MSR by Examples

Iliano Cervesato iliano@itd.nrl.navy.mil

ITT Industries, Inc @ NRL - Washington DC

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

Outline

I. Security Protocols

II. MSR by Examples

- Multiset rewriting
- The Neuman-Stubblebine Protocol
- MSR

III. Intruder Models

- Dolev-Yao intruder

Part I

Security Protocols

What are they?

Exchange of (encrypted) messages for

- Communicating secrets
- Authentication
- Contract signing
- E-commerce
- …

Neuman-Stubblebine – Phase I

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

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

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

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

Neuman-Stubblebine – Phase II

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

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

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

Dolev-Yao Abstraction

- Symbolic data
  - No bit-strings
- Perfect cryptography
  - No guessing of keys
- Public knowledge soup
  - Magic access to data

Why is Protocol Analysis Difficult?

- Subtle cryptographic primitives
  - Dolev-Yao abstraction
- Distributed hostile environment
  - "Prudent engineering practice"
- Inadequate specification languages
  - *… the devil is in details …*

Languages to Specify What?

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

Desirable Properties

- Unambiguous
- Simple
- Flexible
  - Adapts to protocol
  - Applies to a wide class of protocols
- Insightful

Part II

MSR by Examples

What's in MSR ?

- Multiset rewriting with existentials
- Dependent types w/ subsorting 

- Memory predicates 

- Constraints 

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

Neuman-Stubblebine – Phase I

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

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

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

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

NS-I: B's 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 point of view

$A \rightarrow B$: A, n_A

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

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

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

NS-I: *A*'s point of view

$A \rightarrow B$: A, n_A

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

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

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

Ticket

Sending / Receiving Messages

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

Network predicate

N(t): t is a message in transit

$N(\{B, n_A, k_{AB}, T_B\}_{kAS}, X, n_B) \rightarrow N(\{n_B\}_{kAB})$

Terms

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

Definable

Nonces

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

Existential variables

n_A instantiated to a new constant

$$N(\{B, n_A, k_{AB}, T_B\}_{kAS}, X, n_B) \quad \rightarrow \quad N(\{n_B\}_{kAB})$$

MSet Rewriting with Existentials

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

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

- Application

$$M_1 \xrightarrow{r} M_2$$

$$M', F(\underline{\mathbf{t}}) \xrightarrow{r} M', G(\underline{\mathbf{t}},\underline{\mathbf{c}})$$

$\underline{\mathbf{c}}$ not in M_1

Sequencing actions

$\exists L.$

Fresh!

$\bullet \rightarrow \exists n_A.$

$N(A, n_A)$
$L(A, n_A)$

Role state predicate

$L(A, n_A)$
$N(\{B, n_A, k_{AB}, T_B\}_{kAS}, X, n_B)$ $\rightarrow$ $N(\{n_B\}_{kAB})$

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

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

$$L(A, n_A),\ N(\{B, n_A, k_{AB}, T_B\}_{kAS}, X, n_B) \rightarrow N(\{n_B\}_{kAB}),\ Tkt_A(B, k_{AB}, X)$$

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

New

Role owner

The principal executing the role

$\forall A$

$\exists L.$

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

$L(A, n_A)$, $N(\{B, n_A, k_{AB}, T_B\}_{kAS}, X, n_B) \rightarrow N(\{n_B\}_{kAB})$, $Tkt_A(B, k_{AB}, X)$

What is what?

Types

$\forall A$

$\exists L$: princ x nonce.

$\bullet \rightarrow \exists n_A$:nonce.

$L(A,n_A)$
$N(A, n_A)$

$\forall$ B:princ.
$\forall$ n_A,n_B: nonce
$\forall$ k_{AB}: shK A B
$\forall$ k_{AS}: shK A S
$\forall$ X: msg

$L(A,n_A)$
$N(\{B,n_A,k_{AB},T_B\}_{kAS}, X, n_B)$

$\rightarrow$

$N(\{n_B\}_{kAB})$
$Tkt_A(B,k_{AB},X)$

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

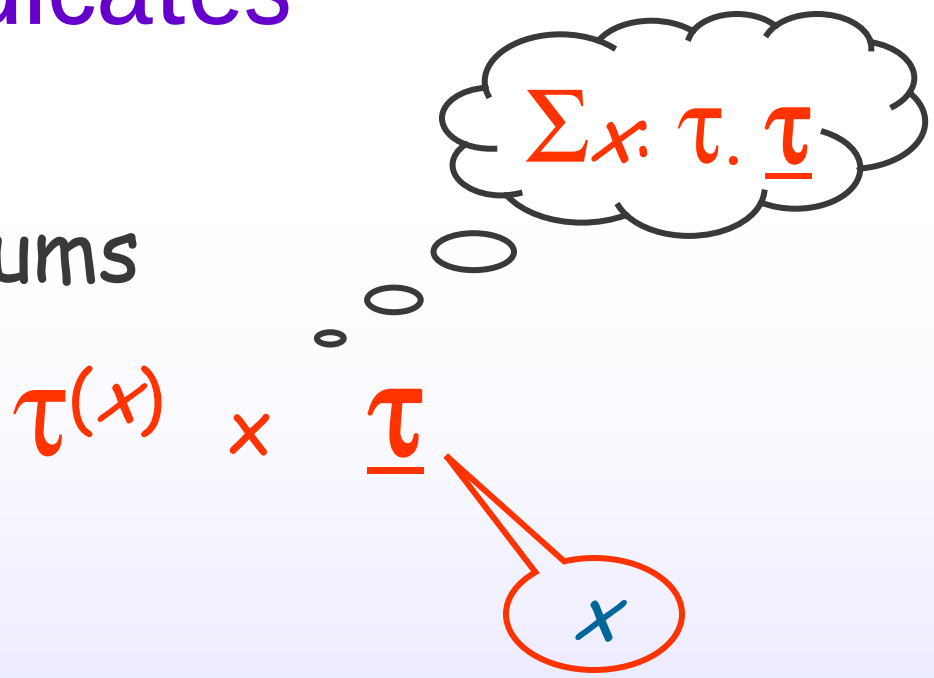
Subtyping

τ :: msg

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

Type of predicates

- Dependent sums



- Forces associations among arguments
  - E.g.: $\text{princ}^{(A)}$ 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

Access Control

$\Sigma \Vdash P$

r is AC-valid for A in Γ

$\Gamma \Vdash_A r$

P is AC-valid in Σ

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

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

$A \rightarrow B$: A, n_A

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

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

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

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

- N(t) Network
- $\mathcal{L}$(t, …, t) Local state
- M_A(t, …, t) Memory
- χ Constraints

$\rightarrow$

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

rhs

- N(t) Network
- $\mathcal{L}$(t, …, t) Local state
- M_A(t, …, t) Memory

Summary: Roles

Role state pred. var. declarations

- Generic roles

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

Role owner

- Anchored roles

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

Summary: Snapshots

Active role set

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

State

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

Signature

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

Summary: Execution Model

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

- 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

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

Properties

- Type preservation
- Access control preservation
- Completeness of Dolev-Yao intruder

Completed Case-Studies

- Full Needham-Schroeder public-key
- Otway-Rees
- Neuman-Stubblebine repeated auth.
- OFT group key management

Part III

The Intruder

Execution with an Attacker

$$P,\ P_I \triangleright C \rightarrow C'$$

- Selected principal(s): I
- Generic capabilities: P_I
  - Well-typed
  - AC-valid
- Modeled completely within MSR

The Dolev-Yao Intruder

- Specific protocol suite P_{DY}
- Underlies every protocol analysis tool
- Completeness still unproved

Capabilities of the D-Y Intruder

- Intercept / emit messages
- Decrypt / encrypt with known key
- Split / form pairs
- Look up public information
- Generate fresh data

DY Intruder – Net Interference

- $M_I(t)$: Intruder knowledge

$$\left[\forall t: msg.\ N(t) \rightarrow\ M_I(t)\right]^I$$

$$\left[\forall t: msg.\ M_I(t) \rightarrow N(t)\right]^I$$

DY Intruder – Decryption

$$\left[\begin{array}{l}\forall A,B\text{: princ}\\ \forall k\text{: shK } A\ B\\ \forall t\text{: msg}\end{array}\quad \begin{array}{l}M_I(\{t\}_k)\\ M_I(k)\end{array}\quad \rightarrow\ M_I(t)\right]^I$$

$$\left[\begin{array}{l}\forall A\text{: princ}\\ \forall k\text{: pubK } A\\ \forall k'\text{: privK } A\\ \forall t\text{: msg}\end{array}\quad \begin{array}{l}M_I(\{t\}_k)\\ M_I(k)\end{array}\quad \rightarrow\ M_I(t)\right]^I$$

DY Intruder – Encryption

$$\left[\begin{array}{l}\forall A,B\text{: princ}\\ \forall k\text{: shK } A\ B\\ \forall t\text{: msg}\end{array}\quad \begin{array}{l}M_I(t)\\ M_I(k)\end{array} \rightarrow M_I(\{t\}_k)\right]^I$$

$$\left[\begin{array}{l}\forall A\text{: princ}\\ \forall k\text{: pubK } A\\ \forall t\text{: msg}\end{array}\quad \begin{array}{l}M_I(t)\\ M_I(k)\end{array} \rightarrow M_I(\{t\}_k)\right]^I$$

DY Intruder – Pairs

$$\left[\ \forall t_1,t_2\colon msg \qquad M_I(t_1,t_2) \rightarrow \begin{matrix} M_I(t_1) \\ M_I(t_2) \end{matrix}\ \right]^I$$

$$\left[\ \forall t_1,t_2\colon msg \qquad \begin{matrix} M_I(t_1) \\ M_I(t_2) \end{matrix} \rightarrow M_I(t_1,t_2)\ \right]^I$$

DY Intruder – Structural rules

$$\left[\ \forall t\colon msg \qquad M_I(t) \rightarrow \begin{matrix} M_I(t) \\ M_I(t) \end{matrix}\ \right]^I$$

$$\left[\ \forall t\colon msg \qquad M_I(t) \rightarrow \bullet\ \right]^I$$

DY Intruder – Data access

$$\left[\forall A\text{: princ.} \bullet \rightarrow M_I(A)\right]^I$$

$$\left[\begin{array}{l}\forall A\text{: princ} \\ \forall k\text{: shK I A}\end{array} \bullet \rightarrow M_I(k)\right]^I \quad \textit{+ dual}$$

$$\left[\begin{array}{l}\forall A\text{: princ} \\ \forall k\text{: pubK A}\end{array} \bullet \rightarrow M_I(k)\right]^I \left[\begin{array}{l}\forall k\text{: pubK I} \\ \forall k'\text{: privK k}\end{array} \bullet \rightarrow M_I(k')\right]^I$$

- No nonces, no other keys, …

DY Intruder – Data Generation

- Safe data

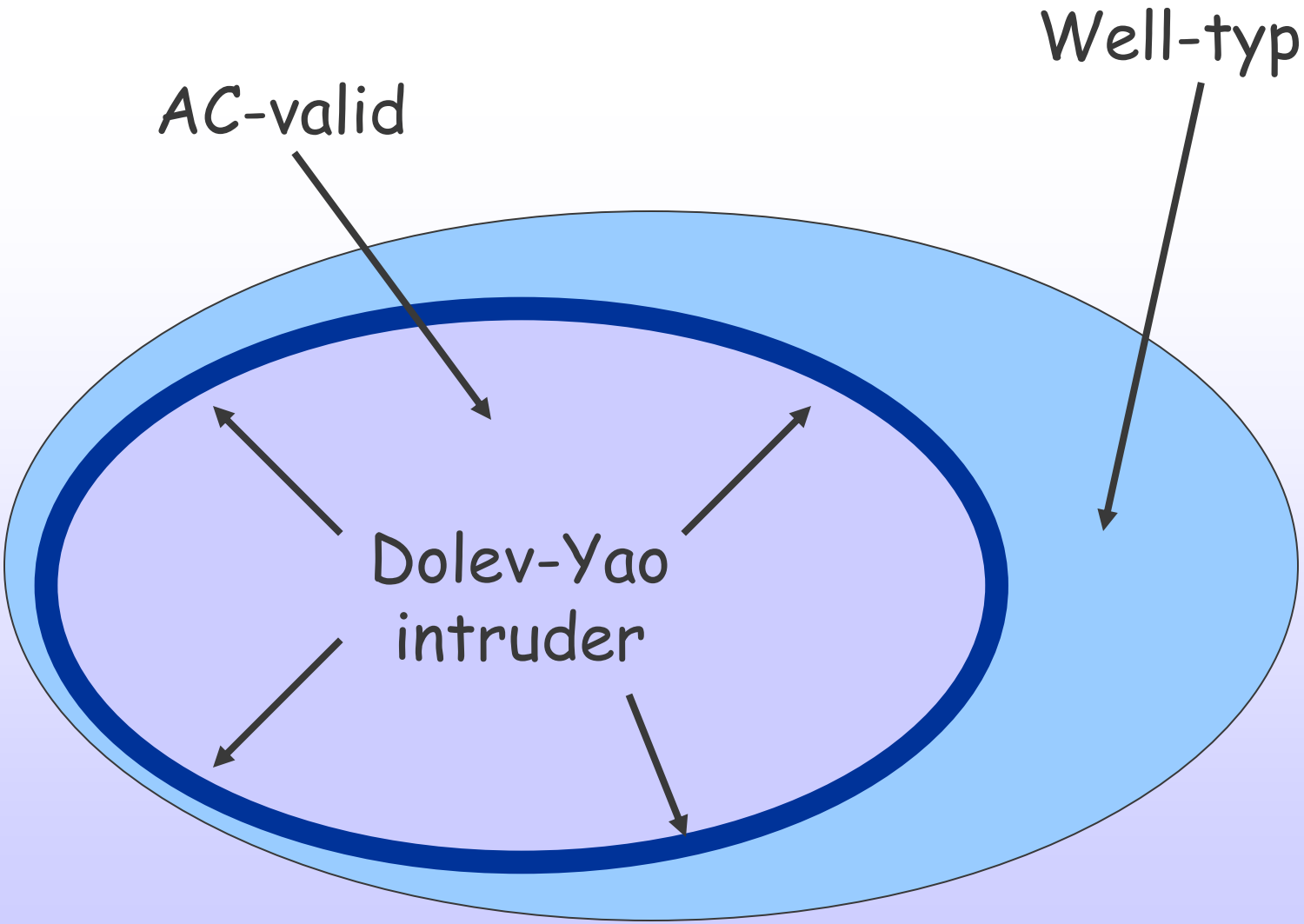
$$\left[\bullet \rightarrow \exists n{:}nonce.\ M_I(n)\right]^I \qquad \left[\bullet \rightarrow \exists m{:}msg.\ M_I(m)\right]^I$$

- Anything else ?

$$\left[\forall A,B{:}princ.\ \bullet \rightarrow \exists k{:}shK\ A\ B.\ M_I(k)\right]^I \qquad ???$$

- It depends on the protocol !!!
  - Automated generation ?

DY Intruder Stretches AC to Limit



Completeness of D-Y Intruder

- If $\mathbf{P} \triangleright [S]^{R}_{\Sigma} \rightarrow [S']^{R'}_{\Sigma'}$

  with all well-typed and *AC*-valid

- Then

$$\underline{\mathbf{P}}, \mathbf{P}_{\mathbf{DY}} \triangleright [\underline{S}]^{R}_{\underline{\Sigma}} \rightarrow [\underline{S'}]^{R'}_{\underline{\Sigma'}}$$

Consequences

- Justifies design of current tools
- Support optimizations
  - D-Y intr. often too general/inefficient
    - Generic optimizations
    - *Per protocol* optimizations
    - Restrictive environments
- Caps multi-intruder situations

Future work

- Experimentation
  - Clark-Jacob library
  - Fair-exchange protocols
  - More multicast
- Pragmatics
  - Type-reconstruction
  - Operational execution model(s)
  - Implementation
- Automated specification techniques