FIRST STEPS IN INTEGRATING MAUDE-NPA WITH CPSA

Catherine Meadows
Santiago Escobar
Jose Meseguer

May 7, 2007
THE TWO TOOLS

- **Maude-NPA**
  - Reachability analysis tool for crypto protocols
  - Provides support for algebraic properties including AC
  - Searches backwards from states described in terms of strands executed and terms known by the intruder

- **CPSA**
  - Given a strand (skeleton) produces strands that must have executed for it to execute, plus relationships between them
    - Partial temporal order among actions in strands
    - Which terms are necessarily secret
    - Which terms are necessarily fresh (nonces)
WHY NOT INTEGRATE THE TWO?

• Provide shapes as goals to Maude-NPA
• Use partial order, secrecy, and freshness information as constraints on search
• Combine best features of both
  – If protocol is secure, get benefits of quick CPSA analysis
  – If protocol is insecure, NPA produces attack
    • Constraints could speed up NPA search
  – If protocol secure, but not provably so by CPSA, can use Maude-NPA as backup proof engine
OUTLINE OF TALK

• Quick overview of Maude-NPA and CPSA
• Description of how we implemented shapes in Maude-NPA
• Our results (somewhat surprising!)
• Suggestions for future work
BASIC STRUCTURE OF MAUDE-NPA

• Uses strand space model
• Searches backwards through strands from final state
  – Final state specified in terms of strands and terms known by intruder
• Set of rewrite rules governs how search is conducted
• Sensitive to past and future
  – Implemented via adding a “now” bar to strands
NOTION OF STATE IN NPA STRANDS

- A state is a set of strands plus the intruder knowledge (i.e., a set of terms)

  1. Each strand is divided into past and future $[m_1^\pm, \ldots, m_i^\pm | m_{i+1}^\pm, \ldots, m_k^\pm]$
  2. Initial strand $[nil \mid m_1^\pm, \ldots, m_k^\pm]$, final strand $[m_1^\pm, \ldots, m_k^\pm \mid nil]$
  3. The intruder knowledge contains terms $m \notin \mathcal{I}$ and $m \in \mathcal{I}$
     - What the intruder knows in the present (i.e., $t \in \mathcal{I}$)
     - What terms the intruder will learn in the future (i.e., $t \notin \mathcal{I}$)
Protocol Rules and Their Execution

To execute a protocol $\mathcal{P}$ associate to it a rewrite theory on sets of strands as follows. Let $\mathcal{I}$ denote the set of terms known by the intruder, and $K$ a set of positive and negative statements about intruder knowledge

1. $[ L \mid M^- \; L'] \& \{ M \in \mathcal{I}, K \} \rightarrow [ L, M^- \mid L'] \& \{ M \in \mathcal{I}, K \}$
   Moves input messages into the past

2. $[ L \mid M^+ \; L'] \& \{ K \} \rightarrow [ L, M^+ \mid L'] \& \{ K \}$
   Moves output message that are not read into the past

3. $[ L \mid M^+ \; L'] \& \{ M \notin \mathcal{I}, K \} \rightarrow [ L, M^+ \mid L'] \& \{ M \in \mathcal{I}, K \}$
   Joins output message with term in intruder knowledge.

For backwards execution, just reverse
Introducing New Strands

• If we want number of strands we use to be unbounded, need some way of introduce new strands in the backwards search

• Specialize rule 3 using each rule of the protocols $\mathcal{P}$:

\[
R_\mathcal{P} = \mathbb{R} \cup \{ [l_1 | u^+, l_2] \& \{u \notin \mathcal{I}, K\} \rightarrow \\
\{u \in \mathcal{I}, K\} \mid [l_1, u^+, l_2] \in \mathcal{P}\}
\]

• GIves us a natural way of switching between bounded and unbounded sessions
CPSA

• Start with protocol and “skeleton”
  – E.g. a single strand executed by an honest principal
  – Can also specify assumptions about secrecy and uniqueness of terms

• Returns
  – Other strands that must have executed
  – Temporal relationships between nodes in the strands
  – Terms that are secret and/or unique
An Example: NSPK

- Protocol
  1. $A \rightarrow B : PK_{pk(B)}(A, N_A)$
  2. $B \rightarrow A : PK_{pk(A)}, N_A, N_B$
  3. $A \rightarrow B : PK_{pk(B)}(N_A)$

- Skeleton: $[-PK_{pk(B)}(A, N_A), +PK_{pk(A)}, (N_A, N_B), -PK_{B}(N_B)]$

- Shape:
  - $resp_1 = [-PK_{pk(B)}(A, N_A), +PK_{pk(A)}, (N_A, N_B), -PK_{pk(B)}(N_B)]$
  - $init_1 = [+PK_{pk(C)}(A, N_A), -PK_{pk(A)}, (N_A, N_B), +PK_{pk(C)}(N_B)]$
Providing shape info to Maude-NPA

- Maude-NPA can already take strands as goals
- We need to add
  - A secrecy predicate
    - $\text{Secret}(W)$ in goal means discard state if $W$ appears in intruder knowledge
  - Notion of fresh
    - Already have that: use $n(A,r)$ where $r$ is a variable of type “fresh”
  - Notion of precedence among nodes
    - Define $\text{past}(S) =$ length of part of strand appearing before the bar
    - Suppose $s_1[n] < s_2[m]$ appears in a shape
    - Discard state if $\text{past}(s_1) < n$ and $\text{past}(s_2) <= m$. 
Tried Out On NSPK and Yahalom

• Surprise!: no reduction in search space between shapes (with constraints) and strands from shape without constraints

• For NSPK, definite increase in size of search space between using strands from shape as goal and responder strand alone (skeleton)
  – 44 states, 6 iterations vs. 9 states, 5 iterations to find attack
A Closer Look at Constraints

– Maude-NPA already has some constraints that had the same effects as CPSA shapes in these particular examples
  • Any use of a fresh variable must occur after the first occurrence of that variable in the strand that created it
  • Partial order reduction: if there is a negative node, execute it first in backwards search

\[
\text{resp}_1 = [-PK_{pk(B)}, (A, N_A), +PK_{pk(A)}, (N_A, N_B), -PK_{pk(B)}(N_B)]
\]

\[
\text{init}_1 = [+PK_{pk(C)}, (A, N_A), -PK_{pk(A)}, (N_A, N_B), +PK_{pk(C)}(N_B)]
\]

\[
\text{resp}_1[2] < \text{init}_1[2], \quad \text{init}_1[3] < \text{resp}[3]
\]
Yahalom Constraints
A Closer Look At Multiple Strands

• When Maude-NPA is matching a term in the intruder knowledge it can either
  – Use an existing strand, or
  – Create a new strand

• So, if Maude-NPA is trying a match a negative term from the responder strand in NSPK
  – It can use an initiator strand, if it exists
  – Or it can create a new initiator strand
  – If we give Maude-NPA an initiator and a responder strand, this can double the number of states
Conclusion

- It’s easy to code shapes in Maude-NPA
- Maude-NPA can be used to find attacks from shapes
  - May be better not to allow Maude-NPA to create new regular strands
- But, more investigation is needed in order to determine whether or not shapes help reduce size of search space