

Relating Multiset Rewriting and Process Algebra for Security Protocol Specification



Iliano Cervesato

iliano@itd.nrl.navy.mil

ITT Industries, inc @ NRL Washington, DC

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

Joint work with S. Bistarelli, G. Lenzini, and F. Martinelli



Objective

- Relate specification languages for security protocols
 - MSR $\leftrightarrow$ strands [CSFW'00]
 - MSR $\leftrightarrow$ linear logic [MFPS'00]
 - **MSR $\leftrightarrow$ Process Algebras**

Non-Objective (for now)

- Reachability analysis $\leftrightarrow$ bisimulation
 - Verification methodologies not considered

Why MSR?

- Model of specification underlies numerous languages and tools
 - CIL/CAPSL
 - NRL Protocol Analyzer
 - Paulson's Isabelle specifications
 - Mur ϕ
 - ...
- Simple and well-understood foundations
 - Distributed systems
 - Petri nets
 - Linear logic
 - Rewriting theory



Multiset Rewriting + Existentials

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

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

- Application

$$\begin{array}{ccc} M_1 & \xrightarrow{r} & M_2 \\ \underbrace{} & & \underbrace{} \\ M', F(\mathbf{t}) & \xrightarrow{r} & M', G(\mathbf{t}, \mathbf{c}) \\ & & \mathbf{c} \text{ not in } M_1 \end{array}$$

- This is **MSR 1.0**

MSR 2.0:

- + strong typing
- + constraints
- + domain-specific enhancements

Which Process Algebra?

“PA”

- Inspired to
 - CCS
 - π -calculus
- Only primitives used for protocols
- As a programming language for protocols
 - Reachability
 - Not simulation/equivalence





“PA”

- Sequential processes

$$P ::= 0 \mid a(x).P \mid \underline{a}(t).P \mid [x=t]P \mid \nu x.P$$

- Parallel processes

$$Q ::= 0 \mid P \parallel Q \mid !P \parallel Q$$

- $(P, \parallel, 0)$ monoid

➤ Equivalence $\equiv$

- Reaction

$$Q \parallel \underline{a}(t).P \parallel a(x).P' \rightarrow Q \parallel P \parallel [t/x]P'$$

MSR $\Leftrightarrow$ PA ... in General

- Very different paradigms

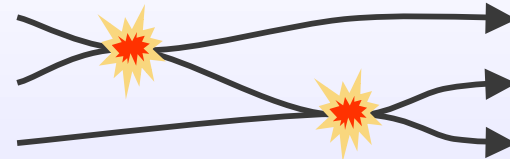
- MSR

- state transition



- PA

- contact evolution



- Non trivial

- MSR $\rightarrow$ PA: *granularity of actions*
 - PA $\rightarrow$ MSR: *excise state*

- Reachability-preserving

- Non bijective





MSR $\Leftrightarrow$ PA ... for Protocols

Much simpler!

- Take natural specifications
 - in MSR
 - in PA
- Bijective correspondence
 - (to a large extent)



MSR for Security Protocols

- Fixed predicates
 - $N(m)$ Network messages
 - $I(m)$ Intruder info.
 - $A_i(t_1, \dots, t_{ni})$ Role states
 - $Pr, PrvK, PubK, \dots$ Persistent info.
- Fixed format
 - Protocol given as set of roles
 - Dolev-Yao intruder spec.
- (more freedom in MSR 2.0)



Roles in MSR

- One instantiation rule

$$\pi(\mathbf{x}) \rightarrow \exists \mathbf{n}. A_0(\mathbf{x}, \mathbf{n}), \pi(\mathbf{x})$$

- Several execution rules

➤ Send

$$A_i(\mathbf{z}) \rightarrow A_{i+1}(\mathbf{z}), N(t)$$

➤ Receive

$$A_i(\mathbf{t}), N(t') \rightarrow A_{i+1}(\mathbf{z}_t, \mathbf{x}_{t'})$$

NSPK (initiator) in MSR

$$\pi_A(A,B) \rightarrow A_0(A,B), \pi_A(A,B)$$

$$A_0(A,B) \rightarrow \exists N_A. A_1(A,B,N_A), N(\{N_A,A\}_{KB})$$

$$A_1(A,B,N_A), N(\{N_A,N_B\}_{KA}) \rightarrow A_2(A,B,N_A,N_B)$$

$$A_2(A,B,N_A,N_B) \rightarrow A_3(A,B,N_A,N_B), N(\{N_B\}_{KB})$$

where $\pi_A(A,B) = Pr(A), PrvK(A,K_A^{-1}),$
 $Pr(B), PubK(B,K_B)$



MSR Configurations

- Rules

- U_{ρ}

Protocol roles

- ρ_I

Intruder role

- State

- $N(t)$

Network messages

- $A_i(t)$

Role state predicates

- $\pi(t)$

Persistent knowledge

- $I(t)$

Intruder knowledge



Security Protocols in PA

- Fixed set of name

➤ N_i, N_o, π, I

- Fixed structure of “Security Process”

{

Q_i

➤ $Q_{!net} = ! N_i(x). \underline{N}_o(x). 0$

➤ $Q_{!p} = ||_p P_p$

▪ $! \pi(\mathbf{x}). \mathbf{vn}. P'$

• input on N_o

• output on $\underline{N}_i$

• pattern matching

➤ $Q_{!I}$

➤ $Q_{!\pi}$

➤ $Q_{!0}$

Network process

Roles

Dolev-Yao Intruder

Persistent information

Initial intruder knowledge

NSPK (initiator) in PA

$\pi_A(A, B). \textcolor{red}{v}N_A$

$\underline{N}_i(\{N_A, A\}_{KB}) .$

$N_o(x). \textcolor{violet}{[}x = \{N_A, N_B\}_{KA}\textcolor{violet}{]}$

$\underline{N}_i(\{N_B\}_{KB}) .$

0



Process State

- Q_i Replicated process
- Q Unreplicated part
 - Q_i Intruder knowledge
 - Q_{net} Buffered network messages
 - Q_p Roles in mid-execution



MSR into PA

• Rules

- $U_p \rightarrow Q_{!p} + Q_{!net}$
 - Instantiation rule $\rightarrow$ “ $! \pi(\mathbf{x}). \mathbf{vn}.$ ” prefix
 - “ $A_i(\mathbf{z}) \rightarrow A_{i+1}(\mathbf{z}), N(t)$ ” $\rightarrow$ $\underline{N}_i(t). \langle r_{i+1} \rangle$
 - “ $A_i(\mathbf{t}), N(t') \rightarrow A_{i+1}(\mathbf{z}_t, \mathbf{x}_{t'})$ ” $\rightarrow$ $N_o(x). [x=t'] [\mathbf{z}=\mathbf{t}] \langle r_{i+1} \rangle$
- $\rho_l \rightarrow Q_{!l}$

• State

- $N(t) \rightarrow Q_{net}$
- $A_i(\mathbf{t}) \rightarrow Q_p$
- $\pi(\mathbf{t}) \rightarrow Q_{! \pi}$
- $l(t) \rightarrow Q_l$

$NSPK_{MSR} \rightarrow NSPK_{PA}$
 (once reducing variable renamings)
 $[x=x']$



PA into MSR

Essentially the inverse transformation

- $Q_{!p} \rightarrow U_p$
 - Invent A_i 's
 - Carry over substitutions
- $Q_{!!} \rightarrow \rho_I$

$$\text{NSPK}_{PA} \rightarrow \text{NSPK}_{MSR}$$

(for α -convertible A_i 's)

The Intruder

1-1 correspondence, but ...

- $I(\langle x_1, x_2 \rangle) \rightarrow I(x_1), I(x_2)$

- $I(\langle x_1, x_2 \rangle). \perp(x_1). 0$
 $I(\langle x_1, x_2 \rangle). \perp(x_2). 0$

- $I(x) \rightarrow I(x), I(x)$

- $I(x). \perp(x). \perp(x). 0$

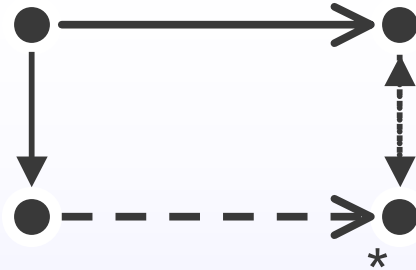
- $I(x_1), I(x_2) \rightarrow I(\langle x_1, x_2 \rangle)$

- $I(x_1). I(x_2). \perp(\langle x_1, x_2 \rangle). 0$

Correspondence

MSR

PA



- Proof technique: weak bi-simulation
 - Observables
 - Network messages
 - Intruder knowledge

Conclusions

- Formal relation between MSR and PA
 - As used for security protocols
 - Non trivial (yet mostly bijective)
 - Technique similar to MSR $\leftrightarrow$ strands

... And future work

- MSR 3.0
- Strict comparison with spi-calculus
- Relating methodologies

