Graduate Course on **Computer Security**

Lecture 7: Specification Languages

Iliano Cervesato `iliano@itd.nrl.navy.mil`

ITT Industries, Inc @ NRL - Washington DC

http://www.cs.stanford.edu/~iliano/

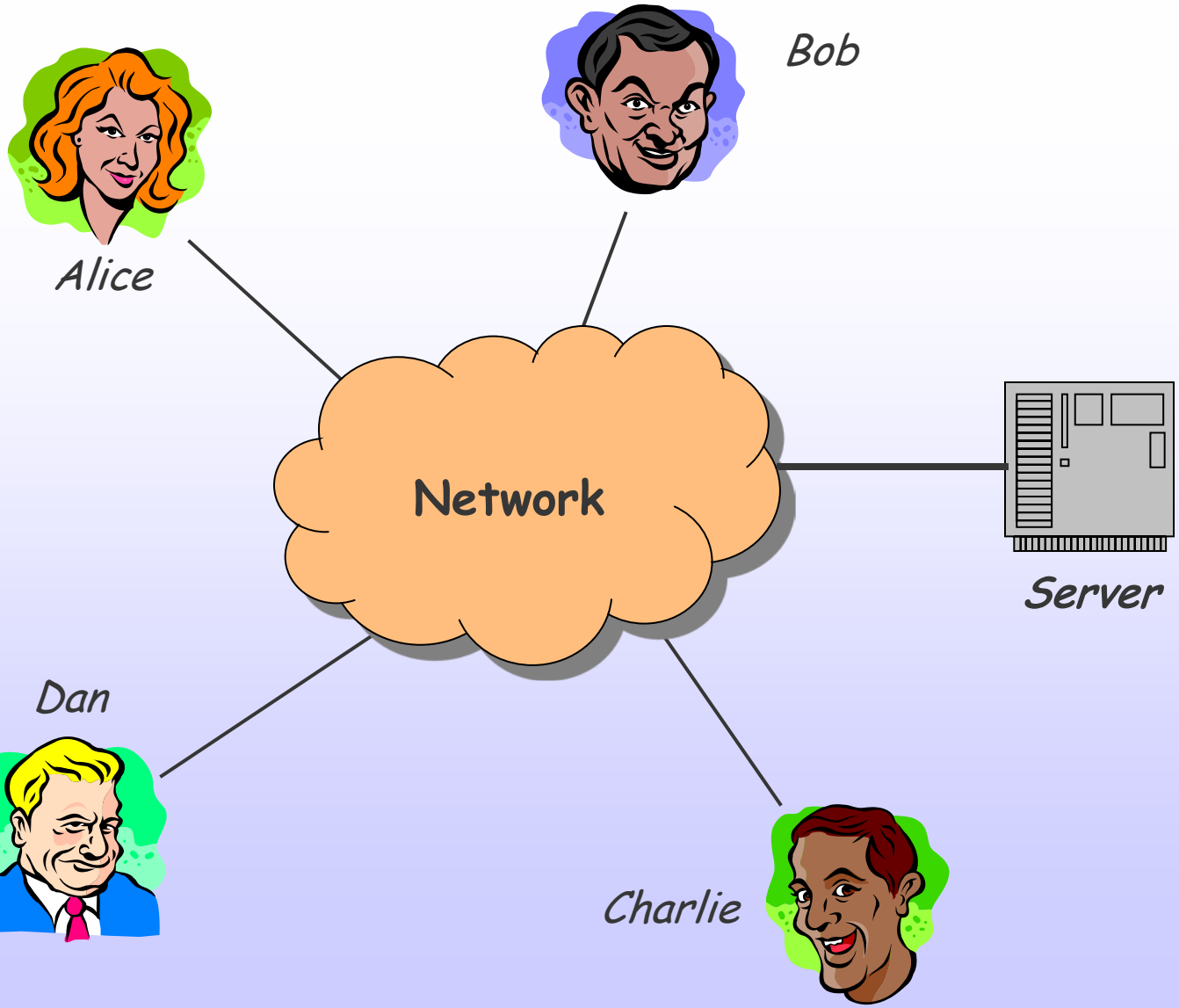
Outline

- Dolev-Yao model
- Specification
  - Evaluation criteria
  - Some languages
    - "Usual notation"
    - BAN logic
    - Spi calculus
    - Strand spaces
    - Inductive methods
    - CAPSL
- MSR
  - Motivations
  - Syntax
  - Type checking
  - DAS – Data Access Specification
  - Execution

Why is Protocol Analysis Difficult?

- Subtle cryptographic primitives
  - Dolev-Yao abstraction
    - Lecture 4 – a bit more in this lecture
- Distributed hostile environment
  - "Prudent engineering practice"
    - Lecture 4
- Inadequate specification languages
    - This lecture

Dolev-Yao Network Model

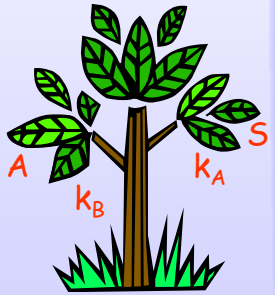


The Dolev-Yao Model of Security

- Symbolic data
  - No bits

- Black-box cryptography
  - No guessing of keys

- Partially abstract data access
  - Knowledge soup



- Found in most protocol analysis tools
  - Tractability

Perfect Cryptography

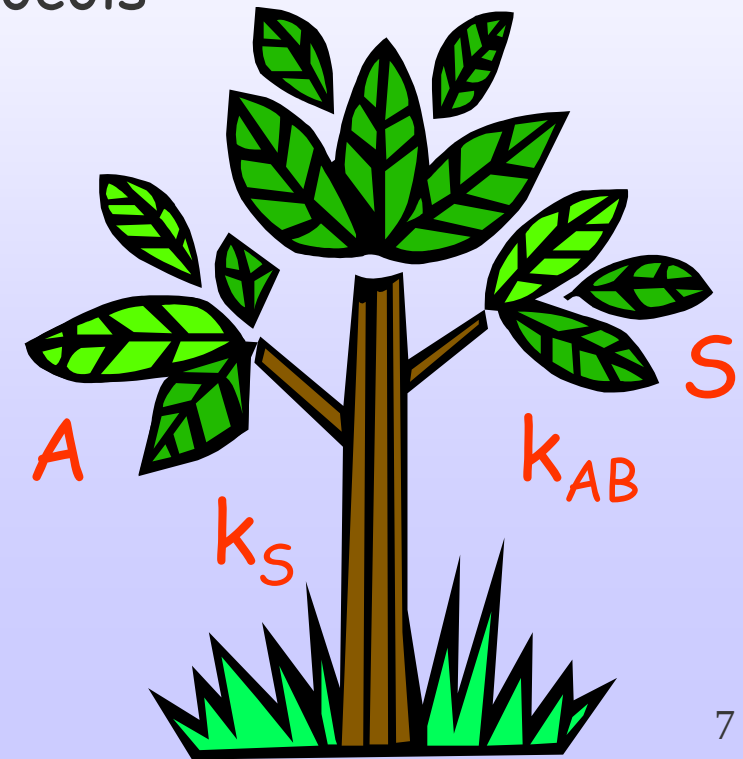
- k^{-1} is needed to decrypt $\{m\}_k$
  - k^{-1} is just k for shared key ciphers
- No collisions
  - $\{m_1\}_{k_A} = \{m_2\}_{k_B}$ *iff* $m_1 = m_2$ and $k_A = k_B$
  - $\{m\}_k = n$ *never*
  - $\{m\}_k = (m_1\ m_2)$ *never*

  We will relax this to handle type violations

Public Knowledge Soup

- Free access to auxiliary data
  - Abstracts actual mechanisms
    - Transmission of certificates
    - Invocation of subprotocols
    - Caching
- But …

  not all data are public
    - keys
    - secrets



Why is specification important?

good

- Documentation
  - Communicate
- Engineering
  - Implementation
  - Verification tools
- Science
  - Foundations
  - Assist engineering

Languages to Specify What?

- Message flow
- Message constituents
- Operating environment
- Protocol goals

Desirable Properties

- Unambiguous
- Simple
- Flexible
  - Adapts to protocols
- Powerful
  - Applies to a wide class of protocols
- Insightful
  - Gives insight about protocols

Language Families

- "Usual notation"
  - (user interfaces)
- Knowledge logic
  - BAN
- Process theory
  - Spi-calculus
  - Strands
  - MSR
  - FDR, Casper
  - Petri nets
- Inductive methods
- Temporal logic
- Automata
  - CAPSL
  - NRL Protocol Analyzer
  - Murϕ
- ...

Why so many?

- Experience from mature fields
- Unifying problem
- Scientifically intriguing
- Funding opportunities

Convergence of approaches

Running Example

Needham-Schroeder public key protocol (fragment)

- Devised in '78
- Broken in '95 !

But …

- purely academic
- attack subject to interpretation

Example of weak specification !

“Usual Notation”

$$A \rightarrow B: \{n_A, A\}_{k_B}$$

$$B \rightarrow A: \{n_A, n_B\}_{k_A}$$

$$A \rightarrow B: \{n_B\}_{k_B}$$

Evaluation of the Usual Notation

- Flow
  - Expected run
- Constituents
  - Side remarks
- Environment
  - Side remarks
- Goals
  - Side remarks

- Unambiguous ☹
- Simple ☺
- Flexible ☺
- Powerful ☺
- Insightful 😐

BAN Logic

[Burrows, Abadi, Needham]

- Roots in *belief logic*
  - Reason about knowledge as protocol unfolds
  - Security: principals share same view
- Specification
  - Usual notation
  - "Idealized protocol"
  - Assumptions
  - Goals
- Verification
  - Logical inference

BAN Idealization

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

n_B is a shared secret between A and B

n_A provides evidence to the fact that ...

$A \rightarrow B: \{\langle A \Leftarrow^{n_B} \Rightarrow B \rangle_{n_A}\}_{k_A}$

$A \rightarrow B: \{\langle A \Leftarrow^{n_A} \Rightarrow B, B \mid\equiv A \Leftarrow^{n_B} \Rightarrow B \rangle_{n_B}\}_{k_B}$

Believes

(more readable syntax proposed later)

BAN Assumptions

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

k_A is the public key of A

n_A is fresh

- $A \mid\equiv \overset{k_A}{\hookrightarrow} A$
- $A \mid\equiv \overset{k_B}{\hookrightarrow} B$
- $A \mid\equiv \#\, n_A$
- $A \mid\equiv A \overset{n_A}{\Leftrightarrow} B$

- $B \mid\equiv \overset{k_B}{\hookrightarrow} B$
- $B \mid\equiv \overset{k_A}{\hookrightarrow} A$
- $B \mid\equiv \#\, n_B$
- $B \mid\equiv A \overset{n_B}{\Leftrightarrow} B$

BAN Goals

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

Authentication goals expressed in terms of

- Mutual beliefs
- Beliefs about freshness

- $B \mid\equiv A \mid\equiv A \Leftarrow^{n_A} \Rightarrow B$
- $A \mid\equiv B \mid\equiv A \Leftarrow^{n_B} \Rightarrow B$

- $A \mid\equiv \# n_B$
- $B \mid\equiv \# n_A$

Formally derived from BAN rules

Evaluation of BAN

- Flow
  - Idealized run
- Constituents
  - Assumptions
- Environment
  - Implicit
- Goals
  - BAN formulas

- Unambiguous 😐
- Simple ☹
- Flexible ☹
- Powerful ☹
- Insightful 😐

The Spi-Calculus

[Abadi, Gordon]

- π-calculus with cryptographic constructs
- Specification
  - 1 process for each role
  - Instance to be studied
  - *Intruder not explicitly modeled*
- Verification
  - Process equivalence to reference process

The Syntax of Spi

Spi: NS Initiator

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

$init(A, B, c_{AB}, k_B^+, k_A^-) =$

$(\nu n_A)\ c_{AB}\langle \{|A, n_A|\}_{k_B^+} \rangle .$

$c_{AB}(x)$. case x of $\{|y|\}_{k_A^-}$ in

let $(y_1, y_2) = y$ in $[y_1 \text{ is } n_A]$

$c_{AB}\langle \{|y_2|\}_{k_B^+} \rangle .$

0

Spi: NS Responder

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

$resp(B, A, c_{AB}, k_A^+, k_B^-) =$

$c_{AB}(x)$. case x of $\{|y|\}_{k_B^-}$ in

let $(y_1, y_2) = y$ in $[y_1 \text{ is } A]$

$(\nu n_B)\ c_{AB}\langle \{|\ y_2, n_B\ |\}_{k_A^+} \rangle$.

$c_{AB}(x')$. case x' of $\{|y'|\}_{k_B^-}$ in $[y' \text{ is } n_B]$

0

Spi: NS Instance

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

$inst(A, B, c_{AB}) =$

$(\nu k_A)\ (\nu k_B)$

$(\quad init(A, B, c_{AB}, k_B^+, k_A^-)$

$\quad |\quad resp(B, A, c_{AB}, k_A^+, k_B^-))$

Evaluation of Spi

- Flow
  - Role-based
- Constituents
  - Informal math.
- Environment
  - Implicit
- Goals
  - Reference process

- Unambiguous 😐
- Simple ☺
- Flexible ☺
- Powerful ☺
- Insightful ☺

Strand Spaces

[Guttman, Thayer]

- Roots in *trace theory*
  - Lamport's causality
  - Mazurkiewicz's traces
- Specification
  - Strands
  - Sets of principals, keys, …
- Verification
  - Authentication tests
  - Model checking

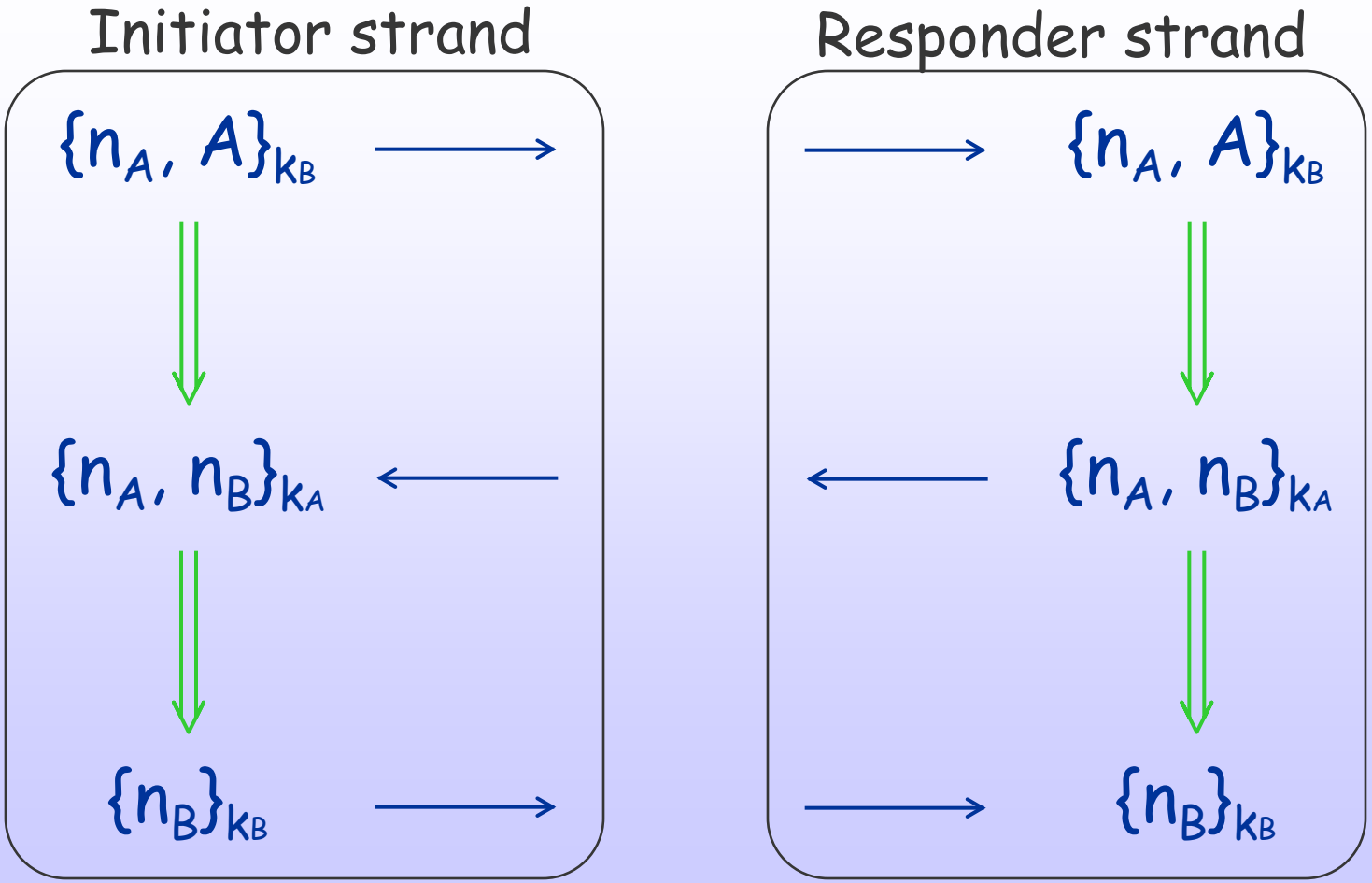
Strands

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]



Evaluation of Strands

- Flow
  - Role-based
- Constituents
  - Informal math.
- Environment
  - Side remarks
- Goals
  - Side remarks

- Unambiguous 😐
- Simple ☺
- Flexible ☹
- Powerful ☹
- Insightful ☺

Inductive Methods [Paulson]

- Protocol inductively defines *traces*
- Specification
  - 1 inductive rule for each protocol rule
  - Universal intruder based on language
- Verification
  - Theorem proving (Isabelle HOL)
- Related methods
  - [Bolignano]

IMs: NS

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

NS1 [evs $\in$ ns; $A \neq B$; Nonce $N_A \notin$ **used** evs]

$\Rightarrow$ Says A B $\{$Nonce N_A, Agent A$\}_{KB}$ # evs $\in$ ns

NS2 [evs $\in$ ns; $A \neq B$; Nonce $N_B \notin$ **used** evs;

Says A' B $\{$Nonce N_A, Agent A$\}_{KB} \in$ **set** evs]

$\Rightarrow$ Says B A $\{$Nonce N_A, Nonce $N_A\}_{KA}$ # evs $\in$ ns

NS3 [evs $\in$ ns;

Says A B $\{$Nonce N_A, Agent A$\}_{KB} \in$ **set** evs;

Says B' A $\{$Nonce N_A, Nonce $N_A\}_{KA} \in$ **set** evs]

$\Rightarrow$ Says A B $\{$Nonce $N_A\}_{KB}$ # evs $\in$ ns

IMs: Environment

Nil $[] \in ns$

Fake $[evs \in ns; B \neq Spy; X \in \mathbf{synth}(\mathbf{analz}\ (\mathbf{spies}\ evs))]$
$\Rightarrow \underline{Says}\ Spy\ B\ X\ \#\ evs \in ns$

synth, **analz**, **spies**, … protocol independent

Evaluation of Inductive Methods

- Flow
  - Trace-based
- Constituents
  - Formalized math.
- Environment
  - Immutable
- Goals
  - Imposs. traces

- Unambiguous ☺
- Simple 😐
- Flexible ☺
- Powerful ☺
- Insightful ☺

CAPSL

[Millen]

- Ad-hoc model checker
- Specification
  - Special-purpose language
  - Intruder built-in
- Implementation
  - CIL [Denker] -> similar to MSR
- Related systems
  - Murϕ [Shmatikov, Stern]
  - SMV [Clarke, Jha, Marrero]

CAPSL

$A \rightarrow B: \{A, n_A\}_{k_B}$

$B \rightarrow A: \{n_A, n_B\}_{k_A}$

$A \rightarrow B: \{n_B\}_{k_B}$

NS-PK [3-5]

```
PROTOCOL NS;

VARIABLES
    A, B: PKUser;
    Na, Nb: Nonce,
    CRYPTO

ASSUMPTIONS
    HOLDS A: B;
```

```
MESSAGES
    A -> B : {A, Na}pk(B);
    B -> A : {Na,Nb}pk(A);
    A -> B : {Nb}pk(B);

GOALS
    SECRET Na;
    SECRET Nb;
    PRECEDES A: B | Na;
    PRECEDES B: A | Nb;
END;
```

Evaluation of CAPSL

- Flow
  - Explicit run
- Constituents
  - Declarations
- Environment
  - Implicit
- Goals
  - Properties

- Unambiguous ☺
- Simple ☺
- Flexible 😐
- Powerful 😐
- Insightful 😐

MSR

[Cervesato, Durgin, Lincoln, Mitchell, Scedrov]

A specification language based on

- MultiSet Rewriting with existentials

- MSR 1.0
  - Designed to prove the undecidability of protocol correctness verification
  - Poor specification language
    - Error-prone
    - Limited automated assistance
    - Limited support for verification

- MSR 2.0
  - Redesign of MSR 1.0 as a specification language
    - Easy to use
    - Support for automation
  - New background in type-theory
  - Margin for verification
    - Current techniques can be adapted

Evaluation of MSR 2.0

- Flow
  - Role-based
- Constituents
  - Strong typing
- Environment
  - *In part*
- ~~Goals~~

- Unambiguous ☺
- Simple ☺
- Flexible ☺
- Powerful ☺
- Insightful ☺

Roadmap to MSR

- Step-by-step specification of example
  - Neuman-Stubblebine protocol
- Language description
  - Syntax
  - Typing
  - Data Access Control – DAS
  - Execution semantics
  - Properties
- More examples
  - NS-PK[3-5]

Multiset Rewriting

- Multiset: set with repetitions allowed
- Rewrite rule:

$$r: N_1 \rightarrow N_2$$

- Application

$$\underbrace{M_1}_{M', N_1} \xrightarrow{r} \underbrace{M_2}_{M', N_2}$$

- Multi-step transition, reachability

NSt-I: B's Role

$A \rightarrow B$: A, n_A

$B \rightarrow S$: $B, \{A, n_A, t_B\}_{k_{BS}}, n_B$

$S \rightarrow A$: $\{B, n_A, k_{AB}, t_B\}_{k_{AS}}, \{A, k_{AB}, t_B\}_{k_{BS}}, n_B$

$A \rightarrow B$: $\{A, k_{AB}, t_B\}_{k_{BS}}, \{n_B\}_{k_{AB}}$

$A \rightarrow B$: A, n_A

$B \rightarrow S$: $B, \{A, n_A, t_B\}_{k_{BS}}, n_B$

$S \rightarrow A$: $\{B, n_A, k_{AB}, t_B\}_{k_{AB}}, T$

$A \rightarrow B$: $T, \{n_B\}_{k_{AB}}$

where $T = \{A, k_{AB}, t_B\}_{k_{BS}}$

Neuman-Stubblebine – phase I

NSt-I: S's Role

$A \rightarrow B$: A, n_A

$B \rightarrow S$: $B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A$: $\{B, n_A, k_{AB}, t_B\}_{K_{AS}}, \{A, k_{AB}, t_B\}_{K_{BS}}, n_B$

$A \rightarrow B$: $\{A, k_{AB}, t_B\}_{K_{BS}}, \{n_B\}_{K_{AB}}$

$A \rightarrow B$: A, n_A

$B \rightarrow S$: $B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A$: $\{B, n_A, k_{AB}, t_B\}_{K_{AS}}, T$

$A \rightarrow B$: $T, \{n_B\}_{K_{AB}}$

where $T = \{A, k_{AB}, t_B\}_{K_{BS}}$

Neuman-Stubblebine – phase I

NSt-I: *A*'s Role

$A \rightarrow B: A, n_A$

$B \rightarrow S: B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A: \{B, n_A, k_{AB}, t_B\}_{K_{AS}}, \underbrace{\{A, k_{AB}, t_B\}_{K_{BS}}}_{X}, n_B$

$A \rightarrow B: \underbrace{\{A, k_{AB}, t_B\}_{K_{BS}}}_{X}, \{n_B\}_{k_{AB}}$

Ticket

$A \rightarrow B: A, n_A$

$B \rightarrow S: B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A: \{B, n_A, k_{AB}, t_B\}_{K_{AS}}, T$

$A \rightarrow B: T, \{n_B\}_{k_{AB}}$

where $T = \{A, k_{AB}, t_B\}_{K_{BS}}$

Neuman-Stubblebine - phase I

Sending / Receiving Messages

$\bullet \rightarrow N(A, n_A)$

Network predicate

N(m): m is a message in transit

$N(\{B, n_A, k_{AB}, t_B\}_{k_{AS}}, X, n_B) \rightarrow N(X, \{n_B\}_{k_{AB}})$

Terms

- Atomic terms
  - Principal names A
  - Keys k
  - Nonces n
  - …
- Term constructors
  - (_ _)
  - {_}_
  - …

Definable

Nonces

$A \rightarrow B: A, n_A$

$B \rightarrow S: B, \{A, n_A, t_B\}_{k_{BS}}, n_B$

$S \rightarrow A: \{B, n_A, k_{AB}, t_B\}_{k_{AB}}, T$

$A \rightarrow B: T, \{n_B\}_{k_{AB}}$

Neuman-Stubblebine – phase I

$\bullet \rightarrow \exists n_A.\ N(A, n_A)$

Existential variables

n_A instantiated to a new constant

$N(\{B, n_A, k_{AB}, t_B\}_{k_{AS}}, X, n_B) \rightarrow N(X, \{n_B\}_{k_{AB}})$

MSet Rewriting with Existentials

- Multisets of 1st-order atomic formulas
- Rules:

$$r: F(\underline{x}) \rightarrow \exists \underline{n}.\ G(\underline{x},\underline{n})$$

- Application

$$\underbrace{M_1}_{M',\ F(\underline{t})} \xrightarrow{r} \underbrace{M_2}_{M',\ G(\underline{t},\underline{c})}$$

$\underline{c}$ not in M_1

Sequencing Actions

$A \rightarrow B: A, n_A$

$B \rightarrow S: B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A: \{B, n_A, k_{AB}, t_B\}_{K_{AB}}, T$

$A \rightarrow B: T, \{n_B\}_{K_{AB}}$

Neuman-Stubblebine – phase I

$\exists L.$ Fresh!

$\bullet \rightarrow \exists n_A.$ $N(A, n_A)$ $L(A, n_A)$

Role state predicate

$L(A, n_A)$
$N(\{B, n_A, k_{AB}, t_B\}_{K_{AS}}, X, n_B) \rightarrow N(X, \{n_B\}_{K_{AB}})$

Role State Predicates

$L_l(A, t, \ldots, t)$

- Hold data local to a role instance
  - Lifespan = role
- Invoke next rule
  - L_l = control
  - $(A, t, \ldots, t)$ = data

Remembering Things

$A \rightarrow B: A, n_A$

$B \rightarrow S: B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A: \{B, n_A, k_{AB}, t_B\}_{K_{AB}}, T$

$A \rightarrow B: T, \{n_B\}_{K_{AB}}$

Neuman-Stubblebine – phase I

$$\exists L. \quad \bullet \rightarrow \exists n_A. \; L(A, n_A), N(A, n_A)$$

$$L(A, n_A), N(\{B, n_A, k_{AB}, t_B\}_{K_{AS}}, X, n_B) \rightarrow N(X, \{n_B\}_{K_{AB}}), Tkt_A(B, k_{AB}, X)$$

Memory predicate

Memory Predicates

$$M_A(t, \ldots, t)$$

- Hold private info. across role exec.
- Support for subprotocols
  - Communicate data
  - Pass control
- Interface to outside system
- Implements intruder

Role Owner

$A \rightarrow B: A, n_A$

$B \rightarrow S: B, \{A, n_A, t_B\}_{K_{BS}}, n_B$

$S \rightarrow A: \{B, n_A, k_{AB}, t_B\}_{K_{AB}}, T$

$A \rightarrow B: T, \{n_B\}_{K_{AB}}$

Neuman-Stubblebine – phase I

Role owner

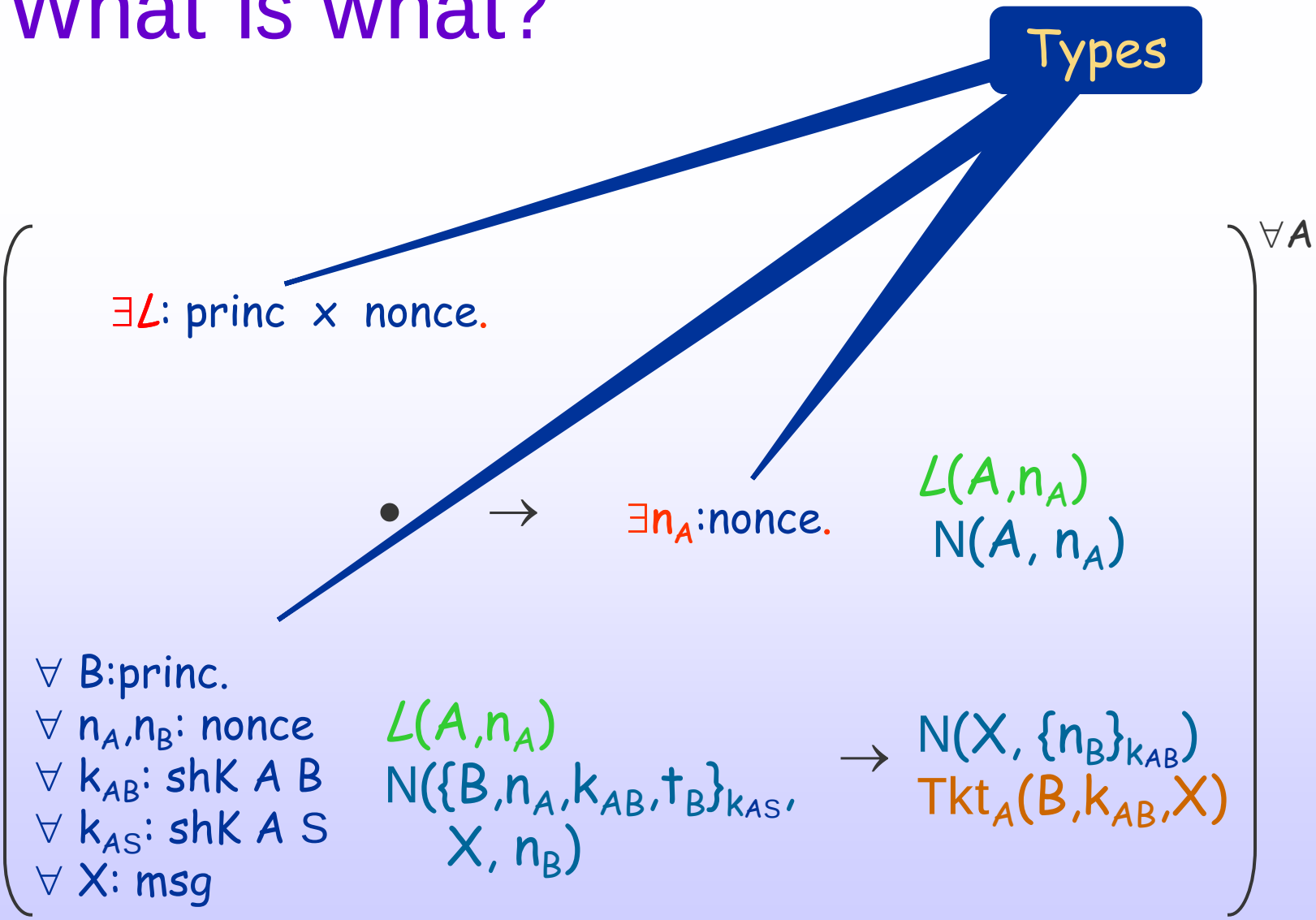
The principal executing the role

$\forall A$

$$\exists L.\ \bullet \rightarrow \exists n_A.\ L(A, n_A)\ N(A, n_A)$$

$$L(A, n_A)\ N(\{B, n_A, k_{AB}, t_B\}_{K_{AS}}, X, n_B) \rightarrow N(X, \{n_B\}_{K_{AB}})\ Tkt_A(B, k_{AB}, X)$$

What is what?


Types
∀A
∃L: princ x nonce.
•
→
∃n_A:nonce.
L(A,n_A)
N(A, n_A)
∀ B:princ.
∀ n_A,n_B: nonce
∀ k_{AB}: shK A B
∀ k_{AS}: shK A S
∀ X: msg
L(A,n_A)
N({B,n_A,k_{AB},t_B}$_{kAS}$,
X, n_B)
→
N(X, {n_B}$_{kAB}$)
Tkt$_A$(B,k_{AB},X)

Types of Terms

- A: princ
- n: nonce
- k: shK A B
- k: pubK A
- k': privK k
- … (definable)

Types can depend on term

- Captures relations between objects
  - Static
  - Local
  - Mandatory

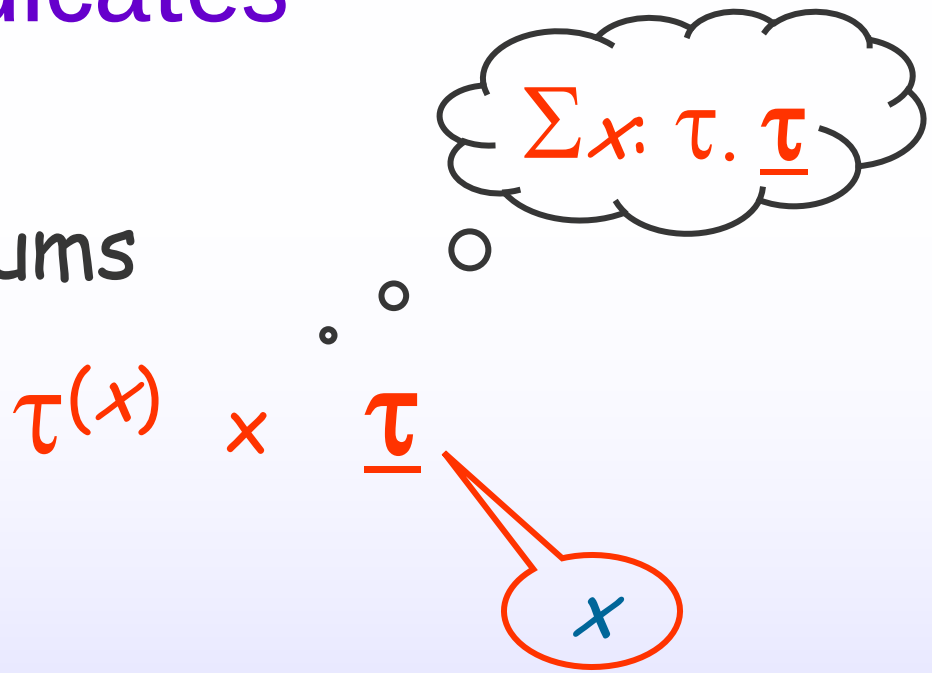
Subtyping

$$\tau :: msg$$

- Allows atomic terms in messages
- Definable
  - Non-transmittable terms
  - Sub-hierarchies

Type of predicates

- Dependent sums



- Forces associations among arguments
  - E.g.: $princ^{(A)}$ x $pubK\ A^{(k_A)}$ x $privK\ k_A$

Type Checking

$\Sigma \vdash P$

$\Gamma \vdash t : \tau$

t has type τ in Γ

P is well-typed in Σ

- Catches:
  - Encryption with a nonce
  - Transmission of a long term key

Typing Terms

$$\frac{\Gamma \vdash t_1 : msg \quad \Gamma \vdash t_2 : msg}{\Gamma \vdash t_1\ t_2 : msg}$$

$$\frac{\Gamma \vdash t : msg \quad \Gamma \vdash k : shK\ A\ B}{\Gamma \vdash \{t\}_k : msg}$$

$$\frac{}{\Gamma, x : \tau, \Gamma' \vdash x : \tau}$$

- Similar rules for
  - Public key encryption
  - Digital signatures, …

Typing Types

$$\frac{}{\Gamma \vdash msg} \qquad \frac{}{\Gamma \vdash nonce} \qquad \frac{}{\Gamma \vdash time}$$

$$\frac{\Gamma \vdash A : princ \qquad \Gamma \vdash B : princ}{\Gamma \vdash shK\ A\ B}$$

- Typing for dependent types relies on typing for terms

Some Subtyping Rules

$$\frac{\tau' :: \tau \quad \Gamma \vdash t : \tau'}{\Gamma \vdash t : \tau}$$

$$\frac{}{princ :: msg} \qquad \frac{}{nonce :: msg} \qquad \frac{}{time :: msg}$$

$$\frac{}{shK\ A\ B :: msg}$$

$$\frac{}{pubK\ A\ B :: msg}$$

- No rule for public keys
  - Prevent transmission

Typing Tuples and Tuple Types

$$\frac{}{\Gamma \vdash \bullet : \bullet} \qquad \frac{\Gamma \vdash x : \tau \quad \Gamma \vdash \underline{t} : [t/x]\underline{\tau}}{\Gamma \vdash (x,\underline{t}) : \tau^{(x)} \times \underline{\tau}}$$

$$\frac{}{\Gamma \vdash \bullet} \qquad \frac{\Gamma \vdash x : \tau \quad \Gamma, x{:}\tau \vdash \underline{\tau}}{\Gamma \vdash \tau^{(x)} \times \underline{\tau}}$$

Typing Predicates

$$\frac{\Gamma \vdash t:msg}{\Gamma \vdash N(t)} \qquad \frac{\Gamma, L:\underline{\tau}, \Gamma' \vdash \underline{t} : \underline{\tau}}{\Gamma, L:\underline{\tau}, \Gamma' \vdash L(\underline{t})}$$

$$\frac{\Gamma, M\_:\underline{\tau}, \Gamma' \vdash (A,\underline{t}) : \underline{\tau}}{\Gamma, M\_:\underline{\tau}, \Gamma' \vdash M_A(\underline{t})}$$

Typing Protocol Rules

$$\frac{\Gamma \vdash lhs \qquad \Gamma \vdash rhs}{\Gamma \vdash lhs \rightarrow rhs}$$

$$\frac{\Gamma \vdash \tau \qquad \Gamma, x{:}\tau \vdash r}{\Gamma \vdash \forall\, x{:}\tau.\ r}$$

Typing Roles

$$\frac{}{\Gamma \vdash \bullet} \qquad \frac{\Gamma \vdash \underline{\tau} \qquad \Gamma, L{:}\underline{\tau} \vdash \rho}{\Gamma \vdash \exists\, L{:}\underline{\tau}.\ \rho}$$

$$\frac{\Gamma \vdash r \qquad \Gamma \vdash \rho}{\Gamma \vdash r, \rho}$$

Typing Protocol Theories

$$\frac{}{\Gamma \vdash \bullet} \qquad \frac{\Sigma \vdash P \qquad \Sigma, A{:}\mathrm{princ} \vdash \rho}{\Sigma \vdash P, \rho^{\forall A}}$$

$$\frac{\Sigma, A{:}\mathrm{princ} \vdash P \qquad \Sigma, A{:}\mathrm{princ} \vdash \rho}{\Sigma, A{:}\mathrm{princ} \vdash P, \rho^{A}}$$

Data Access Specification – DAS

- Catches
  - A signing/encrypting with B's key
  - A accessing B's private data, …
- Gives meaning to Dolev-Yao intruder

An Overview of Access Control

- Interpret incoming information
  - Collect received data
  - Access unknown data
- Construct outgoing information
  - Generate data
  - Use known data
  - Access new data
- Verify access to data

Processing a Rule

Knowledge set:

Collects what A knows

Context

$$\frac{\Gamma \vdash_A lhs \gg \Delta \qquad \Gamma; \Delta \vdash_A rhs}{\Gamma \vdash_A lhs \rightarrow rhs}$$

Processing Predicates on the LHS

- Network messages

$$\frac{\Gamma ; \Delta \Vdash_A t \gg \Delta'}{\Gamma ; \Delta \Vdash_A N(t) \gg \Delta'}$$

- Memory predicates

$$\frac{\Gamma ; \Delta \Vdash_A t_1,\ldots,t_n \gg \Delta'}{\Gamma ; \Delta \Vdash_A M_A(t_1,\ldots,t_n) \gg \Delta'}$$

Interpreting Data on the LHS

- Pairs

$$\frac{\Gamma;\ \Delta \Vdash_A t_1,\ t_2 \gg \Delta'}{\Gamma;\ \Delta \Vdash_A (t_1,\ t_2) \gg \Delta'}$$

- Encrypted terms

$$\frac{\Gamma;\ \Delta \Vdash_A k \gg \Delta' \qquad \Gamma;\ \Delta' \Vdash_A t \gg \Delta''}{\Gamma;\ \Delta \Vdash_A \{t\}_k \gg \Delta''}$$

- Elementary terms

$$\left\{ \begin{array}{l} \dfrac{}{\Gamma;\ (\Delta,x) \Vdash_A x \gg (\Delta,x)} \\[2ex] \dfrac{}{(\Gamma,x{:}\tau);\ \Delta \Vdash_A x \gg (\Delta,x)} \end{array} \right.$$

Accessing Data on the LHS

- Shared keys

$$\frac{}{\Gamma;\ (\Delta,k) \Vdash_A k \gg (\Delta,k)}$$

$$\frac{}{(\Gamma,x{:}\mathit{shK}\ A\ B);\ \Delta \Vdash_A x \gg (\Delta,x)}$$

- Public keys

$$\frac{}{(\Gamma,k{:}\mathit{pubK}\ A,k'{:}\mathit{privK}\ k);\ (\Delta,k') \Vdash_A k \gg (\Delta,k')}$$

$$\frac{}{(\Gamma,k{:}\mathit{pubK}\ A,k'{:}\mathit{privK}\ k);\ \Delta \Vdash_A k \gg (\Delta,k')}$$

Generating Data on the RHS

- Nonces

$$\frac{(\Gamma, x{:}\mathrm{nonce});\ (\Delta, x) \vdash_A \mathrm{rhs}}{\Gamma;\ \Delta \vdash_A \exists x{:}\mathrm{nonce}.\ \mathrm{rhs}}$$

Constructing Terms on the RHS

- Pairs

$$\frac{\Gamma; \Delta \Vdash_A t_1 \qquad \Gamma; \Delta \Vdash_A t_2}{\Gamma; \Delta \Vdash_A (t_1, t_2)}$$

- Shared-key encryptions

$$\frac{\Gamma; \Delta \Vdash_A t \qquad \Gamma; \Delta \Vdash_A k}{\Gamma; \Delta \Vdash_A \{t\}_k}$$

Accessing Data on the RHS

- Principal

$$\frac{}{\Gamma, B{:}princ \Vdash_A B}$$

- Shared key

$$\frac{}{\Gamma, B{:}princ, k{:}shK\ A\ B \Vdash_A k}$$

- Public key

$$\frac{}{\Gamma, B{:}princ, k{:}pubK\ B \Vdash_A k}$$

- Private key

$$\frac{}{\Gamma, k{:}pubK\ A, k'{:}privK\ k \Vdash_A k'}$$

NS-I: B's point of view

$A \rightarrow B: \quad A, n_A$

$B \rightarrow S: \quad B, \{A, n_A, T_B\}_{kBS}, n_B$

$S \rightarrow A: \quad \{B, n_A, k_{AB}, T_B\}_{kAS}, \{A, k_{AB}, T_B\}_{kBS}, n_B$

$A \rightarrow B: \quad \{A, k_{AB}, T_B\}_{kBS}, \{n_B\}_{kAB}$

NS-I: B's role

$\forall B$

$\exists \mathcal{L}$: princ$^{(B)}$ x princ x nonce x shK B S x nonce x time.

$\forall$ A:princ.
$\forall$ n_A: nonce
$\forall$ k_{BS}: shK B S
$\forall$ T_B: time

$N(A, n_A)$
$Clk_B(T_B)$
$\rightarrow$
$\exists n_B$:nonce.
$N(B, \{A, n_A, T_B\}_{kBS}, n_B)$
$Clk_B(T_B)$
$\mathcal{L}(B, A, n_A, k_{BS}, n_B, T_B)$

$\forall$ ....
$\forall$ k_{AB}: shK A B
$\forall$ n_B: nonce
$\forall$ T_{now}: time
$\forall$ T_V, T_e: time

$\mathcal{L}(B, A, n_A, k_{BS}, n_B, T_B)$
$N(\{A, k_{AB}, T_B\}_{kBS}, \{n_B\}_{kAB})$
$Val_B(A, T_B, T_V)$
$(T_e = T_B + T_V)$
$\rightarrow$
$Auth_B(A, k_{AB}, T_B, T_e)$
$Val_B(A, T_B, T_V)$

Constraint

Constraints

χ

- Guards over interpreted domain
  - Abstract
  - Modular
- Invoke constraint handler
- E.g.: timestamps
  - $(T_E = T_N + T_d)$
  - $(T_N < T_E)$

NS-I: S's point of view

$A \rightarrow B: \quad A, n_A$

$B \rightarrow S: \quad B, \{A, n_A, T_B\}_{kBS}, n_B$

$S \rightarrow A: \quad \{B, n_A, k_{AB}, T_B\}_{kAS}, \{A, k_{AB}, T_B\}_{kBS}, n_B$

$A \rightarrow B: \quad \{A, k_{AB}, T_B\}_{kBS}, \{n_B\}_{kAB}$

NS-I: S's role

Anchored role

$$\left(\begin{array}{l} \forall\ A,B\text{:princ.} \\ \forall\ k_{AS}\text{: shK A S} \\ \forall\ k_{BS}\text{: shK B S} \\ \forall\ n_A,n_B\text{: nonce} \\ \forall\ T_B\text{: time} \end{array}\right.\ N(B, \{A,n_A,T_B\}_{kBS}, n_B) \rightarrow \left.\begin{array}{l} \exists k_{AB}\text{: shK A B.} \\ N(\{B,n_A,k_{AB},T_B\}_{kAS}, \\ \quad \{A,k_{AB},T_B\}_{kBS}, \\ \quad n_B) \end{array}\right)_S$$

Neuman-Stubblebine – Phase II

$$A \rightarrow B: \; n'_A , \{A, k_{AB}, T_B\}_{kBS}$$

$$B \rightarrow A: \; n'_B , \{n'_A\}_{kAB}$$

$$A \rightarrow B: \; \{n'_B\}_{kAB}$$

NS-II: *A*'s role

$\forall A$

$\exists L$: princ$^{(A)}$ x princ$^{(B)}$ x shK A B x nonce.

$\exists n'_A$:nonce.

$\forall$ B:princ.
$\forall$ k_{AB}: shK A B
$\forall$ X: msg

$Tkt_A(B,k_{AB},X)$ $\rightarrow$ $N('n_A, X)$, $Tkt_A(B,k_{AB},X)$, $L(A, B, k_{AB}, n'_A)$

$\forall$ ....
$\forall$ n'_A, n'_B: nonce

$L(A, B, k_{AB}, n'_A)$, $N(n'_B, \{n'_A\}_{kAB})$ $\rightarrow$ $N(\{n'_B\}_{kAB})$

NS-II: B's role

$\forall B$

$\exists L$: princ$^{(B)}$ x princ$^{(A)}$ x shK A B x nonce.

$\forall\ n'_A$: nonce
$\forall\ k_{BS}$: shK B S
$\forall$ A:princ.
$\forall\ k_{AB}$: shK A B
$\forall\ T_B, T_e$: time
$\forall\ T_{now}$: time

$N(n'_A, \{A, k_{AB}, T_B\}_{kBS})$
$Auth_B(A, k_{AB}, T_B, T_e)$
$Clk_B(T_{now})$
$(T_{now} < T_e)$

$\rightarrow$

$\exists n'_B$:nonce.
$N(n'_B, \{n'_A\}_{kAB})$
$Auth_B(A, k_{AB}, T_B, T_e)$
$Clk_B(T_{now})$
$L(B, A, k_{AB}, n'_B)$

$\forall$ ....
$\forall\ n'_B$: nonce

$L(B, A, k_{AB}, n'_B)$
$N(\{n'_B\}_{kAB})$

$\rightarrow$

Summary: Rules

$\forall x_1: \tau_1.$
...
$\forall x_n: \tau_n.$

lhs → rhs

$\exists y_1: \tau'_1.$
...
$\exists y_{n'}: \tau'_{n'}.$

- N(t) Network
- $L(t, \ldots, t)$ Local state
- $M_A(t, \ldots, t)$ Memory
- χ Constraints

- N(t) Network
- $L(t, \ldots, t)$ Local state
- $M_A(t, \ldots, t)$ Memory

Summary: Roles

Role state pred. var. declarations

- Generic roles

$$\forall A \left(\begin{array}{l} \exists L: \tau'_1(x_1) \times \ldots \times \tau'_n(x_n) \\ \ldots \\ \forall x{:}\tau. \quad \text{lhs} \quad \exists y{:}\tau'. \quad \rightarrow \quad \text{rhs} \\ \ldots \quad \ldots \quad \ldots \\ \forall x{:}\tau. \quad \text{lhs} \quad \exists y{:}\tau'. \quad \rightarrow \quad \text{rhs} \end{array} \right)$$

Role owner

- Anchored roles

$$A \left(\begin{array}{l} \exists L: \tau'_1(x_1) \times \ldots \times \tau'_n(x_n) \\ \ldots \\ \forall x{:}\tau. \quad \text{lhs} \quad \exists y{:}\tau'. \quad \rightarrow \quad \text{rhs} \\ \ldots \quad \ldots \quad \ldots \\ \forall x{:}\tau. \quad \text{lhs} \quad \exists y{:}\tau'. \quad \rightarrow \quad \text{rhs} \end{array} \right)$$

Summary: Snapshots

Active role set

$C = [S]^{R}_{\Sigma}$

State

- $N(t)$
- $L_l(t, \ldots, t)$
- $M_A(t, \ldots, t)$

Signatur e

- $a : \tau$
- $L_l : \underline{\tau}$
- $M_{\_} : \underline{\tau}$

Summary: Execution Model

$$P \triangleright C \rightarrow C' \circ\circ\circ$$

1-step firing

- Activate roles
- Generates new role state pred. names
- Instantiate variables
- Apply rules
- Skips rules

Summary: Rule application

$$r = \quad F, \chi \rightarrow \exists \underline{n}{:}\underline{\tau}.\ G(\underline{n})$$

- Constraint check

$$\Sigma \models \chi \qquad \text{(constraint handler)}$$

- Firing

$$\underbrace{[S_1]^{R(r,\rho)^A}_{\Sigma}}_{S,\,F} \rightarrow \underbrace{[S_2]^{R\rho^A}_{\Sigma,\,\underline{c}:\underline{\tau}}}_{S,\,G(\underline{c})} \qquad \underline{c} \text{ not in } S_1$$

Configurations

$C = [S]^{R}_{\Sigma}$

Active role set

State

- $N(t)$
- $L_l(t, \ldots, t)$
- $M_A(t, \ldots, t)$

Signatur e

- $a : \tau$
- $L_l : \underline{\tau}$
- $M_{\_} : \underline{\tau}$

Execution Model

$$P \triangleright C \rightarrow C'$$

- Activate roles
- Generates new role state pred. names
- Instantiate variables
- Apply rules
- Skips rules

Variable Instantiation

$$\frac{\Sigma \vdash t : \tau}{[S]^{R}\ (\forall x{:}\tau.r,\rho)^{A}_{\Sigma} \rightarrow [S]^{R}\ ([t/x]r,\rho)^{A}_{\Sigma}}$$

- Not fully realistic for verification
  - Redundancy realizes typing, …
  - … but not completely

Rule Application

$$r = \quad F, \chi \rightarrow \exists \underline{n}{:}\underline{\tau}.\ G(\underline{n})$$

- Constraint check

$$\Sigma \models \chi \qquad \text{(constraint handler)}$$

- Firing

$$\underbrace{[S_1]^{R(r,\rho)^A}_{\Sigma}}_{S,\ F} \rightarrow \underbrace{[S_2]^{R\rho^A}_{\Sigma,\ \underline{c}:\underline{\tau}}}_{S,\ G(\underline{c})} \qquad \underline{c} \text{ not in } S_1$$

Properties

- Admissibility of parallel firing
- Type preservation
- Access control preservation
- Completeness of Dolev-Yao intruder

MSR 2.0 – NS Initiator

$A \rightarrow B: \{n_A, A\}_{kB}$

$B \rightarrow A: \{n_A, n_B\}_{kA}$

$A \rightarrow B: \{n_B\}_{kB}$

$\forall A$

$\exists L$: princ x princ$^{(B)}$ x pubK B x nonce.

$\forall B$: princ
$\forall k_B$: pubK B

•

$\rightarrow$

$\exists n_A$:nonce.

$L(A,B,k_B,n_A)$
$N(\{n_A,A\}_{kB})$

$\forall$ ...
$\forall k_A$: pubK A
$\forall k'_A$: privK k_A
$\forall n_A, n_B$: nonce

$L(A,B,k_B,n_A)$
$N(\{n_A,n_B\}_{kA})$

$\rightarrow$

$N(\{n_B\}_{kB})$

MSR 2.0 – NS Responder

$A \rightarrow B: \{n_A, A\}_{kB}$
$B \rightarrow A: \{n_A, n_B\}_{kA}$
$A \rightarrow B: \{n_B\}_{kB}$

$\forall B$

$\exists L$: princ$^{(B)}$ x pubK $B^{(kB)}$ x privK k_B x nonce.

$\forall k_B$: pubK B
$\forall k'_B$: privK k_B
$\forall A$: princ
$\forall n_A$: nonce
$\forall k_A$: pubK A

$N(\{n_A, A\}_{kB}) \rightarrow \exists n_B$:nonce. $L(B, k_B, k'_B, n_B)$ $N(\{n_A, n_B\}_{kA})$

$\forall$ ...
$\forall n_B$: nonce

$L(B, k_B, k'_B, n_B)$ $N(\{n_B\}_{kB}) \rightarrow \bullet$

Readings

- R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, 1978
- M. Burrows, M. Abadi, and R. Needham, *A Logic of Authentication*, 1989
- M. Abadi and A. Gordon, *A Calculus for Cryptographic Protocols: The Spi-Calculus*, 1999
- J. Thayer-Fabrega, J. Herzog, and J. Guttman, *Strand Spaces: Why is a Security Protocol Correct*, 1998
- Iliano Cervesato, *Typed MSR: Syntax and Examples*, 2000

Exercises for Lecture 7

- Give MSR 2.0 encodings of one of the following protocols from the Clark and Jacob library (`http://www-users.cs.york.ac.uk/~jac/drareview.ps.gz`)
  - Needham-Schroeder public key [6.3.1]
  - Amended Needham-Schroeder [6.3.4]
  - Wide-Mouthed Frog [6.3.5]
  - Yahalom[6.3.6]
  - Woo-Lam [6.3.10 Π]
  - Kehne-Langendorfer-Shoenwalder [6.3.5]
  - Kao-Chow [6.5.4]

Next ...

- Intruder Models