The Maude-NRL Protocol Analyzer

Santiago Escobar[1]
Catherine Meadows[2]
José Meseguer[3]
Sonia Santiago [1] Ralf Sasse[3]

[1]Universidad Politécnica de Valencia (Spain)
[2]Naval Research Laboratory (USA)
[3]University of Illinois at Urbana-Champaign (USA)

FOSAD 2011, Bertinoro, August 29- September 3

Purpose of These Lectures

- Introduce you to a particular protocol tool for crypto protocol analysis, Maude-NPA
  - Tool for automatic analysis of crypto protocols that takes into account equational theories of crypto operators
  - Based on unification and rewrite rules
- On the way, point out connections between research on the tool and open problems in crypto protocol analysis, rewriting logic, and unification

Outline

Example: Diffie-Hellman Without Authentication

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

Well-known attack

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

- 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

"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 the describing the deductions an intruder can make
    - E.g. if intruder knows K and $e(K, M)$, can deduce M
- 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
- This problem well understood since about 2005

Background

- Crypto protocol analysis with the standard free algebra model (Dolev-Yao) well understood.
- But, not adequate to deal with protocols that rely upon algebraic properties of cryptosystems
    1. Cancellation properties, encryption-decryption
    2. Abelian groups
    3. Diffie-Hellman (exponentiation, Abelian group properties)
    4. Homomorphic encryption (distributes over an operator with also has algebraic properties, e.g. Abelian group)
    5. Etc. ..,
- In many cases, a protocol uses some combination of these

Goal of Maude-NPA

Provide tool that

- can be used to reason about protocols with different algebraic properties in the unbounded session model
- supports combinations of algebraic properties to the greatest degree possible

Our approach

- Use rewriting logic as general theoretical framework
  - crypto protocols are specified using rewrite rules
  - algebraic identities as equational theories
- Use narrowing modulo equational theories as a symbolic reachability analysis method
- Combine with state reduction techniques of Maude-NPA's ancestor, the NRL Protocol Analyzer (grammars, optimizations, etc.)
- Implement in Maude programming environment
  - Rewriting logic gives us theoretical framework and understanding
  - Maude implementation gives us tool support

Maude-NPA

- 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
  - Unbounded number of sessions
- Intruder and honest protocol transitions represented using strand space model.
- So far supports a number of equational theories: cancellation (e.g. encryption-decryption), AC, exclusive-or, Diffie-Hellman, bounded associativity. homormorphic encryption over a free theory, various combinations, working on including more

Outline

A Little Background on Unification

- Given a signature Σ and an equational theory E, and two terms s and t built from Σ:
- A unifier of $s =_E ?t$ is a substitution σ to the variables in s and t s.t. σs can be transformed into σt by applying equations from E to σs 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 $d(K, Z)$ and Y.
- The set of most general unifiers of $s = ?t$ is the set Γ s.t. any unifier σ is of the form $\rho\tau$ for some ρ, and some τ in Γ.
- 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)
- Unification problem in general undecidable

Rewriting Logic in a Nutshell

A rewrite theory $\mathcal{R}$ is a triple $\mathcal{R} = (\Sigma, E, R)$, with:

- (Σ, R) a set of rewrite rules of the form $t \to s$
  e.g. $e(K_A, N_A; X) \to e(K_B, X)$
- (Σ, E) a set of equations of the form $t = s$
  e.g. $d(K, e(K, Y)) = Y$

Intuitively, $\mathcal{R}$ specifies a concurrent system,
whose states are elements of the initial algebra $T_{\Sigma/E}$ specified by (Σ, E), and
whose concurrent transitions are specified by the rules R.
Narrowing gives us the rules for executing transitions concurrently.

Narrowing and Backwards Narrowing

Narrowing: $t \leadsto_{\sigma,R,E} s$ if there is

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r \in R$;
- a unifier σ (modulo E) of $t|p =_E ?l$ such that $s = \sigma(t[r]_p)$.

Example:

- $R = \{\ X \to d(k, X)\ \}$, $E = \{\ d(K, e(K, Y)) = Y\ \}$
- $e(k,\ t) \leadsto_{\emptyset,R,E} d(k, e(k, t)) =_E t$

Backwards Narrowing: narrowing with rewrite rules reversed

A Warning About Narrowing

- Full narrowing (narrowing in every possible non-variable location) is often inefficient and even nonterminating
- We need to construct our rewrite systems so that efficient narrowing strategies can be chosen
- Maude-NPA has led to some major advances in this area

Narrowing Reachability Analysis

Narrowing can be used as a general deductive procedure for solving reachability problems of the form

$$(\exists \vec{x})\ t_1(\vec{x}) \longrightarrow t'_1(\vec{x})\ \wedge \ldots \wedge\ t_n(\vec{x}) \longrightarrow t'_n(\vec{x})$$

in a given rewrite theory.

- The terms t_i and t'_i denote sets of states.
- For what subset of states denoted by t_i are the states denoted by t'_i reachable?
- No finiteness assumptions about the state space.
- Maude-NPA rewrite system supports topmost narrowing for state reachability analysis
  - Narrowing steps only need to be applied to entire state

E-Unification

- In order to apply narrowing to search, need an E unification algorithm
- Two approaches:
  1. Built-in unification algorithms for each theory and combination of theories.
  2. Hybrid approach with $E = \Delta \uplus B$
- Hybrid Approach
  - B has built-in unification algorithm
  - Δ confluent and terminating rules modulo B
    - Confluent: Always reach same normal form modulo B, no matter in which order you apply rewrite rules
    - Terminating: Sequence of rewrite rules is finite
- This allows us to use narrowing as a general method for E-unification
- But still need to develop new narrowing methods for theories of interest to crypto protocol verification

Outline

Uses Strand Space Notation

- Strand spaces: popular model introduced by Thayer, Herzog, and Guttman
- Each local execution, or session of an honest principal represented by sequence of positive and negative terms called a strand.
  - Terms made up of variables and function symbols
  - Negative term stand for received message, positive terms stand for sent messages
  - Example:
    $[+(pke(B, N_A; A)),\ -(pke(A, N_A; N_B)),\ +(pke(B, N_B))]$
- Each intruder computation also represented by strand
  - Example: $[-(X), +(pke(A, X))]$

Basic Structure of Maude-NPA

- Uses modified strand space model
- Each local execution and each intruder action represented by a strand, plus a marker denoting the current state
  - Searches backwards through strands from final state
  - Set of rewrite rules governs how search is conducted
  - Sensitive to past and future
- Grammars used to prevent infinite loops
- Learn-only-once rule says intruder can learn term only once
- When an intruder learns term in a backwards search, tool keeps track of this and doesn't allow intruder to learn it again
- Other optimization techniques used to reduce other infinite behavior and to cut down size of search space

What We Need to Represent

- Maude-NPA's use of backwards search means we have incomplete picture of what intruder learned in past. But we need the concrete moment when the intruder learns something:
- Notion of the present
  - What the intruder knows in the present (i.e., $t{\in}\mathcal{I}$)
  - Where the honest principals are in the present (strands)
- Notion of the future
  - What terms the intruder will learn in the future (i.e., $t{\notin}\mathcal{I}$)

$$\longleftarrow\!\!\!\!-\!\!\!-\!\!\!-\!\!\!-\,|\,-\!\!\!-\!\!\!-\!\!\!-\!\!\!-$$

$$t{\notin}\mathcal{I} \qquad\qquad t{\in}\mathcal{I}$$

How Protocols Are Specified in Maude-NPA

- Represent protocols and intruder actions using strands
- Terms in strands obey an equational theory specified by the user
- Terms in strands of different sorts, mostly defined by user
- Special sort Fresh
  - Terms of sort *Fresh* are always constant (used by nonces)
  - Strand annotated with fresh terms generated by the strand

$$:: r :: [+(pke(B, n(A, r); A)), -(pke(A, n(A, r); NB)), +(pke(B, NB))]$$

The 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 $[\ nil\ |\ m_1^{\pm},\ \ldots,\ m_k^{\pm}\ ]$, final strand $[\ m_1^{\pm},\ \ldots,\ m_k^{\pm}\ \ |\ nil\ ]$
    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 $\{\ s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I}\ \}$

Outline

Protocol Rules and Their Execution With Strands Already in State

To execute a protocol $\mathcal{P}$ associate to it a rewrite theory on sets of strands as follows. Let $\mathcal{I}$ informally denote the set of terms known by the intruder, and K the facts known or unknown by the intruder

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 an unbounded number of strands, need some way of introducing new strands in the backwards search
- Specialize rule r3 using each strand $[\ l_1,\ u^+,\ l_2\ ]$ of the protocol $\mathcal{P}$:

$$[\ l_1 \mid u^+] \ \&\ \{u \notin \mathcal{I}, K\} \rightarrow \{u \in \mathcal{I}, K\}$$

- Gives us a natural way of switching between bounded and unbounded sessions
  - Put a bound on the number of times r3 could be invoked with non-intruder strands

Reachability Analysis

- Backwards narrowing protocol execution defines a backwards reachability relation $St \rightsquigarrow^{*}_{\mathcal{P}} St'$
- In initial step, prove lemmas that identify certain states unreachable
- Specify a state describing the attack state, including a set of final strands plus terms $m{\notin}\mathcal{I}$ and $m{\in}\mathcal{I}$
- Execute the protocol backwards to an initial state, if possible
- For each intermediate state found, check if it has been proved unreachable and discard if it is

Outline

Introduction

- Crypto protocols don't exist in isolation, but often rely upon one another
- Protocols that work correctly in one environment may fail when they are composed with new protocols in new environments
  - The properties they guarantee are not quite appropriate for the new environment
  - The composition itself is mishandled
- Research has concentrated on parallel composition, but sequential composition is where most of the problems lie
- The problem is in providing a specification and verification environment that supports sequential composition

Motivating examples

One-parent, one-child protocol composition

- The parent protocol can have only one child instance
- Example: NSL with Distance Bounding (DB)
  - NSL is used to agree on N_A
  - DB reveals N_A, so it cannot be used with the same N_A more than once

Motivating examples

One-parent, one-child protocol composition

- The parent protocol can have only one child instance
- Example: NSL with Distance Bounding (DB)
  - NSL is used to agree on N_A
  - DB reveals N_A, so it cannot be used with the same N_A more than once

One-parent, many-children protocol composition

- The parent protocol has an arbitrary number of child instances
- Example: NSL with Key Distribution
  - The parent protocol generates a master key
  - The child protocol uses the master key and generates a session key

Motivating examples: NSL-DB

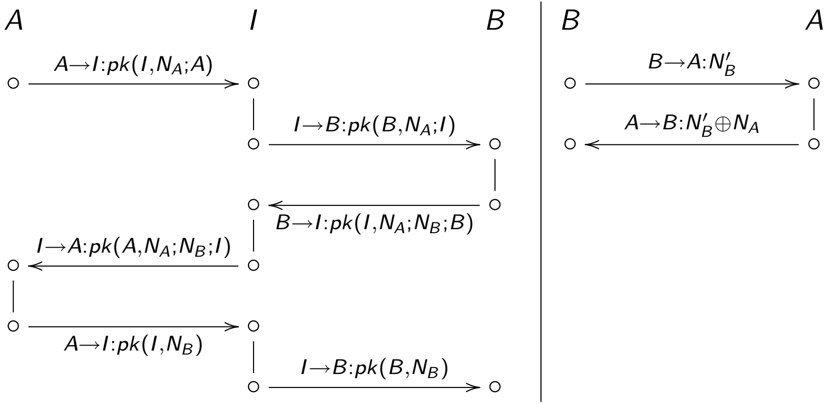
One-parent, one-child: NSL with Distance Bounding (DB)(*)

Alice claims that she is a certain distance δ_{AB} from *Bob*, and *Bob* wants to check this

- Needham-Schroeder-Lowe Public Key Protocol (NSL)
  1. $A \rightarrow B : pke(B, N_A; A)$
  2. $B \rightarrow A : pke(A, N_A; N_B; B)$
  3. $A \rightarrow B : pke(B, N_B)$
- At the end, A and B know that they share two secrets, N_A and N_B. They will use N_A for distance bounding (DB)
  4. $B \rightarrow A : N'_B$
  5. $A \rightarrow B : N_A \oplus N'_B$
- Bob checks time it takes for round trip, and uses it to put upper bound on distance δ_{AB} of Alice

(*) Guttman, Herzog, Swarup, and Thayer, "Strand spaces: From Key Exchange to Secure Location," Workshop on Event-Based Semantics, 2008

Attack on NSL-DB



Bob concludes: N_A, N_B shared with I, and I is distance δ_{AB} from him.

What happened?

- NSL guarantees origin of responder nonce only when responder is honest.
- If responder dishonest, Bob could have got the nonce from someone else.
- What a distance bounding protocol needs is the following:
    - If sender of authenticated response is honest, then sender of rapid response is the same individual.
    - If sender of rapid response is honest, then sender of authenticated response is the same individual.
- One solution: alter rapid response so that composition works.

Fixing the NSL-DB protocol

1. Needham-Schroeder-Lowe Public Key Protocol (NSL)
   1. $A \rightarrow B : pke(B, N_A; A)$
   2. $B \rightarrow A : pke(A, N_A; N_B; B)$
   3. $A \rightarrow B : pke(B, N_B)$
2. Distance bounding using N_A
   4. $B \rightarrow A : N'_B$
   5. $A \rightarrow B : h(A, N_A) \oplus N'_B$

- Alice hashes her nonce with her identity before responding
- If the sender of the rapid response is honest, he will hash with his own identity.

Motivating examples: NSL-KD

One-parent, many-children: NSL with Key Distribution (KD)

- Needham-Schroeder-Lowe Public Key Protocol (NSL)
  1. $A \rightarrow B : pke(B, N_A; A)$
  2. $B \rightarrow A : pke(A, N_A; N_B; B)$
  3. $A \rightarrow B : pke(B, N_B)$
- N_A and N_B will be used for key distribution
- The initiator of the session key protocol can be the child of either the initiator or responder of the NSL protocol

  4. $A \rightarrow B : \{Sk_A\}_{h(N_A,N_B)}$
  5. $B \rightarrow A : \{Sk_A; N'_B\}_{h(N_A,N_B)}$
  6. $A \rightarrow B : \{N'_B\}_{h(N_A,N_B)}$

  4. $B \rightarrow A : \{Sk_B\}_{h(N_A,N_B)}$
  5. $A \rightarrow B : \{Sk_B; N'_A\}_{h(N_A,N_B)}$
  6. $B \rightarrow A : \{N'_A\}_{h(N_A,N_B)}$

Strand Annotations

1. Separate strands for parent and child
2. Annotate strands with role and input and output parameters.

```
prot NSL is
 strand [init]
 :: r :: [{A,B} | +(pk(B,n(A,r);A)), -(pk(A,n(A,r);NB;B)), +(pk(B,NB)),
          {A,B,n(A,r),NB}] .
 strand [resp]
 :: r :: [{A,B} | -(pk(B,NA;A)), +(pk(A,NA;n(B,r);B)), -(pk(B,n(B,r)),
          {A,B,NA,n(B,r)}] .
endp
prot DB is
 strand [init]
 :: r :: [ {B,A,NA} | +(n(B,r)), -(NA * n(B,r)), {A,B,NA,n(B,r)}] .
 strand [resp]
 :: nil :: [ {B,A,NA} | -(NB'), +(NB' * NA), {A,B,NA,NB'}] .
endp
```

Specifying Composition

3. Composition is performed by unifying appropriate output parameters of parent strand with input parameters of child strand
4. Composition section tells you what output terms unified with what input terms, and whether composition is 1-1 or 1-many
   - **One-to-one composition: NSL-DB**

```
prot NSL-DB is NSL ; DB
  NSL.init {A,B,NA,NB} ; {B,A,NA} DB.resp [1-1] .
  NSL.resp {A,B,NA,NB} ; {B,A,NA} DB.init [1-1] .
endp
```

Specifying Composition

3. Composition is performed by unifying appropriate output parameters of parent strand with input parameters of child strand
4. Composition section tells you what output terms unified with what input terms, and whether composition is 1-1 or 1-many
   - **One-to-one composition: NSL-DB**

```
prot NSL-DB is NSL ; DB
  NSL.init {A,B,NA,NB} ; {B,A,NA} DB.resp [1-1] .
  NSL.resp {A,B,NA,NB} ; {B,A,NA} DB.init [1-1] .
endp
```

   - **One-to-many composition: NSL-KD**

```
prot NSL-KD is NSL ; KD
  NSL.init {A,B,NA,NB} ; {B,A,h(NB,NA)} KD.resp [1-*] .
  NSL.init {A,B,NA,NB} ; {A,B,h(NA,NB)} KD.init [1-*] .
  NSL.resp {A,B,NA,NB} ; {B,A,h(NB,NA)} KD.init [1-*] .
  NSL.resp {A,B,NA,NB} ; {A,B,h(NA,NB)} KD.resp [1-*] .
endp
```

Model for One-to-One Composition

for each one-to-one composition $\{a\{\overrightarrow{O}\};\{\overrightarrow{I}\}b\}\,[1{-}1]$ with strand definitions $[\{\overrightarrow{I_a}\}, \overrightarrow{a}, \{\overrightarrow{O_a}\}]and[\{\overrightarrow{I_b}\}, \overrightarrow{b}, \{\overrightarrow{O_b}\}]$ and unifiers σ_a, σ_{ab} s.t. $\overrightarrow{O_a} =_{E_\mathcal{P}} \sigma_a(\overrightarrow{O})$ and $\sigma_a(\overrightarrow{I}) =_{E_\mathcal{P}} \sigma_{ab}(\overrightarrow{I_b})$, add :

$$
\begin{array}{l}
SS \,\&\, [\overrightarrow{a} \mid \{\overrightarrow{O_a}\}] \,\&\, [nil \mid \{\sigma_{ab}(\overrightarrow{I_b})\}, \sigma_{ab}(\overrightarrow{b})] \,\&\, IK \\
\to SS \,\&\, [\overrightarrow{a}, \{\overrightarrow{O_a}\} \mid nil] \,\&\, [\{\sigma_{ab}(\overrightarrow{I_b})\} \mid \sigma_{ab}(\overrightarrow{b})] \,\&\, IK
\end{array}
\tag{1}
$$

Case in which parent already present in right-hand state

$$
\begin{array}{l}
SS \,\&\, [\overrightarrow{a} \mid \{\overrightarrow{O_a}\}] \,\&\, [nil \mid \{\sigma_{ab}(\overrightarrow{I_b})\}, \sigma_{ab}(\overrightarrow{b})] \,\&\, IK \\
\to SS \,\&\, [\{\sigma_{ab}(\overrightarrow{I_b})\} \mid \sigma_{ab}(\overrightarrow{b})] \,\&\, IK
\end{array}
\tag{2}
$$

Case in which parent not already present in right-hand state

Model for One-to-Many Composition

For each one-to-many composition $\{a\{\overrightarrow{O}\}; \{\overrightarrow{I}\}b\}\,[1{-}*]$ with strand definitions $[\{\overrightarrow{I_a}\}, \overrightarrow{a}, \{\overrightarrow{O_a}\}]$ and$[\{\overrightarrow{I_b}\}, \overrightarrow{b}, \{\overrightarrow{O_b}\}]$ and unifiers σ_a, σ_{ab} s.t. $\overrightarrow{O_a} =_{E_{\mathcal{P}}} \sigma_a(\overrightarrow{O})$ and $\sigma_a(\overrightarrow{I}) =_{E_{\mathcal{P}}} \sigma_{ab}(\overrightarrow{I_b})$, add to the previous rules :

$$\begin{aligned} & SS \,\&\, [\overrightarrow{a} \mid \{\overrightarrow{O_a}\}] \,\&\, [nil \mid \{\sigma_{ab}(\overrightarrow{I_b})\}, \sigma_{ab}(\overrightarrow{b})] \,\&\, IK \\ \rightarrow\ & SS \,\&\, [\overrightarrow{a} \mid \{\overrightarrow{O_a}\}] \,\&\, [\{\sigma_{ab}(\overrightarrow{I_b})\} \mid \sigma_{ab}(\overrightarrow{b})] \,\&\, IK \end{aligned} \tag{3}$$

Composition leaving parent available to compose with more children

- Rule 3 describe the interim transitions of one-to-many composition
- Rules 1 and 2 describe the final transition

Example of Backwards Search: NSL-KD

Example one-to-many composition: NSL-KD

```
NSL.init {A,B,NA,NB} ; {A,B,h(NA,NB)} KD.init [1-*] .
```

Suppose we have state with two child responder strands:

```
:: r'' :: [  {A1,B1,h(NA1,NB1} | +(e(h(NA1,NB1),skey(A,r'')), ... ] .
:: r' :: [  {A2,B2,h(NA2,NB2} | +(e(h(NA2,NB2),skey(,r')),... ] .
```

Apply Formula 2 to the first strand to obtain

```
:: r'' :: [  nil | {A1,B1,h(n(A1,r),NB1} ,
      +(e(h(n(A1,r),NB1),skey(A,r'')),... ] .
:: r' :: [  {A2,B2,h(NA2,NB2} | +(e(h(NA2,NB2),skey(A,r')), ... ] .
:: r :: [ +pke(B1,A1; n(A1,r)) , ...  | {A1, B1, n(A1,r) , NB1} ]
```

Apply Formula 3 to the second and third strands to obtain

```
:: r'' :: [  nil | {A1,B1,h(n(A1,r),NB1} ,
      +(e(h(n(A1,r),NB1),skey(A,r'')),... ] .
:: r' :: [  nil | {A1,B1,h(N(A,r_ ,NB1} |
      +(e(h(n(A1,r),NB1),skey(A,r')), ... ] .
:: r :: [ +pke(B1,A1; n(A1,r)) , ...  | {A1, B1, n(A1,r) , NB1} ]
```

Protocol Composition by Protocol Transformation

- Sound and complete protocol transformation to support the Composition Execution Model without re-implementing the Maude-NPA
  1. For each composition
     - Transform input parameters $\{\overrightarrow{I_b}\}$ into input message $-(\overrightarrow{I_b})$,
     - Transform output parameters $\{\overrightarrow{O_a}\}$ into output message $+(\sigma_{ab}(\overrightarrow{I_b}))$.

Protocol Composition by Protocol Transformation

- Sound and complete protocol transformation to support the Composition Execution Model without re-implementing the Maude-NPA
  1. For each composition
     - Transform input parameters $\{\overrightarrow{I_b}\}$ into input message $-(\overrightarrow{I_b})$,
     - Transform output parameters $\{\overrightarrow{O_a}\}$ into output message $+(\sigma_{ab}(\overrightarrow{I_b}))$.
  2. Identify each composition with a `Fresh` variable
     - Composition identifier exchanged between strands via messages of the form $role_j(r)$
     - Make use of fact that `Fresh` variables parametrizing different strands can't be unified to implement both one-to-one and one-to-many composition
- Proof of soundness and completeness in Escobar, S., Meadows, C., Meseguer, J., Santiago, S.: Sequential Protocol Composition in Maude-NPA. Tech. Report DSIC-II/06/10, U. Politecnica de Valencia (June 2010) See `http://maude.cs.uiuc.edu/tools/Maude-NPA/`

What We Have

- Sequential composition of protocols supported in Maude-NPA
- Syntax and operational semantics extends in a natural way
- Sequential composition implemented via a protocol transformation, without having to re-implement Maude-NPA
    - To be done: user input via syntax, not protocol transformation
- Have applied Maude-NPA to protocols described in this lecture
    - Output available at `http://maude.cs.uiuc.edu/tools/Maude-NPA/`

Outline

What Maude-NPA Needs In a Unification Algorithm

1. Reasonably efficient
2. Supports large number of theories and combinations of theories
3. Results of unification support syntactic checks on state information for state space reduction techniques

- We find that so far, variant narrowing supports these requirements the best

Narrowing for $\Delta \uplus B$

1. Start with a decomposition $\Delta \uplus B$
2. Find a rewrite rule $\ell \to r \in \Delta$, a non-variable location p of $s =?t$
3. Attempt to unify ℓ with $s =?t|_p$
4. For each member θ of a set of mgus Θ, replace $s =?t|_p\theta$ with $r\theta$ to obtain $s' =?t'$
5. Then either:
   - Attempt to solve $s' =?t'$ modulo B or;
   - Apply steps 1-5 again on $s' =?t'$

- When B is the empty theory, and Δ terminating and confluent wrt B, the basic narrowing strategy is complete and terminating
  - Avoid narrowing on subterms introduced by previous narrowing step

Example

- $\Delta = \{d(K, e(K, X)) \to X\}$, $B = \phi$
- Solve $d(k, V) =?Z$
    - $Z \mapsto d(k, V)$ is first solution
    - For next, note that $d(k, V)$ unifies with $(d(K, e(K, X))$ via $\sigma = \{V \mapsto e(k, X), K \mapsto k\}$.
    - Replace $\sigma d(k, V) = d(k, e(k, X))$ with $\sigma X = X$ and we're done.
    - No more possible solutions.

Things Begin to Go Wrong when $B = AC$

- Basic narrowing is not complete
- Full narrowing (narrowing at every possible non-variable location) doesn't terminate
- But $B = AC$ is extremely important for crypto protocol analysis
  1. Diffie-Hellman
  2. Exclusive-Or
  3. Homomorphic Encryption Over Abelian Groups

Finite Variant Property to the Rescue

- Introduced by Comon and Delaune
- We say $\Delta \uplus B$ has the finite variant property iff, for every term t, there is a finite set of substitutions Σ such that, for every substitution θ, there is a substitution ρ and a $\sigma \in \Sigma$ such that $t\theta \downarrow_\Delta =_B t\sigma \downarrow_\Delta \rho$.
  - In other words, every term has a finite set of irreducible variants
  - Definition given here is not Comon and Delaune's original definition, but they prove that it is equivalent
- Finite variant property means that can compute a bound on the number of narrowing steps necessary to get a complete solution, this strategy, also due to Comon and Delaune, known as variant narrowing
- Folding variant narrowing of Escobar, Sasse, and Meseguer, eliminates need to compute bounds, also terminates for terms with finite complete sets of variants

The State of Unification in Maude-NPA

- B can be either empty theory or AC
  - Built-in unification for both supplied by Maude
- Limited variant narrowing for subset of finite variant theories including Diffie-Hellman, encryption-decryption cancellation, exclusive-or, Abelian groups, and combinations
- Plan to introduce folding variant narrowing, possibly in Maude
- Also have special-purpose algorithm for encryption homomorphic over a free theory, currently stand-alone
  - Homomorphic operators do not have the finite variant property, so can't use narrowing
    - Variants of $e(K, X * Y)$ are $e(K, X) * e(K, Y), e(K, X) * e(K, Y_1) * e(K, Y_2), \ldots$
  - Possible, however, that the homormorphic axioms $e(K, X * Y) \rightarrow e(K, X) * e(K, Y)$ could go in B
  - Decidability problems if * is Abelian group, but may be able to avoid this with use of sorted unification

Outline

How Maude-NPA Controls the Search Space

- Left to itself, Maude-NPA will search forever
- Uses techniques for ruling out redundant or "obviously" unreachable states which often result in finite search space
- Performed via checks that are usually syntactic, but on terms that obey an equational theory
- Will first describe how we deal with this apparent contradiction via asymmetric unification, then describe the various state reduction techniques used by Maude-NPA
  - Once again, we use the finite variant property

Outline

An Example

- Start with exclusive-or $\oplus$
  - $\oplus$ is AC, with additional equations $x \oplus 0 = x$ and $x \oplus x = 0$.
- Consider the following protocol
  1. $A \rightarrow B : pke(B, N_A)$
  2. $B \rightarrow A : N_B \oplus N_A$
- A checks that the message she receives is $Z \oplus N_A$ for some Z
  - How it works in Maude-NPA
  - Represent A's role by strand `::r::[nil, +(pke(B,n(A,r))),-(Z [+] n(A,r)), nil ]`
  - Consider state `::r::[nil | +(pke(B,n(A,r))), -(Z [+] n(A,r)), nil ], Z [+] n(A,r) inI`
  - Maude-NPA rules this out because Intruder knows expression containing nonce before nonce is generated.
- So, what if after unifying Z with Y, $Z = Y \oplus N_A$? Then $Z \oplus n(A, r) = Y \oplus n(A, r) \oplus n(A, r) = Y$ and the syntax check is no longer valid.

How we handle this in Maude-NPA

- Express equational theory as
  $\Delta = \{X \oplus 0 \to X, X \oplus X \to 0, X \oplus X \oplus Y \to Y\} \uplus (B = \mathrm{AC})$
  - nonce containment invariant under AC
  - Δ is a set of rewrite rules convergent and terminating wrt AC
- Find all the possible reduced forms of `Z [+] n(A,r)` wrt Δ modulo AC
  - There are two:
    - `< Z [+] n(A,r), id >`
    - `< Y, Z |-> Y [+] n(A,r) >`
- One strand for each reduced form
  - `::r::[nil, +(pke(B,n(A,r))),-(Z [+] n(A,r)), nil ]`
  - `::r::[nil, +(pke(B,n(A,r))),-(Y), nil ]`
- Include constraints that negative terms in strands are irreducible wrt Δ
- When unifying with positive terms, only accept unifiers that preserve irreducibility

What we need to make this work

- Characterize theories with decompositions $\Delta \uplus B$ in which every term has a finite number of reduced forms
  - We understand this: this is equivalent to the finite variant property
- Unification algorithms giving a set of unifiers Σ of $x =?y$ most general with respect to the property that for all $\sigma \in \Sigma$, σy is irreducible wrt Δ
  - We call this *asymmetric unification*
  - *Variant narrowing* has this property, we are looking for more efficient algorithms
- What are the properties that we want to remain invariant, and how can we characterize the theories B that preserve them?
  - Presence of subterms such as nonces, depth of terms: cancellation rules should be in Δ
  - Can vary with verification approach and syntactic checks used
  - B = empty theory or AC works well, so does homomorphic property

Asymmetric Unification as a Problem in its Own Right

- As far as we can tell, no-one has studied this before
- Narrowing only algorithm we know of that can achieve this
    - AU at least as hard as symmetric unification (SU)
        - Any SU problem $s =?t$ can be turned into AU problem $s =?X, t =?X$.
    - AU *strictly harder* than SU - XOR without any other symbols is in P for SU but NP-complete for AU
    - Also problems for which SU decidable but AU undecidable (Ertabur, Narendran)
    - SU can be unitary while AU is not (XOR)
- We are working on a general approach for converting equational unification algorithms to asymmetric unification algorithms
- Applying it to XOR with uninterpreted function symbols
- Next steps: combining with other theories, Abelian groups

Outline

Two basic restrictions of the search space

Powerful tools:

1. Learn-only-once: any terms the intruder will learn in the future can't already be known
2. Grammars describing unreachable states: the intruder learns a term in the language described by the grammar only if he/she knew another term in the language in a past state

Motivating Example

- Consider protocol with:
  - Two operators
    - $e(K, X)$ stands for encryption of message X with key K
    - $d(K, X)$ stands for decryption of message X with key K
  - Two regular strands: (Dolev-Yao):
    - $[-(X), +(d(k, X))]$
    - $[+(e(k, r))]$
  - Two Intruder strands
    - $[-(K), -(X), +(d(K, X))]$
    - $[-(K), -(X), +(e(K, X))]$
  - One equation
    - $d(K, e(K, X)) = X$

A Partial (Backwards) Search Tree

$$\begin{tikzcd}[column sep=small]
 & t \arrow[d] & & \\
 & \{e(k,t)\} \arrow[dl] \arrow[dr] & & \\
\{k,t\} \arrow[d] & & \{e(k,e(k,t))\} \arrow[dl] \arrow[dr] & \\
\text{stop} & \{e(k,t),k\} \arrow[d] & & \{e(k,e(k,e(k,t)))\} \arrow[d] \\
 & \text{stop} & & \cdots
\end{tikzcd}$$

Powerful tools:

(1) Learn-only-once: terms the intruder will learn in the future and doesn't know in he past.

(2) Unreachable states: the intruder learns a term only if he/she knew another term in a past state

(1) Learn-Only-Once Restriction

- Suppose in looking for a term t, you find a state where the intruder knows the same t, then cut the search space

$$
\begin{tikzcd}
 & t \arrow[d] \\
 & \{e(k,t)\} \arrow[dl] \\
\{k,t\} \arrow[d] & \\
\text{stop} &
\end{tikzcd}
$$

- Can tell if intruder has not learned X by seeing if intruder will learn X in the future

(2) Languages characterizing unreachable states

$$Z \not\sqsubseteq r$$
$$\downarrow$$
$$\{e(K,Z)\} \longrightarrow \{e(K,e(K,Z))\} \longrightarrow \{e(K,e(K,e(K,Z)))\}$$
$$\downarrow$$
$$\ldots$$

- Discover Grammars providing infinite set of terms intruder can't learn.
  1. $Z \in L \mapsto t \in L$
  2. $Z \in L \mapsto e(Y,Z) \in L$

  1. $Z \notin \mathcal{I},\ e(A,Z) \not\sqsubseteq e(k,r) \mapsto e(A,Z) \in L$
  2. $Z \in L \mapsto e(Y,Z) \in L$
- If the intruder learns a term in the grammar, then he/she must have learned another term in a state in the past.

Grammars - Procedure Is Automated

- Maude-NPA uses function symbol definitions in protocol spec as source for initial grammars
- In cases Maude-NPA fails to generate a grammar, it provides the reasons for its failure
- User can define own initial grammars if desired, either in addition to or in place of Maude-NPA grammars
- Grammar generation heuristics little changed from original NRL Protocol Analyzer
    - Works well on most theories we've tried, with exception of exclusive-or

Outline

Other Ways of Reducing Search Space

- Grammars can reduce infinite to finite, but may still need to cut search space size for efficiency purposes
  - In some cases, grammars alone not enough to reduce infinite to finite, and we need other techniques as well
- We have developed a number of different techniques, and we describe them now
  - Execute Rule 1 First
  - Subsumption Partial Order Reduction
  - Use Power of Strands to See Into Past and Future
  - Super-Lazy Intruder

Execute Rule 1 First

- If there is a strand of the form $[\ l_1, u^- \mid l_2\ ]$ present, execute the rule replacing it by $[\ l_1 \mid u^-, l_2\ ]\ , u \in \mathcal{I}$ first
- If there are several fix an order and execute them all first, in that order
- Removes extra step introduced by converting negative terms to intruder terms
- Implementing this doubled the speed of the tool
    - Not surprising, because replaced two steps by one

Subsumption Partial Order Reduction

- Partial order reduction standard idea in model checking, used in a lot of protocol analysis tools, too
    - Identify when reachability of state S_1 implies reachability of S_2 and remove S_1
    - In Maude-NPA, this happens, roughly, when $S_2 \subseteq S =_B \sigma S_1$ for some substitution σ
    - Can then eliminate S_1

Using the Power of Strands

- Strands allow you to see the past and the future of a local execution
- Helpful since Maude-NPA very sensitive to the past and future
- Things we've done so far
    - If a term $x \notin \mathcal{I}$ and a strand $[\ l_1, -(x), l_2 \mid l_3\ ]$ both appear in a state, then the state is unreachable
        - Reaching it would require violation of intruder-learns-once
    - Let f and g be two terms containing $n(A, r)$. If
        - $f \in \mathcal{I}$ appears in a state, and;
        - $[\ l_1 \mid l_2, +(g), l_3,\ ]$ also appears, with strand identifier containing r and no $n(A, r)$ term in l_1;

    Then reaching the state requires the intruder to learn a nonce before it is generated and thus is unreachable.

Super-Lazy Intruder

- Based on an idea of David Basin, plus a trick used by the old NPA
- If a term $X \in \mathcal{I}$ appears in a state, where X is a variable, we assume that the intruder can easily find x, and so safe to drop it
- Super-lazy intruder: drop terms made out of variable terms, e.g. X;Y and e(K,Y)
- Need to revive variable terms if they later become instantiated
- Solution: keep the term, and state it appears in, around as a "ghost"
    - Revive the ghost, replacing current state by ghost term and ghost state, but with current substitutions to variables if any variable subterm becomes instantiated

Experimental Results 1

Protocol	none					Grammars					%
NSPK	5	19	136	642	4021	4	12	49	185	758	81
NSL	5	19	136	642	4019	4	12	50	190	804	79
SecReT06	1	6	22	119	346	1	2	6	15	36	89
SecReT07	6	20	140	635	4854	6	17	111	493	3823	21
DH	1	14	38	151	816	1	6	14	37	105	87

Protocol	none					Input First					%
NSPK	5	19	136	642	4021	11	123	1669	26432	N/A	0
NSL	5	19	136	642	4019	11	123	1666	26291	N/A	0
SecReT06	1	6	22	119	346	11	133	1977	32098	N/A	0
SecReT07	6	20	140	635	4854	11	127	3402	N/A	N/A	0
DH	1	14	38	151	816	14	135	1991	44157	N/A	0

Protocol	none					Inconsistency					%
NSPK	5	19	136	642	4021	5	18	95	310	650	83
NSL	5	19	136	642	4019	5	18	95	310	650	83
SecReT06	1	6	22	119	346	1	6	22	114	326	5
SecReT07	6	20	140	635	4854	6	18	107	439	3335	31
DH	1	14	38	151	816	1	12	12	56	128	84

Experimental Results 2

Protocol	none					Transition Subsumption					%
NSPK	5	19	136	642	4021	5	15	61	107	237	94
NSL	5	19	136	642	4019	5	15	61	107	237	94
SecReT06	1	6	22	119	346	1	6	15	39	78	77
SecReT07	6	20	140	635	4854	6	15	61	165	506	89
DH	1	14	38	151	816	1	14	26	102	291	64

Protocol	none					Super-lazy Intruder					%
NSPK	5	19	136	642	4021	5	19	136	641	3951	1
NSL	5	19	136	642	4019	5	19	136	641	3949	2
SecReT06	1	6	22	119	346	1	6	22	119	340	2
SecReT07	6	20	140	635	4854	6	16	44	134	424	91
DH	1	14	38	151	816	1	14	38	138	525	35

Protocol	none					All optimizations					%
NSPK	5	19	136	642	4021	4	6	4	2	1	99
NSL	5	19	136	642	4019	4	7	6	2	0	99
SecReT06	1	6	22	119	346	2	3	2	-	-	99
SecReT07	6	20	140	635	4854	5	1	1	1	-	99
DH	1	14	38	151	816	4	6	10	9	12	99

Maude-NPA References

- Maude-NPA 1.0 and relevant papers available at `http://maude.cs.uiuc.edu/tools/Maude-NPA/` . Next version should be out soon.
- S. Escobar, C. Meadows, J. Meseguer. Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties. FOSAD 2007/2008/2009 Tutorial Lectures, LNCS 5705, pages 1-50. Springer-Verlag.
- S. Escobar, C. Meadows, and J. Meseguer. State Space Reduction in the Maude-NRL Protocol Analyzer. In Proc. of 13th European Symposium on Research in Computer Security (ESORICS08), LNCS 5283, pages 548-562, Springer, 2008. (journal version under review)
- S. Escobar, C. Meadows, J. Meseguer, S. Santiago. Sequential Protocol Composition in Maude-NPA. In Proc. of European Symposium on Research in Computer Security (ESORICS 2010), LNCS 6345, pages 303-318. 2010. Technical report DSIC-II/06/10, Departamento de Sistemas Informaticos y Computacion, Universidad Polit cnica de Valencia, 2010.

Narrowing References

- H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In RTA'05, LNCS 3467, pages 294-307. Springer, 2005.
- Santiago Escobar, José Meseguer, Ralf Sasse. Effectively Checking or Disproving the Finite Variant Property In proceedings of 19th International Conference on Rewriting Techniques and Applications (RTA 2008), LNCS 5117, pages 79-93. 2008.
- Santiago Escobar, Ralf Sasse, José Meseguer. Folding variant narrowing and optimal variant termination. The Journal of Logic and Algebraic Programming, 2011, to appear. Earlier version appeared in RTA '09.
- Serdar Erbatur, Santiago Escobar, Deepak Kapur, Zhiqiang Liu, Christopher Lynch, Catherine Meadows, Jose Meseguer, Paliath Narendran and Ralf Sasse Asymmetric Unification: A New Unification Paradigm for Cryptographic Protocol Analysis, UNIF 2011.