

Towards an Execution Environment
for the
MSR Cryptoprotocol Specification Language

Mark-Oliver Stehr
University of Illinois at Urbana-Champaign

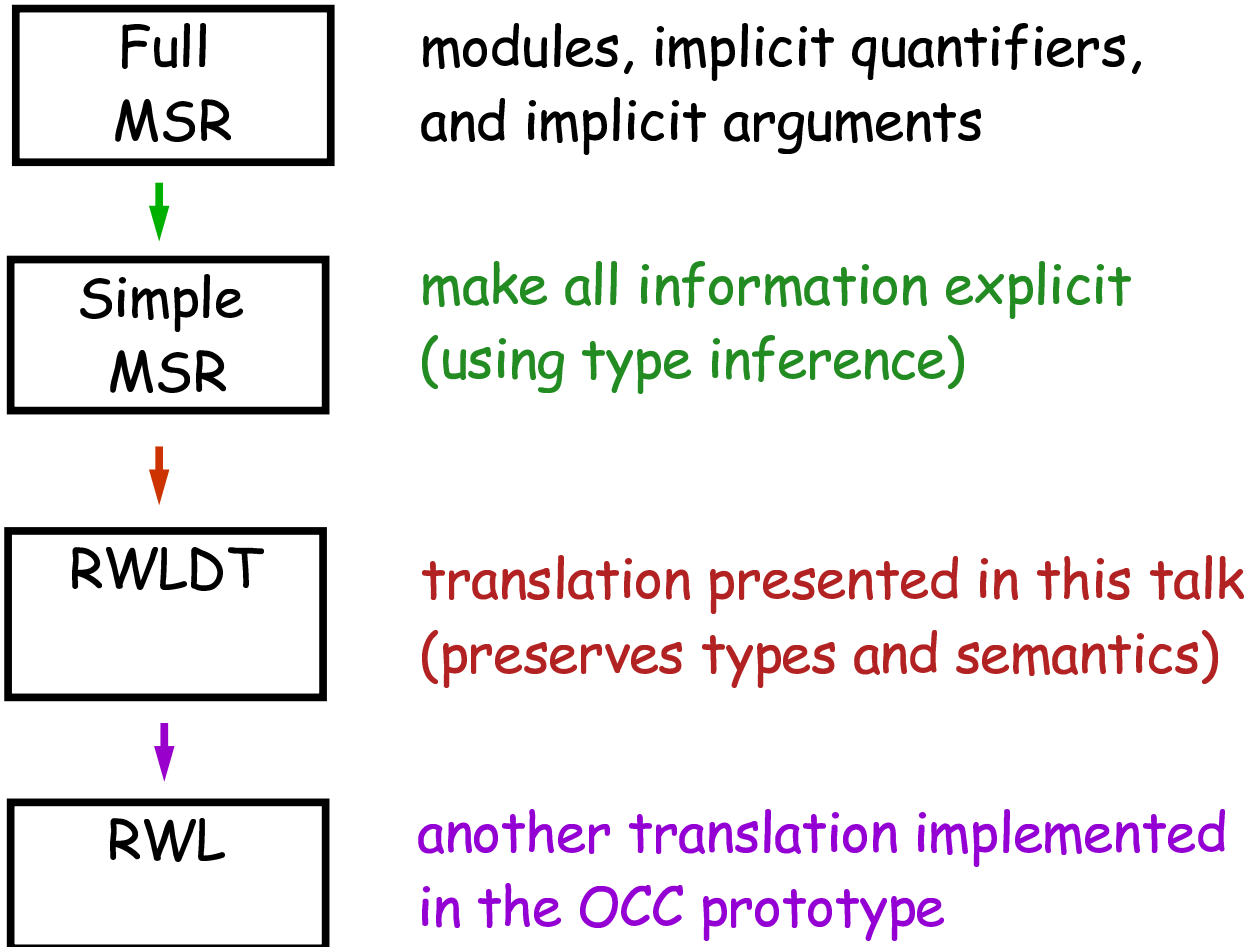
joint work with

Iliano Cervesato
ITT Industries, Inc.

Overview

- Introduction
- Architecture of the Tool
- Multiset Rewriting (MSR) Specification Language
- Rewriting Logic with Dependent Types (RWLDT)
- Representing Types and Protocol State of MSR
- Transforming MSR Rules into First-Order Rewrite Rules
- Running Example
- Conclusion

Architecture of the Tool



MSR Specification Language

- Atomic Types:
 - princ, nonce, msg
- Dependent Types:
 - shK A B, pubK A, privK k, stK A B, ItK A B
- Subtyping:
 - princ <: msg, nonce <: msg,
 - pubK A <: msg, privK k <: msg, stK A B <: msg
 - stK AB, <: shK A B, ItK A B <: shK A B
- Data Access Specification

MSR Specification Language

- State is Multiset of Message Predicates:
 - $N(t)$ represents info on the network
 - $L(t_1, \dots, t_n)$ represents local state of a role
 - $M_A(t_1, \dots, t_n)$ represents memory of principal A
- Generic Roles:
 - $\forall A . \exists L : \tau_1, \dots, \tau_n . \exists \dots$
 - $\forall x : \sigma . LHS \Rightarrow \exists n : \rho . RHS,$
 - $\forall \dots . \dots \Rightarrow \exists \dots . \dots$
 - arbitrary number of $\exists L : \tau_1, \dots, \tau_n$ and rules per role
 - arbitrary number of $\forall x : \sigma$ and $\exists n : \rho$ per rule
- Anchored Roles:
 - A is fixed instead of quantified

MSR Specification Language

$\forall A . \exists L : \tau_1, \dots, \tau_n .$

$\forall x : \sigma . \text{LHS} \Rightarrow \exists n : \rho . \text{RHS},$

$\forall \dots . \dots \Rightarrow \exists \dots . \dots$

Operational Semantics:

- a. instantiate A with arbitrary principal (if not anchored)
- b. instantiate L with fresh symbol
- c. add resulting role instance to active role set, and independently
 1. select role instance from active role set
 2. select an applicable rule from this role instance
 3. instantiate x with arbitrary term of type σ
 4. instantiate n with fresh symbol
 5. apply rule instance to current state
 6. remove the rule from active role set

} can be
skipped

Example

Otway-Rees Authentication Protocol:

1. $A \rightarrow B : n \ A \ B \ \{n_A \ n \ A \ B\}_{k_{AS}}$
2. $B \rightarrow S : n \ A \ B \ \{n_A \ n \ A \ B\}_{k_{AS}} \ \{n_B \ n \ A \ B\}_{k_{BS}}$
3. $S \rightarrow B : n \ \{n_A \ k_{AB}\}_{k_{AS}} \ \{n_B \ k_{AB}\}_{k_{BS}}$
4. $B \rightarrow A : n \ \{n_A \ k_{AB}\}_{k_{AS}}$

Note:

- A, B range over arbitrary principals
- S is a fixed principal, the key server
- k_{AS}, k_{BS} are long term keys
- k_{AB} is the new short term key

Specification in MSR

Generic Role B

$\forall B . \exists L : \text{princ}^{(B)} \times \text{nonce} \times \text{nonce} \times \text{ItK } B \ S .$

$\forall A : \text{princ} . \forall n : \text{nonce} . \forall k_{BS} : \text{ItK } B \ S . \forall X : \text{msg} .$

$N(n \ A \ B \ X) \Rightarrow$

$\exists n_B : \text{nonce} . N(n \ A \ B \ X \ \{n_B \ n \ A \ B\}_{k_{BS}})$

$L(B, A, n, n_B, k_{BS}),$

$\forall A : \text{princ} . \forall n, n_A : \text{nonce} .$

$\forall k_{BS} : \text{ItK } B \ S . \forall k_{AB} : \text{stK } A \ B . \forall Y : \text{msg} .$

$N(n \ Y \ \{n_B \ k_{AB}\}_{k_{BS}})$

$L(B, A, n, n_B, k_{BS}) \Rightarrow$

$N(n \ Y)$

RWL with Dependent Types

Combination of two key concepts:

- Conditional rewriting modulo equations:
 - $(\forall x : S) A = B \text{ if } C$ (generalizes equational logic)
 - $(\forall x : S) A \Rightarrow B \text{ if } C$ (generalizes rewriting logic)
- Lambda-Calculus with dependent function types
 - $[x : S] M : \{x : S\} T$ (generalizes ordinary type theory)

Propositions-as-Types Interpretation:

$(\forall x : S) P(x)$ is interpreted as $\{x : S\} P(x)$
 \Rightarrow provides an expressive higher-order logic

RWL with Dependent Types

- Operational Semantics:
 - open computation system
 - conditional rewriting modulo equations
 - automatic proof search for conditions
 - uniform for execution and type checking
 - equational/rewriting logic executable sublanguage
- Model-theoretic Semantics:
 - classical, set-theoretic
- Current Implementation:
 - prototype in rewriting logic (on top of Maude)
 - mapping of higher-order to first-order concepts

Example

$fms : Type \rightarrow Type .$

$empty : \{T : Type\} (fms T) .$

$single : \{T : Type\} T \rightarrow (fms T) .$

$union : \{T : Type\} (fms T) \rightarrow (fms T) \rightarrow (fms T) .$

$comm : || \{T : Type\} \{m1, m2 : (fms T)\}$
 $(union\ m1\ m2) = (union\ m2\ m1) .$

$assoc : || \{T : Type\} \{m1, m2, m3 : (fms T)\}$
 $(union\ m1\ (union\ m2\ m3)) = (union\ (union\ m1\ m2)\ m3) .$

$id : || \{T : Type\} \{m : (fms T)\}$
 $(union\ m\ empty) = m .$

Example

`select` : {T | Type} (T -> Prop) -> (fms T) -> (fms T) .

`select_empty` : !! {T : Type}{P : (T -> Prop)}
(`select` P `empty`) = `empty` .

`select_single_1` : !! {T : Type}{P : (T -> Prop)}{x : T}
(P x) -> (`select` P (`single` x)) = (`single` x) .

`select_single_2` : !! {T : Type}{P : (T -> Prop)}{x : T}
(Not (P x)) -> (`select` P (`single` x)) = `empty` .

`select_union` : !! {T : Type}{m,m' : (fms T)}{P : (T -> Prop)}
(Not (empty? m)) -> (Not (empty? m')) ->
(`select` P (`union` m m')) = (`union` (`select` P m) (`select` P m')) .

Representing MSR Types

`princ` : Type .

`nonce` : Type .

`pubK` : `princ` -> Type .

`privK` : {`A` : `princ`} (`pubK` `A`) -> Type .

`shK` : `princ` -> `princ` -> Type .

`ltK` : `princ` -> `princ` -> Type .

`stK` : `princ` -> `princ` -> Type .

`ltK-shK` : {`A,B` : `princ`} (`ltK` `A` `B`) -> (`shK` `A` `B`) .
`stK-shK` : {`A,B` : `princ`} (`stK` `A` `B`) -> (`shK` `A` `B`) .

} Coercions

Representing MSR Types

msg : Type .

nonce-msg : nonce -> msg .

princ-msg : princ -> msg .

stkey-msg : {A, B : princ} (stK A B) -> msg .

} Coercions

nil : msg .

append : msg -> msg -> msg .

encrypt : {A,B : princ} msg -> (shK A B) -> msg .

append_assoc : || {m1,m2,m3 : msg}

(append m1 (append m2 m3)) = (append (append m1 m2) m3) .

append_id : || {m : msg}

(append m nil) = m .

Representing MSR State

state : Type .

empty : state .

union : state -> state -> state .

union_comm : || {s1,s2 : state}

(union s1 s2) = (union s2 s1) .

union_assoc : || {s1,s2,s3 : state}

(union s1 (union s2 s3)) = (union (union s1 s2) s3) .

union_id : || {s : state}

(union s empty) = s .

N : msg -> state .

Representing MSR Roles

$\forall \exists$ over rules

$\forall A . \exists L . LHS \Rightarrow RHS$ ($A : \text{princ}$ implicit)

becomes

$P(A) F(L) LHS \Rightarrow P(A) F(\text{succ}(L)) RHS$

Exception:

- $P(A)$ not needed if LHS contains A

Finiteness Requirement:

- princ must be finite type, and
- all its elements are enumerated as $P(A)$ in the state

Representing MSR Rules

\exists on the right-hand side

$$\forall x : \sigma . LHS \Rightarrow \exists n . RHS$$

becomes

$$\sigma(x) F'(n) LHS \Rightarrow \sigma(x) F'(\text{suc}(n)) RHS$$

Exception:

- $\sigma(x)$ not needed if LHS contains x

Finiteness Requirement:

- σ must be finite type, and
- all its elements are enumerated as $\sigma(x)$ in the state

Representing Multiple Rules

$$\text{LHS}_1 \Rightarrow \text{RHS}_1,$$

$$\text{LHS}_2 \Rightarrow \text{RHS}_2$$

becomes

$$\Rightarrow T_1 T_2$$

$$T_1 \text{LHS}_1 \Rightarrow \text{RHS}_1$$

$$T_2 \text{LHS}_2 \Rightarrow \text{RHS}_2$$

becomes

$$\text{LHS}_1 \Rightarrow \text{RHS}_1 T_2$$

$$T_1 \text{LHS}_1 \Rightarrow \text{RHS}_1$$

$$\text{LHS}_2 \Rightarrow \text{RHS}_2 T_1$$

$$T_2 \text{LHS}_2 \Rightarrow \text{RHS}_2$$

Motivation for 2nd step: elimination of intermediate states

Representing Multiple Rules

$$\forall A . \exists L . \text{LHS}_1 \Rightarrow \text{RHS}_1, \\ \text{LHS}_2 \Rightarrow \text{RHS}_2$$

becomes

$$P(A) F(L) \Rightarrow P(A) F(\text{succ}(L)) T(A, L)$$

$$T(A, L) \Rightarrow T_1(A, L) T_2(A, L)$$

$$T_1(A, L) \text{LHS}_1 \Rightarrow \text{RHS}_1$$

$$T_2(A, L) \text{LHS}_2 \Rightarrow \text{RHS}_2$$

Typed case and multiple \exists :
similar to \exists on right-hand side

Representing Multiple Rules

$$P(A) F(L) \Rightarrow P(A) F(\text{succ}(L)) T(A,L)$$

$$T(A,L) \Rightarrow T_1(A,L) T_2(A,L)$$

$$T_1(A,L) \text{ LHS}_1 \Rightarrow \text{RHS}_1$$

$$T_2(A,L) \text{ LHS}_2 \Rightarrow \text{RHS}_2$$

becomes

$$P(A) F(L) \text{ LHS}_1 \Rightarrow P(A) F(\text{succ}(L)) \text{ RHS}_1 T_2(A,L)$$

$$T_1(A,L) \text{ LHS}_1 \Rightarrow \text{RHS}_1$$

$$P(A) F(L) \text{ LHS}_2 \Rightarrow P(A) F(\text{succ}(L)) \text{ RHS}_2 T_1(A,L)$$

$$T_2(A,L) \text{ LHS}_2 \Rightarrow \text{RHS}_2$$

Recall MSR Example

Generic Role B

$\forall B . \exists L : \text{princ}^{(B)} \times \text{nonce} \times \text{nonce} \times \text{ItK } B \ S .$

$\forall A : \text{princ} . \forall n : \text{nonce} . \forall k_{BS} : \text{ItK } B \ S . \forall X : \text{msg} .$

$N(n \ A \ B \ X) \Rightarrow$

$\exists n_B : \text{nonce} . N(n \ A \ B \ X \ \{n_B \ n \ A \ B\}_{k_{BS}})$

$L(B, A, n, n_B, k_{BS}),$

$\forall A : \text{princ} . \forall n, n_A : \text{nonce} .$

$\forall k_{BS} : \text{ItK } B \ S . \forall k_{AB} : \text{stK } A \ B . \forall Y : \text{msg} .$

$N(n \ Y \ \{n_B \ k_{AB}\}_{k_{BS}})$

$L(B, A, n, n_B, k_{BS}) \Rightarrow$

$N(n \ Y)$

Translation

R211: !! {B : princ}

{L : {B : princ} princ -> nonce -> nonce -> (ItK B S) -> state}

{A : princ}{kBS : ItK B S}{X : msg}

{fresh, fresh' : nat}{n, nB : nonce}

(L := (LB (suc fresh))) ->

(nB := (nonce fresh)) ->

(fresh' := (suc fresh)) ->

(A211 : (union ((LTK B S kBS), (F fresh),
(N (append (n, A, B, X))))))

=> (union ((LTK B S kBS), (F fresh'),
(N (append (n, A, B, X,
(encrypt (append (nB, n, A, B)) kBS))))),
(L B A n nB kBS),
(TB2 A B L))))

Translation

R222 : !! {B : princ}

{L : {B : princ} princ -> nonce -> nonce -> (ItK B S) -> state}

{A : princ}{kAB : stK A B}{kBS : ItK B S}{Y : msg}{n,nB : nonce}

(A222: (union ((N (append (n, Y, (encrypt (append (nB, kAB)) kBS))))

(L B A n nB kBS)

(TB2 A B L))

=> (N (append (n, Y))))

(optimized and coercions omitted)

Symbolic Execution

To restrict and observe the protocol execution, we add **START** and **TERMINATED** tokens to all roles.

A : princ . B : princ .

rew (union ((F 0), (P A), (P B), (LTK A S kAS), (LTK B S kBS),
(START1 A), (START2 B), (START3 S)) .

trace:

A111 A211 A311 A222 A122

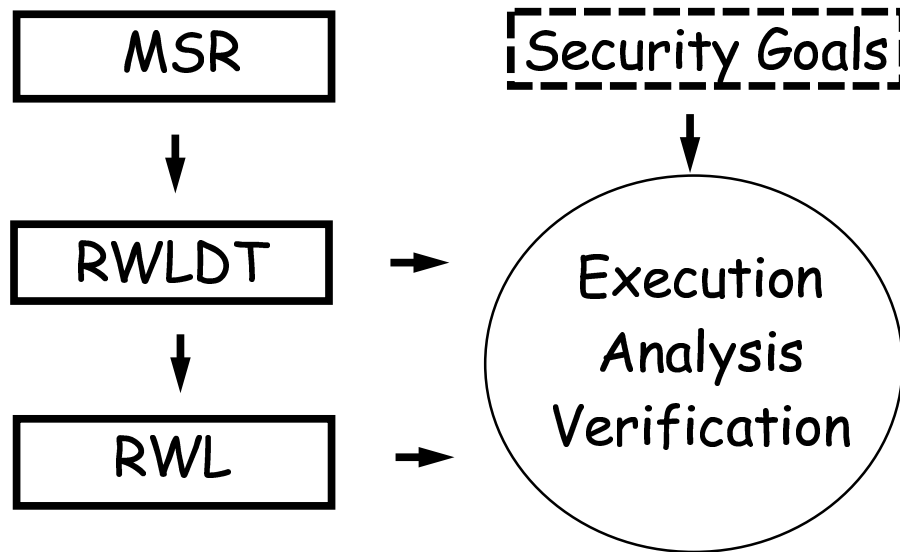
result:

(union ((F 6), (P A), (P B), (LTK A S kAS), (LTK B S kBS),
(TERMINATED1 A), (TERMINATED2 B), (TERMINATED3 C))

Symbolic search shows that this is the only possible execution !

Conclusion

- Embedding into RWL with Dependent Types provides:
 - operational semantics:
basis for symbolic execution and analysis
 - model-theoretic semantics:
basis for formal verification



Conclusion

- Future Work:
 - Implementation of the transformation
 - Specification & execution environment for MSR
 - Taking into account the data access specification
 - Extensions of MSR (constraints, equations,...)
 - Language for Security Goals
- Web Site:
 - URL: <http://formal.cs.uiuc.edu/stehr/msr.html>
 - contains our paper
 - and two fully worked out examples
(Needham-Schroeder and Otway-Rees Protocols)