format and less rules!
Example: E-mail exchange

Alice sends Bob a signed and encrypted message; Bob checks Alice’s signature and decrypts the message.

Initial state:

\[
\begin{align*}
I &= \text{agentState}(\text{roleA, step1, Hi, ka, } kb^{-1}) . \\
&\quad \text{agentState}(\text{roleB, step1, ka}^{-1}, kb) . \\
&\quad \text{intKnows}(ka^{-1}, kb^{-1}, \text{dummy})
\end{align*}
\]

send rule:

\[
\begin{align*}
\text{msgTransit}(\text{dummy}) . \\
&\quad \text{agentState}(\text{roleA, step1, M, KA, KB}^{-1}) \\
&\Rightarrow \text{agentState}(\text{roleA, step2, M, KA, KB}^{-1}) . \\
&\quad \text{msgTransit}(M|_{KB^{-1}}, \text{md5}(M)|_{KA})
\end{align*}
\]
E-mail exchange (2)

Substitution:

\[ \sigma = \{ M \mapsto "\text{Hi}" , \ KA \mapsto ka , \ KB \mapsto kb \} \]

Successor state:

\[ S = \text{agentState}(\text{roleA}, \text{step2}, "\text{Hi}", ka, kb^{-1}) \cdot \text{agentState}(\text{roleB}, \text{step1}, ka^{-1}, kb) \cdot \text{intKnows}(ka^{-1}, kb^{-1}, \text{dummy}, "\text{Hi}"|_{kb^{-1}}, \text{md5}("\text{Hi}")|_{ka}) \]

receive rule:

\[ \text{msgTransit}(M|_{KB^{-1}}, \text{md5}(M)|_{KA}) \cdot \text{agentState}(\text{roleB}, \text{step1}, KA^{-1}, KB) \Rightarrow \text{agentState}(\text{roleB}, \text{step2}, M|_{KB^{-1}}, KB, \text{md5}(M)|_{KA}, KA^{-1}, KB) \cdot \text{msgTransit}(\text{dummy}) \]
Rules

Rules describe possible state transitions: “LHS applicable to the current state ⇒ system changes state to RHS“

Rule ::= LHS ⇒ RHS
LHS ::= msgTransit(Msg) . agentState(Msg) (. PosFact)* (. NegFact)* Condition
RHS ::= agentState(Msg) . msgTransit(Msg) (. PosFact)*

NegFact ::= ¬PosFact
Condition ::= (\∧ Msg ≠ Msg)*

- Rule format corresponds to concept of step compression
- Can model a wide range of protocols
- Insert dummy messages if needed
Protocols

**Observation:** LHS of rules can be used to model attack situations: 
agentState(...) \* msgTransit(...) \* intKnows(secret)

**Definition 4**
A protocol is a triple \((I, R, A)\) where

- \(I\) is a ground initial state,
- \(R\) a set of rules,
- \(A\) is a set of LHS that describe attack situations

and the following holds for every rule in \(R\):

- no new variables are introduced on the RHS
- all variables in conditions and negative facts must occur in positive facts
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Ground models</th>
<th>Lazy models</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenges</td>
<td>Basic definitions</td>
<td>Constraint reduction</td>
<td></td>
</tr>
<tr>
<td>Modelling a state transition system</td>
<td>Protocol specification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ground models**

- Basic definitions
- Protocol specification

**State reachability**

**Lazy models**

- Lazy states
- Constraint reduction

**Conclusion**
Rule applicability

\[ \text{msgTransit}(m_1) \cdot \text{agentState}(m_2) \cdot P_1 \cdot N_1 \land C \]
\[ \implies \text{agentState}(m_3) \cdot \text{msgTransit}(m_4) \cdot P_2 \]

Formalize LHS semantics by a function that yields all substitutions under which rule \( lhs \implies rhs \) can be applied to ground state \( S \):

\[
\text{applicable}_{lhs}(S) = \{ \sigma \mid \\
ground(\sigma) \land \text{dom}(\sigma) = \text{vars}(m_1) \cup \text{vars}(m_2) \cup \text{vars}(P_1) \land \\
\{ m_1 \sigma \} \cup \{ m \sigma \mid \text{intKnows}(m) \in P_1 \} \\
\subseteq \mathcal{D} \mathcal{Y}(\{ m \mid \text{intKnows}(m) \in S \}) \land \\
\text{agentState}(m_2 \sigma) \in S \land \overline{P_1} \sigma \subseteq S \land \\
(\forall f. \neg f \in N_1 \implies f \sigma \notin S) \land \sigma \models C \\
\}
\]

where \( \overline{P_1} \) is \( P_1 \) without \text{intKnows}() facts.
Step function

\[ \text{msgTransit}(m_1) \cdot \text{agentState}(m_2) \cdot P_1 \cdot N_1 \land C \Rightarrow \text{agentState}(m_3) \cdot \text{msgTransit}(m_4) \cdot P_2 \]

Formalize RHS semantics by step function:

\[
\text{step}_{lhs\Rightarrow rhs}(S) = \{ S' \mid \exists \sigma. \\
\sigma \in \text{applicable}_{lhs}(S) \land \\
S' = (S \setminus (\text{agentState}(m_2\sigma) \cup \overline{P_1\sigma})) \\
\cup \{ \text{agentState}(m_3\sigma), \text{intKnows}(m_4\sigma), P_2\sigma \}\}
\]
Reachability

For a protocol $(I, R, A)$ and a state $S$, we define a successor function:

$$\text{succ}_R(S) = \bigcup_{r \in R} \text{step}_r(S)$$

The ground model of the protocol is the set of states that are reachable from the initial state:

$$\text{reach}(I, R) = \bigcup_{n \in \mathbb{N}} \text{succ}_R^n(I)$$

A protocol is secure iff

$$\text{applicable}_a(S) = \emptyset \ \forall a \in A, S \in \text{reach}(I, R).$$
Discussion

- Attack rule integrity:

  \[
  \text{agentState}(\text{roleA}, \text{step2}, M, \ldots) \cdot \text{agentState}(\text{roleB}, \text{step2}, DM, \ldots) \land M \neq DM
  \]

  ... does not work as intended since $M \neq DM$ is true for $DM \equiv \text{“Hi“}|_{b-1}|_b$ and $M \equiv \text{“Hi“}$ (free algebra assumption)

- What about $md5(M_1) \neq md5(M_2)$?

- In general: infinity of possible substitutions due to DY intruder!
  
  (E-mail exchange without signature: substitution of arbitrary message possible!)
Introduction
  Challenges
  Modelling a state transition system

Ground models
  Basic definitions
  Protocol specification
  State reachability

Lazy models
  Lazy states
  Constraint reduction

Conclusion
Motivation

**Observation:** Instantiation of message variables is often irrelevant!

**Idea:** Don’t branch into different states for possible messages, but keep message variables and store constraints about messages generated during transition!

\[
\text{msgTransit}(M|_k)
\]

\[
IK = \{m_1, m_2, k\}
\]

\[
\cdots m_1 / M 
\]

\[
\cdots m_2 / M \text{from}(M(m_1, m_2, k)} \)
\]
Constraints

Definition 5
A constraint is of the form from \((T, IK)\) where

- \(T\) is the set of message terms to be generated (according to \(DY\) rules);
- \(IK\) is a set of message terms, representing intruder knowledge.

A constraint from \((T, IK)\) is satisfiable if there exists a ground substitution \(\sigma\) such that \(T\sigma \subseteq DY(IK\sigma)\).
Lazy states

- Facts may contain variables
- Negative facts (and conditions) cannot be evaluated if they contain variables
  $\rightarrow$ store them in inequalities of the form $m_1 \neq m_2$, $m_1, m_2 \in L(Msg)$

**Definition 6**
A lazy state is a triple $(P, C, N)$, where $P$ is a set of (not necessarily ground) facts, $C$ is a set of constraints and $N$ is a conjunction of disjunctions of inequalities.
Crash course: Unification

**Goal:** For two terms $t_1$ and $t_2$, find $\sigma$ s.t. $t_1\sigma = t_2\sigma$.

**Example 7**

$t_1 := f(A)$, $t_2 := f(g(B))$ unify under $\sigma_1 := \{A \mapsto g(B)\}$ and $\sigma_2 := \{A \mapsto g(b), B \mapsto b\}$.

- In the example, $\sigma_1$ is the most general unifier (mgu) because it substitutes less free variables.
- **Matching:** The special case where one of $t_1$ and $t_2$ is ground (cf. applicability of rules in ground model)
Adaption of state transitions

\[
\text{msgTransit}(m_1) \cdot \text{agentState}(m_2) \cdot P_1 \cdot N_1 \land C \\
\Rightarrow \text{agentState}(m_3) \cdot \text{msgTransit}(m_4) \cdot P_2
\]

Rule applicability to lazy states:

\[
\text{applicable}_{lhs}(P, C, N) = \{(\sigma, C', N') | \}
\]
\[
\text{dom}(\sigma) \subseteq \text{vars}(m_1) \cup \text{vars}(m_2) \cup \text{vars}(P_1) \cup \text{vars}(P, C, N) \land
\]
\[
C' = C \cup \text{from}(m_1 \cup \{m | \text{intKnows}(m) \in P_1\}, \}
\]
\[
\{i | \text{intKnows}(i) \in P\}) \land
\]
\[
\text{agentState}(m_2\sigma) \in P_\sigma \land \overline{P_1}\sigma \subseteq P_\sigma \land
\]
\[
N' = N_\sigma \land \bigwedge \phi \land \text{Cond}\sigma
\]
\[
\phi \in \text{subCont}(N_1\sigma, P\sigma)
\]

where \(\overline{P_1}\) is \(P_1\) without \(\text{intKnows}\) facts. where

\(\text{subCont}(N_1\sigma, P\sigma)\) is a formula that excludes all unifiers under which a positive fact in the state occurs in the negative facts in the rule.
Definitions of successor function, reachability and protocol security: straightforward extension of ground model.

**Theorem 8 (Lazy reachability theorem, informal)**

The set of reachable ground states is exactly the set of reachable lazy states under substitutions that satisfy their constraints and inequalities.

**Implementation tasks:**

- Compute lazy successor states by unification with mgu → finite branching
- Prune lazy states with unsatisfiable constraints or inequalities
- Check remaining states for applicability of attack-rules

**Problem:** How to check satisfiability?
Simple constraints

- A constraint from \((T, IK)\) is simple iff \(T \subseteq V\).
- Simple constraints are always satisfiable.
- For a simple constraint set \(C\) and a satisfiable set of inequalities \(N\), there is always a substitution that satisfies both \(C\) and \(N\).
A motivating example

**Observation:** Constraints can be simplified!

**Example 9**

\[
\text{from } (\{ h(M)|_{A^{-1}}, M|_{a^{-1}} \} , \{ a^{-1} \parallel b, b, h \}) \\
\rightarrow \text{from } (\{ \ldots \} , \{ a^{-1}, a^{-1} \parallel b, b, h \}) \cup \text{from } (\{ b \} , \{ a^{-1} \parallel b, b, h \}) \\
\rightarrow \text{from } (\{ h(M), A^{-1}, M, a^{-1} \} , \{ \ldots \}) \\
\rightarrow \text{from } (\{ h, M, a^{-1} \} , \{ \ldots \}) \\
\rightarrow \text{from } (\{ M \} , \{ \ldots \})
\]
The constraint reduction technique

- **Generation rules:** reduce the set of terms to be generated
- **Analysis rules:** “normalize“ intruder knowledge
- **Reduction step:** $(C_1, σ_1) \vdash (C_2, σ_2)$, where $σ_2$ extends $σ_1$
- **Goal:** $(C_0, id) \vdash^n (C_n, σ_n)$ and $C_n$ is simple
- $σ$ is extended by means of unification
- In general different outcomes $(C_n, σ_n)$ are possible

**Theorem 10 (Correctness and completeness, informal)**

*Constraint reduction reduces every set of constraints to (a set of) constraint sets where every constraint is simple or unsatisfiable, preserving the set of satisfying substitutions.*
To do

- Decidability?
- Search strategy
  - Reachability and constraint satisfiability are independent
  - Constraint reduction may lead to case splits
  - Goal: Combine constraint reduction and search for reachable states in efficient manner, without excluding solutions
- Symbolic sessions
  - Agents can pose as other agents (man-in-the-middle attack)
  - Enumeration of all possibilities is costly
  - Idea: Introduce variables for roles and let the lazy model do the work!
The techniques in this talk have been used to implement the model checker OFMC.

OFMC uses a high-level protocol specification language (HLPSL).

In a test suite of 37 flawed industrial-grade protocols, OFMC generates attack traces within seconds on a standard PC.

OFMC found a previously unknown security flaw in the Siemens H.503 mobile communication protocol, which caused Siemens to revise the protocol.