Term Rewriting applied to Cryptographic Protocol Analysis: the Maude-NPA tool

Santiago Escobar

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
sescobar@dsic.upv.es

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities
2. Introduction to Rewriting Logic
3. How Maude-NPA works
4. Examples of execution

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities
2. Introduction to Rewriting Logic
3. How Maude-NPA works
4. Examples of execution

Formal Analysis of Protocols

- Crypto protocol analysis in the standard model is well understood.
- Need to support algebraic properties of some protocols
  - Diffie-Hellman exponentiation,
  - exclusive-or,
  - homomorphism (one-sided distributivity)
- These operations well understood in the bounded sessions case
  - Decidability results for exclusive-or, exponentiation, homomorphisms, etc.
- What is lacking:
  (1) more general understanding, especially for unbounded sessions,
  (2) tool support.

Our approach

- Use rewriting logic as general theoretical framework
  - protocols and intruder rules specified as transition rewrite rules
  - crypto properties as oriented equational properties and axioms
- Use narrowing modulo equational theories in two ways
  - as a symbolic reachability analysis method
  - as an extensible equational unification method
- Combine with state reduction techniques (grammars, optimizations, etc.)
- Implement in Maude programming environment
  - Rewriting logic gives us theoretical framework and understanding
  - Maude implementation gives us tool support

Our Plans

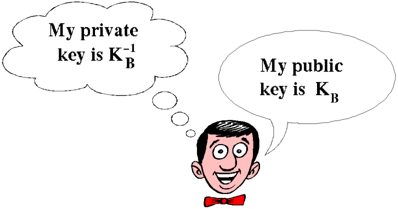
1. Start by formalizing NPA techniques in rewriting logic (2005)
2. Extend model to different types of equational theories (2006)
   - Explicit Encryption and Decryption, AC-unification, Diffie-Hellman Exponentiation, Exclusive-or
3. Include state reduction techniques (2008, 2013)
4. Document and distribute the tool (v1.0 2007)
5. Sequential protocol composition: specification and analysis (2010)
6. Integrate dedicated unification algorithms (2011)
   - Homomorphism, Exclusive-or
7. Document and distribute the tool (v2.0 2012)
8. Extensive protocol analysis (2012-now)
   - Homomorphism, Exclusive-or, Abelian groups
9. Advanced properties:
   - Indistinguishability (2013-now), Conditional protocols (2016)
10. Standard APIs: IBM CCA, PKCS#11, Yubikey (2014-now)
11. Document and distribute the tool (v3.0 2016)

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities

Building Blocks for Security Protocols

Cryptographic Procedures: encryption of messages.



$$\{\{M\}_{K_B}\}_{K_B^{-1}} = M$$

(Pseudo-)Random Number Generators: to generate "nonces", e.g. for "challenge/response".

Protocols: recipe for exchanging messages.

Steps like: *A sends B her name together with the message M. The pair $\{A, M\}$ is encrypted with B's public key*.

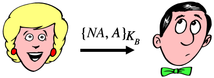
$$A \to B : \{A, M\}_{K_B}$$

An authentication protocol

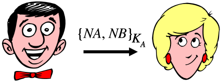
The Needham-Schroeder Public Key protocol (NSPK):

$$\begin{array}{lll} 1. & A \to B: & \{NA, A\}_{K_B} \\ 2. & B \to A: & \{NA, NB\}_{K_A} \\ 3. & A \to B: & \{NB\}_{K_B} \end{array}$$

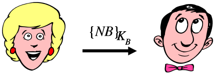
Goal: mutual authentication. Translation:



"This is Alice and I have chosen a nonce *NA*."



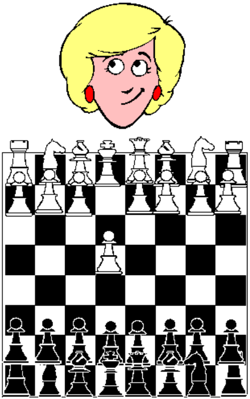
"Here is your nonce *NA*. Since I could read it,
I must be Bob. I also have a challenge *NB* for you."

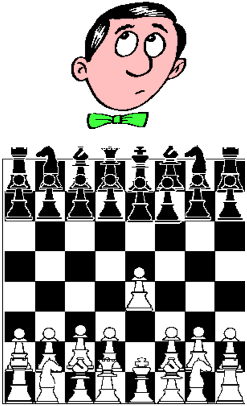


"You sent me *NB*. Since only Alice can read this
and I sent it back, you must be Alice."

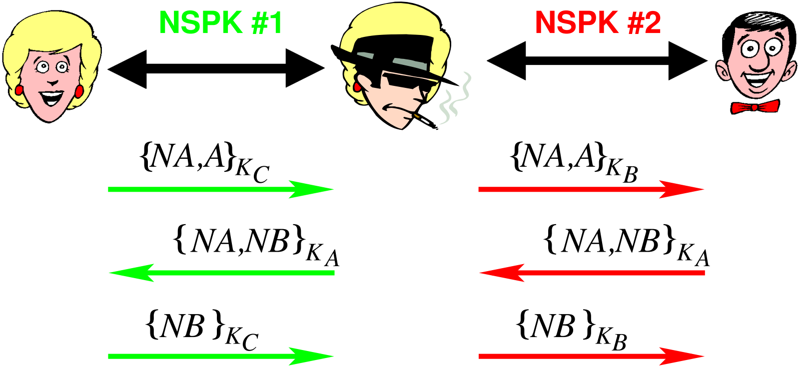
NSPK proposed in 1970s and used for decades, until...
Protocols are typically small and convincing... and often wrong!

How to at least tie against a Chess Grandmaster





Man-in-the-middle attack on NSPK



B believes he is speaking with *A*!

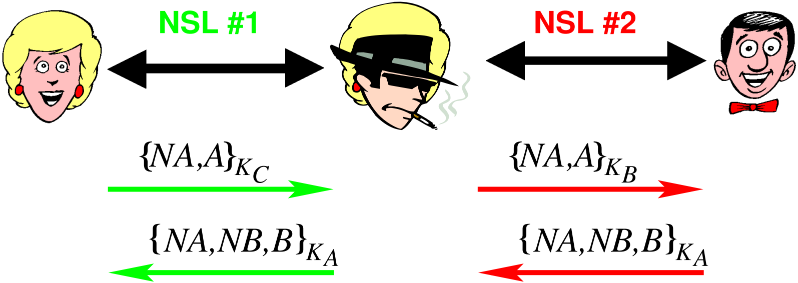
What went wrong?

- Problem in step 2:

$$B \mapsto A : \{N_A, N_B\}_{K_A}$$

- Agent B should also give his name: NA, NB, BKA .
- The improved version is called NSL protocol by Gavin Lowe.
- Is the protocol now correct?

Needham-Schroeder-Lowe Public Key Exchange Protocol



A aborts the protocol execution!
(or ignores the message)

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities

Example: Needham-Schroeder Public Key Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B)$
$A \longrightarrow B : pk(B, N_B)$

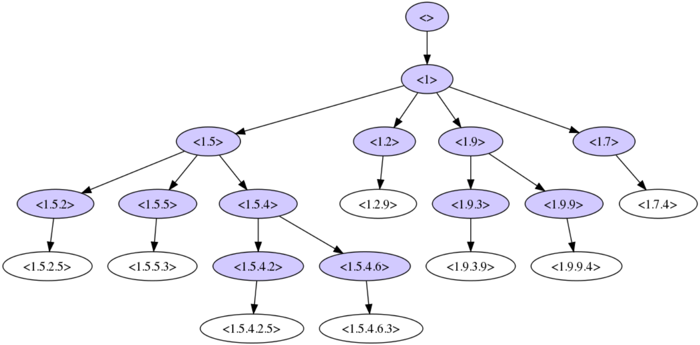
Attack sequence

1. $(pk(i, a; n(a, r1)))^+$
2. $(pk(i, n(b, r2)))^-$
3. $(a; n(a, r1))^+$
4. $(a; n(a, r1))^-$
5. $(pk(b, a; n(a, r1)))^+$
6. $(pk(b, a; n(a, r1)))^-$
7. $(pk(a, n(a, r1); n(b, r2)))^+$
8. $(pk(a, n(a, r1); n(b, r2)))^-$
9. $(pk(i, n(b, r2)))^+$
10. $(pk(i, n(b, r2)))^-$
11. $(n(b, r2))^+$
12. $(n(b, r2))^-$
13. $(pk(b, n(b, r2)))^+$
14. $(pk(b, n(b, r2)))^-$

Example: Needham-Schroeder-Lowe Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B; B)$
$A \longrightarrow B : pk(B, N_B)$



Example: NSL-xor Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$

$B \longrightarrow A : pk(A, N_A; N_B \oplus B)$

$A \longrightarrow B : pk(B, N_B)$

Attack sequence

1. $(pk(i, a; n(a, r1)))^+$
2. $(pk(i, n(b, r2)))^-$
3. $(a; n(a, r1))^+$
4. $(a; n(a, r1))^-$
5. $(pk(b, a; n(a, r1)))^+$
6. $generatedByIntruder(b \oplus i)$
7. $(pk(b, a; n(a, r1)))^-$
8. $(pk(a, n(a, r1); n(b, r2); b))^+$
9. $(pk(a, n(a, r1); n(b, r2); b))^-$
10. $(pk(i, n(b, r2) \oplus b \oplus i))^+$
11. $(pk(i, n(b, r2) \oplus b \oplus i))^-$
12. $(n(b, r2) \oplus b \oplus i)^+$
13. $(b \oplus i)^-$
14. $(n(b, r2) \oplus b \oplus i)^+$
15. $(n(b, r2)))^+$
16. $(n(b, r2)))^-$
17. $(pk(b, n(b, r2)))^+$
18. $(pk(b, n(b, r2)))^-$

Example: NSL-homomorphism Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B; B)$
$A \longrightarrow B : pk(B, N_B)$

Attack sequence

1. $generatedByIntruder(pk(a,i))$
2. $generatedByIntruder(pk(b,a;NI))$
3. $(pk(b,a;NI))^-$
4. $(pk(a,NI;n(b,r2);b))^+$
5. $(pk(a,NI);pk(a,n(b,r2));pk(a,b))^-$
6. $(pk(a,n(b,r2));pk(a,b))^+$
7. $(pk(a,n(b,r2));pk(a,b))^-$
8. $(pk(a,n(b,r2)))^+$
9. $(pk(a,i)^-$
10. $(pk(a,n(b,r2)))^-$
11. $(pk(i,a);pk(a,n(b,r2)))^+$
12. $pk(a,i;n(b,r2))^-$
13. $(pk(i,n(b,r1);n(a,r1);a))^+$
14. $(pk(i,n(b,r2));pk(i,n(a,r1));pk(i,a))^-$
15. $(pk(i,n(b,r2)))^+$
16. $(pk(i,n(b,r2)))^-$
17. $(n(b,r2))^+$
18. $(n(b,r2))^-$
19. $(pk(b,n(b,r2)))^+$
20. $(pk(b,n(b,r2)))^-$

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities

Explicit Encryption and Decryption

- Most formal models lack explicit decryption operator and assume implicit decryption
- If a principal knows an encrypted message and the key, assume principal can decrypt message under the following conditions
  - Implicit assumption that principal never decrypts a message that wasn't encrypted with a key known by the principal
  - Assumption that principals can check format of decrypted message
- What if these assumptions do not hold?
- In that case, need to model both encryption and decryption symbols explicitly, plus their cancellation, e.g. $d(K, e(K, Y)) = Y$.

Example: Needham-Schroeder Public Key (NSPK)

Modular Exponentiation in Diffie-Hellman

- Basic DH example protocol (each nonzero residue mod P is a power of g)
  1. $A \rightarrow B : g^{N_A}\ mod\ P$
     B computes $(g^{N_A})^{N_B}\ mod\ P$
  2. $B \rightarrow A : g^{N_B}\ mod\ P$
     A and B compute $(g^{N_B})^{N_A} = (g^{N_A})^{N_B}\ mod\ P$ and get a shared secret key.
- Properties:

$$(g^X)^Y = g^{X*Y} = g^{Y*X} = (g^Y)^X$$
$$(X*Y)*Z = X*(Y*Z) \qquad X*Y = Y*X$$

of modular exponentiation in order to faithfully represent this protocol

Example: Diffie-Hellman Protocol

Exclusive-Or

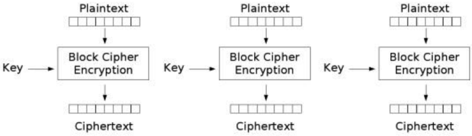
- Cheap and has provable security properties
  - If we send $X \oplus R$, where R a random secret, observer learns no more about X than before it saw message
- On the other hand, associativity-commutativity and cancellation properties make it tricky to reason about

$$\begin{array}{ll} X \oplus Y = Y \oplus X & X \oplus X = 0 \\ (X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) & X \oplus 0 = X \end{array}$$

Example: Needham-Schroeder-Lowe with XOR (NSL-xor)

Homomorphism

- The electronic codebook (ECB) encryption splits a message into blocks and cyphers the blocks using the same key



Electronic Codebook (ECB) mode encryption

- Identical plaintext blocks are encrypted into identical ciphertext blocks (does not hide data patterns well). Sensitive to the property:

$$e(K, X; Y) = e(K, X); e(K, Y)$$

Example: NSL with homomorphic encryption

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities
2. Introduction to Rewriting Logic
3. How Maude-NPA works
4. Examples of execution

Rewriting Logic in a Nutshell

Definition

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. $e(K, N_A; X) \to e(K, X)$
- (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.

$$e(k, n_a; m) \in T_{\Sigma/E}$$

$$d(k_2, e(k_2, e(k, n_a; m))) \notin T_{\Sigma/E}$$

Rewriting modulo

Definition

Given (Σ, E, R), $t \rightarrow_{R,E} s$ if there is

- a position $p \in Pos(t)$;
- a rule $l \rightarrow r$ in R;
- a matching σ (modulo E)
  such that $t|_p =_E \sigma(l)$, and $s = t[\sigma(r)]_p$.

Example:

- $R = \{\ e(K, N_A; X) \rightarrow e(K, X)\ \}$
- $E = \{\ \ d(K, e(K, Y)) = Y\ \}$
- $e(k,\ n_A; m) \rightarrow_{R,E} e(k, m)$
  $d(k,\ e(k, e(k_2, n_A; m))) =_E e(k_2, n_a; m) \rightarrow_{R,E} e(k_2, m)$

Narrowing and Backwards Narrowing

Definition

Given (Σ, E, R), $t \rightsquigarrow_{\sigma,R,E} s$ if there is

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

Example:

- $R = \{\ e(K, N_A; X) \to e(K, X)\ \}$
- $E = \{\ \ d(K, e(K, Y)) = Y\ \}$
- $$\begin{array}{lll} e(k,\ X) & \rightsquigarrow_{\{X \mapsto N_A; X'\},R,E} & e(k, X') \\ d(k,\ X) & \rightsquigarrow_{\{X \mapsto e(k, e(K, N_A; X'))\},R,E} & e(K, X') \end{array}$$

Backwards Narrowing: Narrowing with rewrite rules reversed

Narrowing Reachability Analysis

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

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

in a given rewrite theory.

- The terms t_i and t'_i denote sets of states (all the possible instances of the term)
- Symbolyc reachability means for what subset of states denoted by t_i are the states denoted by t'_i reachable?
- No finiteness assumptions about the state space.

Equational Unification

Definition

Given an order-sorted equational theory $(\Sigma, Ax \uplus E)$ and $t \stackrel{?}{=} t'$, an $(Ax \uplus E)$-unifier is an order-sorted subst. σ s.t. $\sigma(t) =_{Ax \uplus E} \sigma(t')$.

Compared to syntactic unification:

- $f(a, X) = f(Y, b)$ has solution $X \mapsto b, Y \mapsto a$
- $f(a, X) =_{AC} f(b, Y)$ has solution $X \mapsto b, Y \mapsto a$
- $X + 0 =_{ACU} X$, where 0 is the identity, has solution id
- $X + a + b =_{XOR} a$ has solution $X \mapsto b, Y \mapsto a$

Equational Unification - Complete

When $Ax = \emptyset$ and E convergent TRS

Narrowing provides a complete (but semi-decidable) E-unification procedure [Hullot80]. e.g. cancellation $d(K,e(K,M)) \to M$.

When $Ax \neq \emptyset$ and E convergent and coherent TRS modulo Ax

Narrowing provides a complete (but semi-decidable) E-unification procedure [Jouannaud-Kirchner-Kirchner-83] e.g. exclusive-or
$X * 0 \to X, X * X \to 0 \mid (X * Y) * Z = X * (Y * Z), X * Y = Y * X$

Equational Unification - Decidable

When $Ax = \emptyset$

Basic narrowing strategy [Hullot80] is complete for normalized substitutions.
Cases where basic narrowing terminates have been studied [Alpuente-Escobar-Iborra-TCS09].

When $Ax \neq \emptyset$

Folding variant-narrowing [Escobar-Meseguer-Sasse-JLAP12] is the most promising strategy for equational unification. Fully implemented in Maude.

E,Ax-variants

E,Ax-variant

Given a term t and an equational theory $Ax \uplus E$, (t',θ) is an E,Ax-variant of t if $\theta(t)\downarrow_{E,Ax} =_{Ax} t'$ [Comon-Delaune-RTA05]

Finite and complete set of E,Ax-variants

$\forall\sigma$ s.t. $\sigma(t)\downarrow_{E,Ax} = t'$, $\exists(t'',\theta) \in V_{E,Ax}(t)$ s.t.

1. t'' is in $\rightarrow_{E,Ax}$-normal form
2. t' and t'' ($\sigma\downarrow_{E,Ax}$ and θ) are just renamings modulo Ax.

Finite Variant Property

Theory has FVP if there is a finite number of most general E,Ax-variants for every term.

E,Ax-variants - Example

$$X \oplus 0 \to X$$
$$X \oplus X \to 0$$
$$X \oplus X \oplus Y \to Y$$
(cancellation rules: E)

$$X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z$$
$$X \oplus Y = Y \oplus X$$
(axioms: Ax)

- For $X \oplus X$ only E,Ax-variant is: $(0, id)$
- For $X \oplus Y$ there are 7 most general E,Ax-variants
  1. $(X \oplus Y, id)$
  2. $(0, \{X \mapsto U, Y \mapsto U\})$
  3. $(Z, \{X \mapsto 0, Y \mapsto Z\})$
  4. $(Z, \{X \mapsto Z \oplus U, Y \mapsto U\})$
  5. $(Z, \{X \mapsto Z, Y \mapsto 0\})$
  6. $(Z, \{X \mapsto U, Y \mapsto Z \oplus U\})$
  7. $(Z_1 \oplus Z_2, \{X \mapsto U \oplus Z_1, Y \mapsto U \oplus Z_2\})$

Narrowing & Unification in Maude-NPA

- Cryptographic protocols are modeled as a rewrite theory $\mathcal{P} = (\Sigma, \Delta \uplus B, R)$
- Narrowing at two levels in Maude-NPA
  1. a theory $(\Sigma, \Delta \uplus B, R)$: ($\Delta \uplus B$-narrowing with rules R)
  2. for $\Delta \uplus B$-unification (B-narrowing with rules Δ)
- $\Delta \uplus B$-unification for each backwards step using R
  1. Built-in Maude ACU unification algorithms
  2. Dedicated unification algorithms (xor, homomorphism)
  3. Hybrid approach: built-in algorithms for B, and a generic algorithm (variant narrowing) for Δ.

Outline

1 Formal Analysis of Protocols
The Needham-Schroeder Public Key
Motivating Protocols
Some Examples of Algebraic Identities

2 Introduction to Rewriting Logic

3 How Maude-NPA works

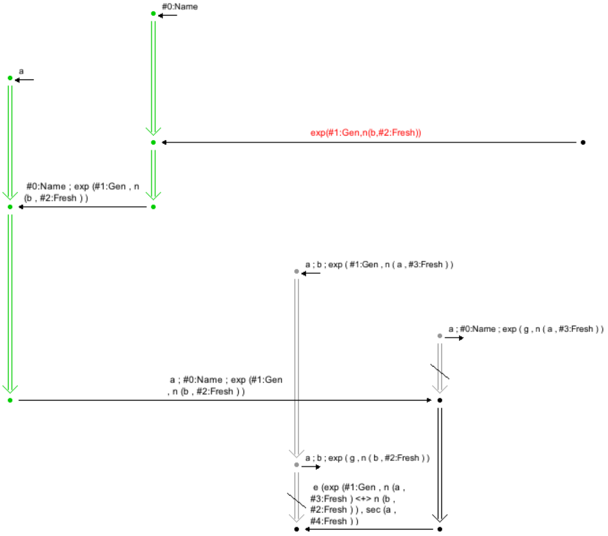
4 Examples of execution

Maude-NPA

- A tool to find or prove the absence of attacks
- Analyzes infinite state systems:
    - Active Dolev-Yao intruder
    - No abstraction or approximation of nonces
    - Unbounded number of sessions
- Performs symbolic backwards search from an insecure state to find attacks or to prove unreachability of cryptographic protocols
- Sensitive to past and future

Basic Structure of Maude-NPA

- Honest principal and intruder actions are modeled as a strand space (Thayer, Herzog, and Guttman)



Basic Structure of Maude-NPA

- A strand is a sequence of positive and negative terms
  - Negative term stand for received message
  - Positive terms stand for sent messages
  - **Example**:
    (honest) $[\ pke(B, N_A; A)^+,\ pke(A, N_A; N_B)^-,\ pke(B, N_B)^+\ ]$
    (intruder $[X^-, pke(A, X)^+]$ and $[X^-, Y^-, (X; Y)^+]$
- Modified strand notation: a marker denoting the current state
  - **Example**: $[\ pke(B, N_A; A)^+ \mid pke(A, N_A; N_B)^-,\ pke(B, N_B)^+\ ]$
- Strand annotated with fresh terms generated by principal executing strands (to obtain an infinite number of nonces)
  $:: r :: [\ pke(B, n(A, r); A)^+ \mid pke(A, n(A, r); N_B)^-,\ pke(B, N_B)^+\ ]$
- Intruder knowledge explicitly represented
  - $m \in \mathcal{I}$: terms already learnt by the intruder
  - $m \notin \mathcal{I}$: terms the intruder does not know, but that will be learnt

Basic Structure of Maude-NPA

- A state is a set of strands plus the intruder knowledge

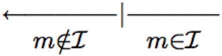
$$\ldots [\ nil, m_1^{\pm}, \ldots, m_i^{\pm} \mid m_{i+1}^{\pm}, \ldots, m_k^{\pm}, nil\ ]\ \&$$
$$\{t_1 \notin \mathcal{I}, \ldots, t_j \notin \mathcal{I}\}, \{s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I}\}$$

- Initial strand $[\ nil \mid m_1^{\pm}, \ldots, m_n^{\pm}, nil\ ]$
- Final strand $[\ nil, m_1^{\pm}, \ldots, m_n^{\pm}, \mid nil\ ]$
- Initial Intruder knowledge $\{t_1 \notin \mathcal{I}, \ldots, t_n \notin \mathcal{I}\}$
- Final Intruder knowledge $\{t_1 \in \mathcal{I}, \ldots, t_n \in \mathcal{I}\}$

Protocol Rules and Their Execution

$$\ldots [\ nil, m_1^{\pm}, \ldots, m_i^{\pm} \mid m_{i+1}^{\pm}, \ldots, m_k^{\pm}, nil\ ] \ \&$$
$$\{t_1 \notin \mathcal{I}, \ldots, t_j \notin \mathcal{I}\}, \{s_1 \in \mathcal{I}, \ldots, s_m \in \mathcal{I}\}$$

- Negative message m_i^- in the past part of the strand is
  - E-unified with a term already known by the intruder $s_p \in \mathcal{I}$
  - or introduced into the intruder knowledge as $m_i \in \mathcal{I}$
- Positive message m_i^+ in the past part of the strand is
  - E-unified with term known by the intruder $s_p \in \mathcal{I}$, and then $s_p \in \mathcal{I}$ is transformed into $s_p \notin \mathcal{I}$



Protocol Rules and Their Execution

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

1. $[\ L \mid M^{-}, L'\ ]\ \&\ \{M \in \mathcal{I}, K\} \to [\ L, M^{-} \mid L'\ ]\ \&\ \{M \in \mathcal{I}, K\}$
   Moves input messages into the past
2. $[\ L \mid M^{+}, L'\ ]\ \&\ \{K\} \to [\ L, M^{+} \mid L'\ ]\ \&\ \{K\}$
   Moves output message that are not read into the past
3. $[\ L \mid M^{+}, L'\ ]\ \&\ \{M \notin \mathcal{I}, K\} \to [\ L, M^{+} \mid L'\ ]\ \&\ \{M \in \mathcal{I}, K\}$
   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 3 using each strand of the protocol $\mathcal{P}$:

$$\{\ [\ l_1 \mid u^+, l_2\ ]\ \&\ \{u \notin \mathcal{I}, K\} \rightarrow \{u \in \mathcal{I}, K\}$$
$$\text{s.t. } [\ l_1,\ u^+,\ l_2\ ] \in \mathcal{P}\}$$

Backwards Reachability Analysis

- Backwards narrowing protocol execution defines a backwards reachability relation
- Specify a state describing the attack state, including a set of final strands plus terms $u \in \mathcal{I}$ and $u \notin \mathcal{I}$
- Execute the protocol backwards to an initial state, if possible
- In initial step, prove lemmas that identify certain states unreachable (if necessary)
- For each intermediate state found, several optimizations available (check if it can be proved unreachable and discard)
- Also global optimizations (super lazy intruder, state subsumption)

Outline

1. Formal Analysis of Protocols
   - The Needham-Schroeder Public Key
   - Motivating Protocols
   - Some Examples of Algebraic Identities
2. Introduction to Rewriting Logic
3. How Maude-NPA works
4. Examples of execution

Example: Needham-Schroeder Public Key Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B)$
$A \longrightarrow B : pk(B, N_B)$

Protocol (strand spaces)

$$
\begin{aligned}
&:: r1 :: [nil \mid (pk(B, A; n(A, r1)))^+,\\
&\qquad (pk(A, n(A, r1); N_B))^-,\\
&\qquad pk(B, N_B)^+]\\
&:: r2 :: [nil \mid (pk(B, A; N_A))^-,\\
&\qquad (pk(A, N_A; n(B, r2)))^+,\\
&\qquad (pk(B, n(B, r2)))^-]
\end{aligned}
$$

Intruder capabilities

$[nil \mid (M_1; M_2)^-, M_1^+]$
$[nil \mid (M_1; M_2)^-, M_2^+]$
$[nil \mid M_1^-, M_2^-, (M_1; M_2)^+]$
$[nil \mid M^-, (sk(i, M))^+]$
$[nil \mid M^-, (pk(Ke, M))^+]$

Equational Theory - Algebraic properties

$B = \{\ (X\ ;\ Y)\ ;\ Z = X\ ;\ (Y\ ;\ Z)\}$

$$
\begin{aligned}
\Delta = \{\ & pk(Ke, sk(Ke, X)) = X,\\
& sk(Ke, pk(Ke, X)) = X\}
\end{aligned}
$$

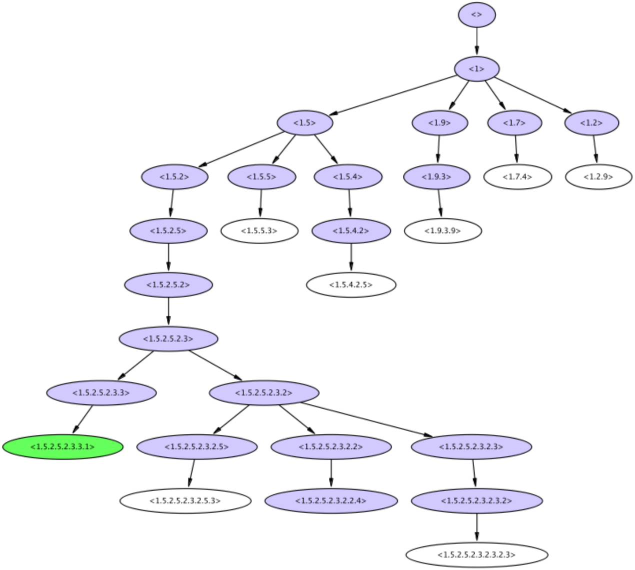
Needham-Schroeder Public Key: Attack State Pattern

$$:: r2 ::$$

$$[nil, (pk(B, A; N_A))^-, (pk(A, N_A; n(B, r2)))^+, (pk(B, n(B, r2)))^- \mid nil\ ]$$

$$\&\ SS\ \&\ \{n(B, r2) \in \mathcal{I},\ IK\}$$

Needham-Schroeder Public Key: Search State Space



Needham-Schroeder Public Key: Initial State

$$
\begin{array}{l}
[nil \mid (pk(i,n(b,r2)))^-, (n(b,r2))^+, nil] \ \& \\
[nil \mid (pk(i,a;n(a,r1)))^-, (a;n(a,r1))^+, nil] \ \& \\
[nil \mid (n(b,r2))^-, (pk(b,n(b,r2)))^+, nil] \ \& \\
[nil \mid (a;n(a,r1))^-, (pk(b,a;n(a,r1)))^+, nil] \ \& \\
:: r1 :: \\
[nil \mid (pk(i,a;n(a,r1)))^+, (pk(a,n(a,r1);n(b,r2)))^-, (pk(i,n(b,r2)))^+, nil] \ \& \\
:: r2 :: \\
[nil \mid (pk(b,a;n(a,r1)))^-, (pk(a,n(a,r1);n(b,r2)))^+, (pk(b,n(b,r2)))^-, nil]
\end{array}
$$

Needham-Schroeder Public Key: Attack sequence

1. $(pk(i,a;n(a,r1)))^+$
2. $(pk(i,n(b,r2)))^-$
3. $(a;n(a,r1))^+$
4. $(a;n(a,r1))^-$
5. $(pk(b,a;n(a,r1)))^+$
6. $(pk(b,a;n(a,r1)))^-$
7. $(pk(a,n(a,r1);n(b,r2)))^+$
8. $(pk(a,n(a,r1);n(b,r2)))^-$
9. $(pk(i,n(b,r2)))^+$
10. $(pk(i,n(b,r2)))^-$
11. $(n(b,r2))^+$
12. $(n(b,r2))^-$
13. $(pk(b,n(b,r2)))^+$
14. $(pk(b,n(b,r2)))^-$

Example: Needham-Schroeder-Lowe Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B; B)$
$A \longrightarrow B : pk(B, N_B)$

Protocol (strand spaces)

$$
\begin{array}{l}
:: r1 :: [nil \mid (pk(B, A; n(A, r1)))^+, \\
\qquad\qquad (pk(A, n(A, r1); N_B; B))^-, \\
\qquad\qquad pk(B, N_B)^+] \\
:: r2 :: [nil \mid (pk(B, A; N_A))^-, \\
\qquad\qquad (pk(A, N_A; n(B, r2); B))^+, \\
\qquad\qquad (pk(B, n(B, r2)))^-]
\end{array}
$$

Intruder capabilities

$[nil \mid (M_1; M_2)^-, M_1^+]$
$[nil \mid (M_1; M_2)^-, M_2^+]$
$[nil \mid M_1^-, M_2^-, (M_1; M_2)^+]$
$[nil \mid M^-, (sk(i, M))^+]$
$[nil \mid M^-, (pk(Ke, M))^+]$

Equational Theory - Algebraic properties

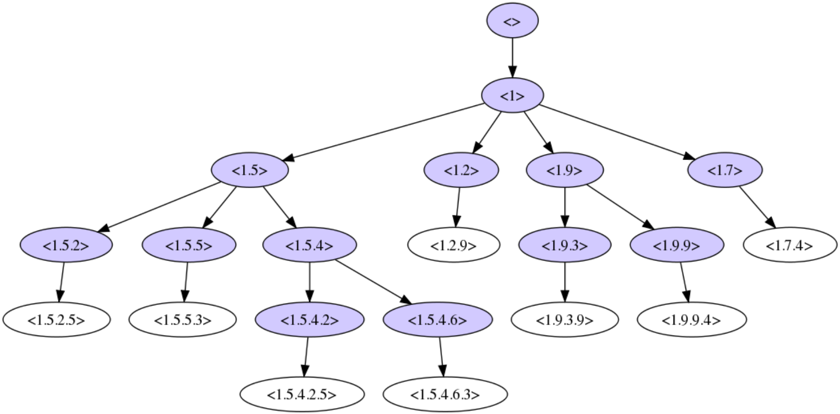
$B = \{\ (X\ ;\ Y)\ ;\ Z = X\ ;\ (Y\ ;\ Z)\}$
$\Delta = \{\ pk(Ke, sk(Ke, X)) = X,$
$\qquad sk(Ke, pk(Ke, X)) = X\}$

Needham-Schroeder-Lowe: Attack State Pattern

$$:: r2 ::$$

$$[nil, (pk(B, A; N_A))^-, (pk(A, N_A; n(B, r2); B))^+, (pk(B, n(B, r2)))^- \mid nil\ ]$$
$$\&\ SS\ \&\ \{n(B, r2) \in \mathcal{I},\ IK\}$$

Needham-Schroeder-Lowe: Search State Space



Example: NSL-xor Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B \oplus B)$
$A \longrightarrow B : pk(B, N_B)$

Protocol (strand spaces)

$$
\begin{array}{l}
:: r1 :: [nil \mid (pk(B, A; n(A, r1)))^+, \\
\qquad (pk(A, n(A, r1); N_B \oplus B))^-, \\
\qquad pk(B, N_B)^+] \\
:: r2 :: [nil \mid (pk(B, A; N_A))^-, \\
\qquad (pk(A, N_A; n(B, r2) \oplus B))^+, \\
\qquad (pk(B, n(B, r2)))^-]
\end{array}
$$

Intruder capabilities

$$
\begin{array}{l}
[nil \mid (M_1; M_2)^-, M_1^+] \\
[nil \mid (M_1; M_2)^-, M_2^+] \\
[nil \mid M_1^-, M_2^-, (M_1; M_2)^+] \\
[nil \mid NS_1^-, NS_2^-, (NS_1 \oplus NS_2)^+] \\
[nil \mid null^+] \\
[nil \mid M^-, (sk(i, M))^+] \\
[nil \mid M^-, (pk(Ke, M))^+]
\end{array}
$$

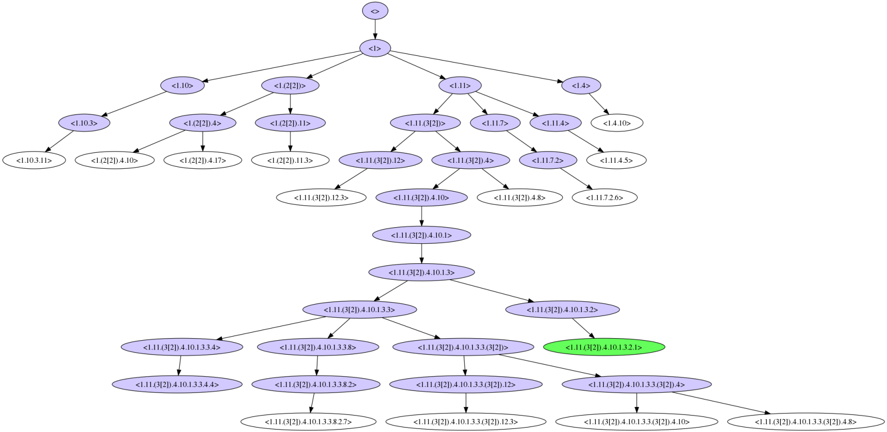
Equational Theory - Algebraic properties

$$
\begin{array}{l}
B = \{\ (X \ \oplus \ Y) \ \oplus \ Z = X \ \oplus \ (Y \ \oplus \ Z), \\
\qquad X \ \oplus \ Y = Y \ \oplus \ X\} \\
\Delta = \{\ pk(Ke, sk(Ke, X)) = X,\ sk(Ke, pk(Ke, X)) = X,\ NS \oplus NS = null, \\
\qquad NS1 \oplus NS1 \oplus NS2 = NS2,\ NS \oplus null = NS\}
\end{array}
$$

NSL-xor: Attack State Pattern

$$
\begin{array}{l}
:: r2 :: \\
[nil, (pk(B, A; NS_A))^-, (pk(A, NS_A; n(B, r2) \oplus B))^+, \\
\quad (pk(B, n(B, r2)))^- \mid nil\ ] \\
\ \& \ SS \ \& \ \{n(B, r2) \in \mathcal{I},\ IK\}
\end{array}
$$

NSL-xor: Search State Space



NSL-xor: Initial State

$[nil \mid (pk(i,a;n(a,r1)))^-, (a;n(a,r1))^+, nil]$ &
$[nil \mid (pk(i,b \oplus i \oplus n(b,r1)))^-, (b \oplus i \oplus n(b,r1))^+, nil]$ &
$[nil \mid (a;n(a,r1))^-, (pk(b,a;n(a,r1))^+, nil]$ &
$[nil \mid (n(b,r2))^-, (pk(b,n(b,r2)))^+, nil]$ &
$[nil \mid (b \oplus i)^-, (b \oplus i \oplus n(b,r2))^-, (n(b,r2))^+, nil]$ &
$:: r1 ::$
$[nil \mid (pk(i,a;n(a,r1)))^+, (pk(a,n(a,r1);n(b,r2) \oplus b))^-, (pk(i,b \oplus i \oplus n(b,r1)))^+, nil]$ &
$:: r2 ::$
$[nil \mid (pk(b,a;n(a,r1)))^-, (pk(a,n(a,r1);n(b,r2) \oplus b))^+, (pk(b,n(b,r2)))^-, nil]$

NSL-xor: Attack sequence

1. $(pk(i,a;n(a,r1)))^{+}$
2. $(pk(i,n(b,r2)))^{-}$
3. $(a;n(a,r1))^{+}$
4. $(a;n(a,r1))^{-}$
5. $(pk(b,a;n(a,r1)))^{+}$
6. $generatedByIntruder(b \oplus i)$
7. $(pk(b,a;n(a,r1)))^{-}$
8. $(pk(a,n(a,r1);n(b,r2);b))^{+}$
9. $(pk(a,n(a,r1);n(b,r2);b))^{-}$
10. $(pk(i,n(b,r2) \oplus b \oplus i))^{+}$
11. $(pk(i,n(b,r2) \oplus b \oplus i))^{-}$
12. $(n(b,r2) \oplus b \oplus i)^{+}$
13. $(b \oplus i)^{-}$
14. $(n(b,r2) \oplus b \oplus i)^{+}$
15. $(n(b,r2)))+$
16. $(n(b,r2)))-$
17. $(pk(b,n(b,r2)))+$
18. $(pk(b,n(b,r2)))-$

Example: NSL-homomorphism Protocol

Protocol (text-book)

$A \longrightarrow B : pk(B, A; N_A)$
$B \longrightarrow A : pk(A, N_A; N_B; B)$
$A \longrightarrow B : pk(B, N_B)$

Protocol (strand spaces)

$$:: r1 :: [nil \mid (pk(B, A; n(A, r1)))^+, \; (pk(A, n(A, r1); N_B; B))^-, \; pk(B, N_B)^+]$$
$$:: r2 :: [nil \mid (pk(B, A; N_A))^-, \; (pk(A, N_A; n(B, r2); B))^+, \; (pk(B, n(B, r2)))^-]$$

Intruder capabilities

$[nil \mid (M_1; M_2)^-, M_1^+]$
$[nil \mid (M_1; M_2)^-, M_2^+]$
$[nil \mid M_1^-, M_2^-, (M_1; M_2)^+]$
$[nil \mid M^-, (pk(Ke, M))^+]$
$[nil \mid (pk(i, M))^+, M^+]$

Equational Theory - Algebraic properties

$B = \{ \ (X \ ; \ Y) \ ; \ Z = X \ ; \ (Y \ ; \ Z)\}$
$\Delta = \{ \ pk(Ke, X; Y) = pk(Ke, X); pk(Ke, Y)\}$

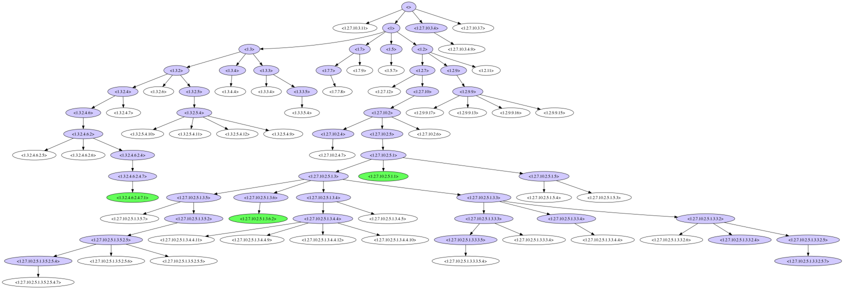
NSL-homomorphism: Attack State Pattern

$:: r2 ::$

$[nil, (pk(B, A; N_A))^-, (pk(A, N_A; n(B, r2); B))^+, (pk(B, n(B, r2)))^- \mid nil\ ]$

$\& \ SS \ \& \ \{n(B, r2) \in \mathcal{I}, \ IK\}$

NSL-homomorphism: Search State Space



NSL-homomorphic: Initial State

$$
\begin{array}{l}
[nil \mid (pk(a,i)^-, (pk(a,n(b,r2)))^-, (pk(i,a);pk(a,n(b,r2)))^+, nil] \\
[nil \mid (pk(i,n(b,r2)))^-, (n(b,r2))^+, nil] \\
[nil \mid (n(b,r2))^-, (pk(b,n(b,r2)))^+, nil] \\
[nil \mid (pk(a,NI);pk(a,n(b,r2));pk(a,b))^-, (pk(a,n(b,r2));pk(a,b))^+, nil] \\
[nil \mid (pk(a,n(b,r2));pk(a,b))^-, (pk(a,n(b,r2)))^+, nil] \\
[nil \mid (pk(i,n(b,r2));pk(i,n(a,r1));pk(i,a))^-, (pk(i,n(b,r2)))^+, nil] \\
::\ r1\ :: \\
[nil \mid (pk(a,i;n(b,r2))^-, (pk(i,n(b,r2);n(a,r1);a))^+, nil] \\
::\ r2\ :: \\
[nil \mid (pk(b,a;NI))^-, (pk(a,NI;n(b,r2);b))^+, (pk(b,n(b,r2)))^-, nil]
\end{array}
$$

NSL-homomorphism: Attack sequence

1. $generatedByIntruder(pk(a,i))$
2. $generatedByIntruder(pk(b,a;NI))$
3. $(pk(b,a;NI))^-$
4. $(pk(a,NI;n(b,r2);b))^+$
5. $(pk(a,NI);pk(a,n(b,r2));pk(a,b))^-$
6. $(pk(a,n(b,r2));pk(a,b))^+$
7. $(pk(a,n(b,r2));pk(a,b))^-$
8. $(pk(a,n(b,r2)))^+$
9. $(pk(a,i)^-$
10. $(pk(a,n(b,r2)))^-$
11. $(pk(i,a);pk(a,n(b,r2)))^+$
12. $pk(a,i;n(b,r2))^-$
13. $(pk(i,n(b,r1);n(a,r1);a))^+$
14. $(pk(i,n(b,r2));pk(i,n(a,r1));pk(i,a))^-$
15. $(pk(i,n(b,r2)))^+$
16. $(pk(i,n(b,r2)))^-$
17. $(n(b,r2))^+$
18. $(n(b,r2))^-$
19. $(pk(b,n(b,r2)))^+$
20. $(pk(b,n(b,r2)))^-$

Example: Diffie-Hellman Protocol

Protocol (text-book)

$A \longrightarrow B : (A;B;exp(g,N_A))$
$B \longrightarrow A : (A;B;exp(g,N_B))$
$A \longrightarrow B : e(exp(exp(g,N_B),N_A),sec(A,B)\ )$

Protocol (strand spaces)

$$
\begin{array}{l}
:: r1, r2 :: [nil \mid (A;B;exp(g,n(A,r1)))^{+}, \\
\qquad (A;B;XB)^{-}, \\
\qquad e(exp(XB,n(A,r1)),sec(A,r2))^{+}] \\
:: r3 :: [nil \mid (A;B;XA)^{-}, \\
\qquad (A;B;exp(g,n(B,r3)))^{+}, \\
\qquad (e(exp(XA,n(B,r3)),S))^{-}]
\end{array}
$$

Intruder capabilities

$[nil \mid (M_1;M_2)^{-}, M_1^{+}]$
$[nil \mid (M_1;M_2)^{-}, M_2^{+}]$
$[nil \mid M_1^{-}, M_2^{-}, (M_1;M_2)^{+}]$
$[nil \mid K^{-}, M^{-}, e(K,M)^{+}]$
$[nil \mid K^{-}, M^{-}, d(K,M)^{+}]$
$[nil \mid N1^{-}, N2^{-}, (N1 * N2)^{+}]$
$[nil \mid E^{-}, N^{-}, exp(E,N)^{+}]$

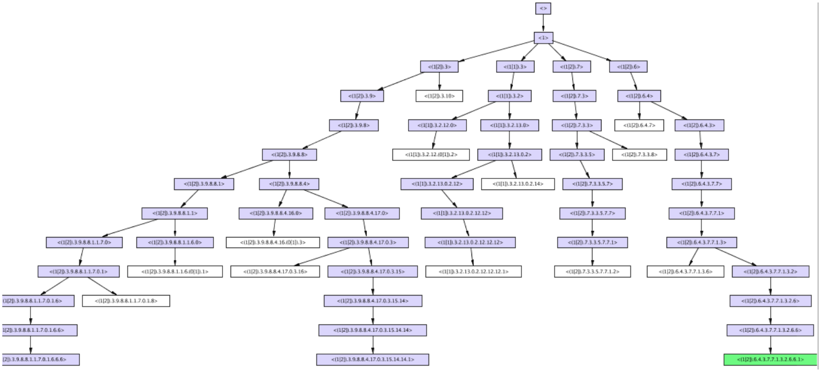
Equational Theory Algebraic properties

$B = \{\ (X * Y) * Z = X * (Y * Z),\ (X * Y) = Y * X\ \}$
$\Delta = \{\ dec(K,enc(K,X)) = X, exp(exp(W,Y),Z) = exp(W,Y * Z)\ \}$

Diffie-Helllman: Attack State Pattern

$$:: r' :: [\, (A; B; Y)^-, (B; A; exp(g, n(B, r')))^+, (e(exp(Y, n(B, r')), sec(a, r'')))^- \mid nil \,]$$
$$\&\ SS\ \&\ (sec(a, r'') {\in} \mathcal{I},\ IK)$$

Diffie-Helllman: Attack Space



Diffie-Helllman: Initial State

$$
\begin{array}{l}
[nil \mid exp(g, n(a,r)))^-, Z^-, exp(g, Z * n(a,r))^+] \ \& \\
[nil \mid exp(g, Z * n(a,r))^-, e(exp(g, Z * n(a,r)), sec(a,r''))^-, sec(a,r'')^+] \ \& \\
[nil \mid exp(g, n(b,r')))^-, W^-, exp(g, W * n(b,r'))^+] \ \& \\
[nil \mid exp(g, W * n(b,r'))^-, sec(a,r'')^-, e(exp(g, W * n(b,r')), sec(a,r''))^+] \ \& \\
[nil \mid (a; b; exp(g, n(b,r')))^-, (b; exp(g, n(b,r')))^+] \ \& \\
[nil \mid (b; exp(g, n(b,r')))^-, exp(g, n(b,r'))^+] \ \& \\
[nil \mid (a; B'; exp(g, n(a,r)))^-, (B'; exp(g, n(a,r)))^+] \ \& \\
[nil \mid (B'; exp(g, n(a,r)))^-, exp(g, n(a,r))^+] \ \& \\
:: r' :: \\
[nil \mid (a; b; exp(g, W))^-, (a; b; exp(g, n(b,r')))^+, e(exp(g, W * n(b,r')), sec(a,r''))^-] \ \& \\
:: r'', r :: \\
[nil \mid (a; B'; exp(g, n(a,r)))^+, (a; B'; exp(g, Z))^-, e(exp(g, Z * n(a,r)), sec(a,r''))^+]
\end{array}
$$

Diffie-Helllman: Attack sequence

$1.(a;b;exp(g,W))^-$
$2.(a;b;exp(g,n(b,r')))^+$
$3.(a;b;exp(g,n(b,r')))^-$
$4.(b;exp(g,n(b,r')))^+$
$5.(b;exp(g,n(b,r')))^-$
$6.(exp(g,n(b,r')))^+$
$7.(exp(g,n(b,r')))^-$
$8.W^-$
$9.exp(g,W*n(b,r'))^+$
$10.(a;B';exp(g,n(a,r)))^+$
$11.(a;B';exp(g,n(a,r)))^-$
$12.(B';exp(g,n(a,r)))^+$
$13.(B';exp(g,n(a,r)))^-$
$14.(exp(g,n(a,r)))^+$
$15.(exp(g,n(a,r)))^-$
$16.Z^-$
$17.exp(g,Z*n(a,r))^+$
$18.(a;B';exp(g,Z))^-$
$19.e(exp(g,Z*n(a,r)),sec(a,r''))^+$
$20.e(exp(g,Z*n(a,r)),sec(a,r''))^-$
$21.exp(g,Z*n(a,r))^-$
$22.sec(a,r'')^+$
$23.exp(g,W*n(b,r'))^-$
$24.sec(a,r'')^-$
$25.e(exp(g,W*n(b,r'),sec(a,r''))^+$
$26.e(exp(g,W*n(b,r')).sec(a,r''))^-$

Many thanks