A Rewriting-based Forwards Semantics for Maude-NPA

Santiago Escobar\textsuperscript{1} \quad Catherine Meadows\textsuperscript{2} \quad José Meseguer\textsuperscript{3}
Sonia Santiago\textsuperscript{1}

\textsuperscript{1}Technical University of Valencia, Valencia, Spain
\textsuperscript{2}Naval Research Laboratory, Washington DC, USA
\textsuperscript{3}University of Illinois at Urbana-Champaign, USA

HotSoS, April 8, 2014
Outline

1. Introduction
2. Maude-NPA: A Peek Under the Hood
3. Forwards Semantics
4. Soundness and Completeness
5. Implementation
6. Conclusion
To prove properties of a program, we need to make use of some logical system.

Different components, different aspects, different properties of a program may require different logical systems.

- This is especially the case in security, a many-faceted problem.

We need to show these different logics can work together, and what is proved in one system remains true in another.

In this talk, I will show how applied this to a formal tool for cryptographic protocol analysis Maude-NPA.
Example: Diffie-Hellman Without Authentication

1. $A \rightarrow B : g^{N_A}$
2. $B \rightarrow A : g^{N_B}$
3. $A$ and $B$ compute $g^{N_A * N_B} = g^{N_B * N_A}$

Well-known attack

1. $A \rightarrow I_B : g^{N_A}$
2. $I_A \rightarrow B : g^{N_I}$
3. $B \rightarrow I_A : g^{N_B}$
4. $I_B \rightarrow A : g^{N_I}$

- $A$ thinks she shares $g^{N_I * N_A}$ with $B$, but she shares it with $I$
- $B$ thinks he shares $g^{N_I * N_A}$ with $A$, but he shares it with $I$
- Commutative properties of $*$ and fact that $(G^X)^Y = G^{X * Y}$ crucial to understanding both the protocol and the attack
Symbolic "Dolev-Yao" Model for Automated Cryptographic Protocol Analysis

- Start with a signature, giving a set of function symbols and variables
- For each role, give a program describing how a principal executing that role sends and receives messages
- Give a set of inference rules and equations the describing the deductions an intruder can make
  - E.g. if intruder knows $K$ and $e(K, M)$, can deduce $M$, or;
  - $d(K, e(K, M)) = M$, where $d$ is a decryption operator
- Assume that all messages go through intruder who can
  - Stop or redirect messages
  - Alter messages
  - Create new messages from already sent messages using inference rules
The Maude-NPA Tool

- A tool to **find or prove the absence** of attacks using **backwards search**
- Analyzes **infinite state systems**
  - **Active intruder**
  - **No** abstraction or **approximation** of nonces
    - If Maude-NPA finds path from initial state to insecure **attack** state, it is a genuine path
- **Unbounded** number of sessions
  - If Maude-NPA terminates without finding path no such path exists
  - Problem is in general undecidable, so Maude-NPA may not terminate
  - Uses search-space pruning mechanisms making termination more likely
- Supports a number of equational theories, including: cancellation (e.g. encryption-decryption), AC, exclusive-or, Diffie-Hellman, bounded associativity, homomorphic encryption over various theories, various combinations, working on including more
- **Executable semantics** based on rewrite rules
Executable Formal Semantics

- Logical system that can also be executed
  - In our case, as state-exploration-based cryptographic protocol analysis tool, Maude-NPA

- By proving things about the logical system, we can prove things about results of the execution

- If we want to make modifications to the tool, we make modifications to the semantics
  - Prove new semantics sound and/or complete to the old
  - Have applied this approach to extend the capabilities of Maude-NPA and prove that these extensions are sound and complete
What Happens When the Process Breaks?

- Require major changes to semantics in order to achieve the functionality we want
  - In our case, we needed to reverse the direction of the execution
  - In this talk, we show how we handled this problem
Outline

1. Introduction
2. Maude-NPA: A Peek Under the Hood
3. Forwards Semantics
4. Soundness and Completeness
5. Implementation
6. Conclusion
Important Tools Used by Maude-NPA: Equational Unification

- Given a signature $\Sigma$ and an equational theory $E$, and two terms $s$ and $t$ built from $\Sigma$:

- A unifier of $s =_E t$ is a substitution $\sigma$ to the variables in $s$ and $t$ s.t. $\sigma s$ can be transformed into $\sigma t$ by applying equations from $E$ to $\sigma s$ and its subterms.

- Example: $\Sigma = \{d/2, e/2, m/0, k/0\}$, $E = \{d(K, e(K, X)) = X\}$. The substitution $\sigma = \{Z \mapsto e(T, Y)\}$ is a unifier of $d(T, Z)$ and $Y$.

- The set of most general unifiers of $s =_E t$ is the set $\Gamma$ s.t. any unifier $\sigma$ is of the form $\rho \tau$ for some $\rho$, and some $\tau$ in $\Gamma$.

- Example: $\{Z \mapsto e(T, Y), Y \mapsto d(T, Z)\}$ mgu’s of $d(T, Z)$ and $Y$.

- Given the theory, can have:
  - at most one mgu (empty theory)
  - a finite number (AC)
  - an infinite number (associativity)

- Problem can also be undecidable.
Important Tools Used by Maude-NPA: Rewrite Rules and Narrowing

- A rewrite theory $\mathcal{R}$ is a triple $\mathcal{R} = (\Sigma, E, R)$, with:
  - $\Sigma$ a signature
  - $(\Sigma, R)$ a set of rewrite rules of the form $t \rightarrow s$
    - e.g. $e(K_A, N_A; X) \rightarrow e(K_B, X)$
  - $E$ a set of equations of the form $t = s$

- Rewriting: If $t$ is a ground term (no variables), $t \rightarrow_{\sigma, R, E} s$ if there are
  - a non-variable position $p \in \text{Pos}(t)$;
  - a rule $l \rightarrow r \in R$;
  - a substitution $\sigma$ (modulo $E$) such that $t\theta =_E l$ and $s = \theta(t[r]_p)$

- Narrowing: If $t$ is a symbolic term (may have variables) $t \sim_{\sigma, R, E} s$ if there are
  - a non-variable position $p \in \text{Pos}(t)$;
  - a rule $l \rightarrow r \in R$;
  - a unifier $\sigma$ (modulo $E$) of $t|p =_E ?l$ such that $s = \sigma(t[r]_p)$. 
Comparison of Rewriting and Narrowing

- In favor of narrowing
  - Narrowing wrt symbolic terms means you can handle a possibly infinite number of terms in one narrowing step
  - For that reason, good for reasoning about infinite state systems

- In favor of rewriting
  - Rewriting simpler and faster than narrowing
  - Software support for rewriting (in particular, Maude itself!)

- Conclusion: Use narrowing when it can most benefit you, rewriting otherwise
Protocols Specified Using Strand Spaces

- Maude-NPA uses concept of **strand spaces** due to Thayer, Herzog, and Gutmann (2001)
- A **strand** is a sequence of messages representing the actions of a principal executing a **role**, or of an intruder making a computation
  - A **negative** term represents a message received by a principal
  - A **positive** term represents a message sent by a principal
- Example: Initiator’s strand in DH
  
  :: r, r’ :: [nil , +(A ; B ; exp(g,n(A,r))), -(A ; B ; XE),
  +(e(exp(XE,n(A,r)),sec(A,r’))), nil]
- Example: Attacker exponentiation strand in DH
  
  :: nil :: [ nil | -(GE), -(NS), +(exp(GE,NS)), nil ]
- Note: Capital letters stand for logical variables, terms inside “::” are special variables used to construct nonces
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
   \[ \[ m_1^\pm, \ldots, m_i^\pm \mid m_{i+1}^\pm, \ldots, m_k^\pm \] \]
2. Initial strand \[ [ \text{nil} | m_1^\pm, \ldots, m_k^\pm ] \], final strand \[ [ m_1^\pm, \ldots, m_k^\pm | \text{nil} ] \]
3. The intruder knowledge contains terms \( m \notin \mathcal{I} \) and \( m \in \mathcal{I} \)
   \[ \{ t_1 \notin \mathcal{I}, \ldots, t_n \notin \mathcal{I}, s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I} \} \]
4. Initial intruder knowledge \( \{ t_1 \notin \mathcal{I}, \ldots, t_n \notin \mathcal{I} \} \),
   final intruder knowledge \( \{ s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I} \} \)
State in which initiator has sent first message, attacker has learned that message, and attacker will learn secret value in future

\[
SS \& :: r, r' :: [\text{nil}, +(a; b; \text{exp}(g, n(a, r))) \mid \neg(a; b; XE), \neg (e(\text{exp}(XE, n(a, r)), \text{sec}(a, r'))), \text{nil}] \& \{\text{exp}(g, n(a, r) \text{ inI}, \text{sec}(a, r') \text{ notinI}, K}
\]

Note that it is possible (and expected) for states to contain variables

Since XE hasn’t been received yet, we don’t know what it is
Maude-NPA Backwards Semantics

- Expressed in terms of forwards executing rewrite rules
- Rewrite rule: a rule of the form \( \ell \rightarrow r \) meaning “replace expression \( \ell \) with expression \( r \)
  
  1. \( SS \& [ L \mid M^- \mid L'] \& \{ M \in \mathcal{I}, K \} \rightarrow SS \& [ L \mid M^- \mid L'] \& \{ M \in \mathcal{I}, K \} \)
     Moves input messages into the past
  
  2. \( SS \& [ L \mid M^+ \mid L'] \& \{ K \} \rightarrow SS \& [ L \mid M^+ \mid L'] \& \{ K \} \)
     Moves output message that are not read into the past
  
  3. \( SS \& [ L \mid M^+ \mid L'] \& \{ M \notin \mathcal{I}, K \} \rightarrow SS \& [ L \mid M^+ \mid L'] \& \{ M \in \mathcal{I}, K \} \)
     Joins output message with term in intruder knowledge.
  
  4. \( SS \& [ l_1 \mid u^+] \& SS \& \{ u \notin \mathcal{I}, K \} \rightarrow \{ u \in \mathcal{I}, K \} \) where \( [ l_1 \mid u^+] \) is a prefix of a strand in the protocol specification
     Introduces new strand or prefix of strand, and joins output message with term in intruder knowledge.

- To obtain backwards semantics, just reverse the arrows!
Begin by specifying an attack state pattern

- An **attack state pattern** describes an insecure state and may contain variables.
- Example: Attack state in which responder $B$ has finished execution of protocol, apparently with initiator $A$, but attacker knows the secret

\[
:: r :: [\text{nil}, -(a ; b ; XE), +(a ; b ; \exp(g,n(b,r))), \\
-(e(\exp(XE,n(b,r)),\sec(a,r')))) | \text{nil}] \\
| \sec(a,r') \in I
\]

- Use backward narrowing via the rewrite rules, to determine if an initial state can be reached.
- If you reach an initial state, you will have constructed a path to an instance of the attack pattern.
When We May Need Forward Execution

- **Practical Reasons**
  - Narrowing is powerful, but computationally expensive
  - If you execute forwards instead of backwards, states will contain no variables, and you can use rewriting instead of narrowing
  - Example: Suppose that you want to simulate protocol to see if it can reach a final state in absence of attackers
    - Narrowing is overkill

- **Theoretical Reasons**
  - In many cases, it is more natural to reason about forward rather than backwards execution
  - We found this when developing a theory of indistinguishability for Maude-NPA
Important: Forwards semantics must be sound and complete with respect to backwards semantics

- Allows us to switch between forwards and backwards semantics
- We use simulation to verify protocol specified correctly using forwards semantics, but verify security using backwards semantics
- We use forwards semantics to formulate our indistinguishability framework, but prove indistinguishability using backwards semantics
Why Can’t We Just Execute the Backwards Semantics Forwards?

- Maude-NPA already has a forwards semantics, obtained by reversing the backwards semantics
  - Why can’t we just use that and save ourselves a lot of work?
- Backwards semantics contains too much information about the future!
  - Initial state contains all strands and intruder knowledge used to reach the final state
  - Part of the strand after the bar may need to contain variables
    - This is problematic for rewriting
How We Represent States in the Forwards Semantics

- No variables allowed in state
- Only information about the past allowed, not the future
  - Terms $t \notin I$ can't appear, since they represent future knowledge of the intruder
  - Information after the bar in a strand can't appear, since it represents future execution
Some Rules in the Forwards Semantics

- Adding a positive term the intruder doesn’t know already to a strand

\[
\forall [u_1^+, \ldots, u_{j-1}^+, u_j^+, u_{j+1}^+, \ldots, u_n^+] \in P \land j > 1 : \\
\{ SS \& \{ IK \} \& [u_1^+, \ldots, u_{j-1}^+] \& \langle N \rangle \} \\
\rightarrow \\
\{ SS \& \{ u_j^+ \uparrow^M_N \in I, IK \} \& [u_1^+, \ldots, u_{j-1}^+, (u_j^+ \uparrow^M_N)^+] \& \langle M \rangle \} \\
\text{IF } (u_j^+ \uparrow^M_N \in I) \notin IK
\]  

(1)

- Adding a strand that begins with a positive term the intruder doesn’t know already

\[
\forall [u_1^+, \ldots, u_n^+] \in P : \\
\{ SS \& \{ IK \} \& \langle N \rangle \} \rightarrow \{ SS \& [(u_1^+ \uparrow^M_N)^+] \& \{ IK \} \& \langle M \rangle \}
\]  

(2)
Lifting Relation

Definition (Lifting relation)
Given a symbolic $P$-state $S$ and a ground state $s$ we say that $s$ lifts to $S$, or that $S$ instantiates to $s$ with a grounding substitution $\theta : (Var(S) - \{SS, IK\}) \rightarrow T_{\Sigma}$, written $S >^\theta s$ iff

- for each strand :: $r_1, \ldots, r_m :: [u_1^\pm, \ldots u_{i-1}^\pm | u_i^\pm, \ldots, u_n^\pm]$ in $S$, there exists a strand $[v_1^\pm, \ldots v_{i-1}^\pm]$ in $s$ such that $\forall 1 \leq j \leq i - 1$, $v_j =_{E_P} u_j^\theta$.

- for each positive intruder fact $w \in I$ in $S$, there exists a positive intruder fact $w' \in I$ in $s$ such that $w' =_{E_P} w^\theta$, and

- for each negative intruder fact $w \notin I$ in $S$, there is no positive intruder fact $w' \in I$ in $s$ such that $w' =_{E_P} w^\theta$. 
Example of Lifting Relation

- **Symbolic state**
  
  $SS \land (r, r') :: [\text{nil}, +(a; b; \exp(g,n(a,r))) | -(a; b; XE), +\left(\exp(XE,n(a,r)), \sec(a,r')\right)), \text{nil}] \land \{\exp(g,n(a,r)) \in \mathcal{I}, \sec(a,r') \notin \mathcal{I}, K\}$

- **Ground State**

  $[+(a; b; \exp(g,n(a,1)))] \land \{\exp(g,n(a,1)) \in \mathcal{I}, a \in \mathcal{I}, b \in \mathcal{I}, a; b; \exp(g,n(a,1)) \in \mathcal{I}\}$

- **Lifting via $\theta$**

  $\theta = \{r \rightarrow 1\}$
Soundness and Completeness Theorems

**Theorem (Completeness)**

*Given a protocol $\mathcal{P}$, two ground states $s, s_0$, a symbolic $\mathcal{P}$-state $S$, a substitution $\theta$ s.t. (i) $s_0$ is an initial state, (ii) $s_0 \rightarrow^n s$, and (iii) $S >^\theta s$ then there exist a symbolic initial $\mathcal{P}$-state $S_0$, two substitutions $\mu$ and $\theta'$, and $k \leq n$, s.t. $S_0 \leftarrow^k_\mu S$, and $S_0 >^\theta' s_0$.***

**Theorem (Soundness)**

*Given a protocol $\mathcal{P}$, two symbolic $\mathcal{P}$-states $S_0, S'$, an initial ground state $s_0$ and a substitution $\theta$ s.t. (i) $S_0$ is a symbolic initial state, and (ii) $S_0 \leftarrow^* S'$, and (iii) $S_0 >^\theta s_0$ then there exist a ground state $s'$ and a substitution $\theta'$, s.t. (i) $s_0 \rightarrow^* s'$, and (ii) $S' >^\theta' s'$.***
Proof of Soundness and Completeness

- (Lifting Lemma) Given rewriting step \( s' \rightarrow s \) and lifting relation \( S \triangleright \theta s \) we can complete the diagram with \( S' \) as follows:

\[
\begin{array}{c}
S' & \sim & S \\
\parallel & \downarrow & \downarrow \\
\triangleleft \triangleright & \triangleright \theta & \triangleright \theta \\
\downarrow & \downarrow & \downarrow \\
s' & \rightarrow & s
\end{array}
\]

**Soundness:** Given a forward rewriting sequence iterate lifting lemma to get corresponding backwards narrowing sequence

- (Grounding Lemma) Given narrowing step \( S \leftarrow \sim S' \) and lifting relation \( S \triangleright \theta s \) we can complete the diagram with an \( s' \) as follows:

\[
\begin{array}{c}
S & \sim & S' \\
\triangleright \theta & \parallel & \parallel \triangleright \theta \\
\downarrow & \downarrow & \downarrow \\
s & \rightarrow & s'
\end{array}
\]

**Completeness:** Given a backwards narrowing sequence iterate grounding lemma to get corresponding forwards rewriting sequence
Implemented rewriting-based forward semantics in Maude

Maude’s support for rewriting made it possible to do this very quickly

Implemented some heuristic state space reduction techniques to reduce state space explosion
  - Plan to investigate these further in the future, in particular adapting Maude-NPA’s state space reduction techniques to a forwards setting
  - Expect soundness and completeness result to help us here

Applied it to various protocols in the literature, tool was able to reproduce attacks found by Maude-NPA
Conclusion

- We started out wanting a theoretical tool to help us reason about indistinguishability, but we wound up with:
  - A novel executable semantics for model-checking cryptographic protocols
  - A new logical foundation for Maude-NPA, designed for model-checking
  - The beginnings of a new crypto protocol model-checker

- And we got a new theoretical tool to help us reason about indistinguishability!