Title	On-the-fly Model Checking Security Protocols and Its Implementation by Maude
Author(s)	Li, Guoqiang; Ogawa, Mizuhito
Citation	
Issue Date	2006-11-29
Type	Presentation
Text version	publisher
URL	http://hdl.handle.net/10119/8307
Rights	
Description	Theorem Proving and Provers Meeting(2nd TPP)での発表資料，開催：2006年11月29日～30日，開催場所：JAIST 情報科学研究科棟II・Collaboration Room 7 (5F)

On-the-fly Model Checking Security Protocols and Its Implementation by Maude

Guoqiang Li, Mizuhito Ogawa

Japan Advanced Institute of Science and Technology

Nov. 29, 2006

Problems

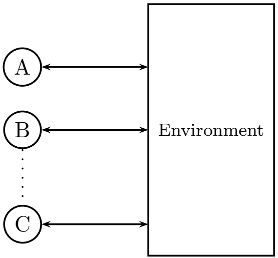
- When model checking security protocols, it suffers from infinite states. Such infinity comes from:
  - Infinitely many sessions of protocols: each principal can initiate or act as a responser infinitely many protocol sessions.
  - Infinitely many principals in the network: each principal may communicate with infinitely many other principals.
  - Infinitely many messages that intruders can generate: each intruder can produce infinitely many messages based on messages leaked in the network(Dolev-Yao).

Our approaches

- A typed process calculus that avoids recursive operations is proposed, so that only finitely many sessions are considered.
- A bound variable is introduced to represent a sender's intended destination, so that the unbounded number of principals are finitely described.
  - $(\nu x : I)\overline{a1}\{M\}_{\mathrm{k}[A,x]}$
- Messages with the same effect in a protocol are unified to a parametric message based on type information.
  - $a1(x).\overline{a2}x$
- Each possible run of a protocol is represented as a trace.

Model a network

- Principals exchange the messages with the environment.
- A message that a receiver receives may not be the same as what a sender sends.
- Environment can produce, modify messages during the communication of principals (represented as a deductive system).



Syntax

$M, N, L ::= \quad n \mid x \mid (M, N) \mid \{M\}_L \mid \mathrm{m}[M_1, \ldots, M_n]$

$P, Q, R ::=$

$\mathbf{0}$	Nil
$\overline{a}M.P$	output
$a(x).P$	input
$[M = N]\,P$	match
$(\nu x : \mathcal{A})P$	range
$let\ (x, y) = M\ in\ P$	pair splitting
$case\ M\ of\ \{x\}_L\ in\ P$	decryption
$P \| Q$	composition

An approximation on sending a message
(Usages of ranges and binders)

- Ranges and binders are used when a principal initiates a protocol, or one can not obtain his communicator's name.
  - $(\nu x : \mathcal{I})\overline{a1}\{A, N_A\}_{+\mathrm{k}[x]} \ldots z \ldots \overline{a3}\{z\}_{+\mathrm{k}[x]}$
  - $(\nu x : \mathcal{I})\overline{a1}\{A, N_A\}_{+\mathrm{k}[x]} \ldots y_b, z \ldots [y_b = x] \ldots \overline{a3}\{z\}_{+\mathrm{k}[y_b]}$
- An approximation is used that the principal sends the same message randomly to different principals.

NSPK protocol

$$\begin{array}{ll} A \longrightarrow B : & \{A, N_A\}_{+K_B} \\ B \longrightarrow A : & \{N_A, N_B\}_{+K_A} \\ A \longrightarrow B : & \{N_B\}_{+K_B} \end{array}$$

Fixed NSPK protocol

$$\begin{array}{ll} A \longrightarrow B : & \{A, N_A\}_{+K_B} \\ B \longrightarrow A : & \{B, N_A, N_B\}_{+K_A} \\ A \longrightarrow B : & \{N_B\}_{+K_B} \end{array}$$

Representation of Abadi-Gordon protocol

(An example of the binder)

$$A \longrightarrow S : \quad A, \{B, K_{AB}\}_{K_{AS}}$$
$$S \longrightarrow B : \quad \{A, K_{AB}\}_{K_{SB}}$$
$$A \longrightarrow B : \quad A, \{A, M\}_{K_{AB}}$$

$$A \triangleq (\nu x : \mathcal{I})\overline{a1}(A, \{x, \mathrm{k}[A, x]\}_{\mathrm{k}[A,S]}).\overline{a2}(A, \{A, M\}_{\mathrm{k}[A,x]}).\mathbf{0}$$

$$B \triangleq b1(x).case\ x\ of \{x'\}_{\mathrm{k}[B,S]}\ in\ let\ (y, z) = x'\ in$$
$$b2(w).let\ (w', w'') = w\ in\ [w' = y]\ case\ w''\ of\ \{u\}_z\ in$$
$$let\ (u', u'') = u\ in\ [u' = y]\ \overline{acc}\ w.\mathbf{0}$$

$$S \triangleq s1(x).let\ (y, z) = x\ in\ case\ z\ of\ \{u\}_{\mathrm{k}[y,S]}\ in\ let\ (u', u'') = u\ in$$
$$\overline{s2}\{y, u''\}_{\mathrm{k}[u',S]}.\mathbf{0}$$

$$SYS \triangleq A \| S \| B$$

Representation of Woo-Lam protocol

(An example of the Decryption)

$$
\begin{array}{lll}
A \longrightarrow B : & & A \\
B \longrightarrow A : & & N_B \\
A \longrightarrow B : & & \{N_B\}_{K_{AS}} \\
B \longrightarrow S : & & B, \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
S \longrightarrow B : & & \{A, N_B\}_{K_{BS}}
\end{array}
$$

$$
\begin{array}{l}
A \triangleq \overline{a1}\, A.a2(x_a).\overline{a3}\, \{x_a\}_{\mathrm{k}[A,S]}.\mathbf{0} \\
B \triangleq b1(x_b).\,\overline{b2}\, N_B.b3(y_b).\overline{b4}\,(B, \{x_b, y_b\}_{\mathrm{k}[B,S]}).b5(z_b). \\
\quad case\ z_b\ of\ \{u_b\}_{\mathrm{k}[B,S]}\ in\ let(w_b, t_b) = u_b\ in\ [w_b = x_b][u_b = N_B]\,\overline{acc}\, y_b.\mathbf{0} \\
S \triangleq s1(x_s).let\ (x_s', x_s'') = x_s\ in\ case\ x_s''\ of\ \{y_s\}_{\mathrm{k}[x_s',S]}\ in\ let\ (z_s, w_s) = y_s \\
\quad in\ case\ w_s\ of\ \{u_s\}_{\mathrm{k}[z_s,S]}\ in\ \overline{s2}\, \{z_s, u_s\}_{\mathrm{k}[x_s',S]}.\mathbf{0} \\
SYS \triangleq A \| S \| B
\end{array}
$$

Representing each possible run as a trace

$$\begin{array}{lll} A \longrightarrow S: & A, \{B, K_{AB}\}_{K_{AS}} \\ S \longrightarrow B: & \{A, K_{AB}\}_{K_{SB}} \\ A \longrightarrow B: & A, \{A, M\}_{K_{AB}} \end{array}$$

$$\begin{array}{lll} A \longrightarrow B: & A \\ B \longrightarrow A: & N_B \\ A \longrightarrow B: & \{N_B\}_{K_{AS}} \\ B \longrightarrow S: & B, \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\ S \longrightarrow B: & \{A, N_B\}_{K_{BS}} \end{array}$$

- $\overline{a1}(A, \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]})$
- $\overline{a1}(A, \{I, \mathrm{k}[A, I]\}_{\mathrm{k}[A,S]})$
- $\overline{a1}(A, \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]}).$
  $s1(A, \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]})$
- $\overline{a1}(A, \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]}).$
  $b1(\{A, \mathrm{k}[A, B]\}_{\mathrm{k}[B,S]})$ $\times$

- $\overline{a1}A.b1(A).\overline{b2}N_B$
- $b1(A).\overline{b2}N_B.\overline{a1}A.a2(N_I)$
- $b1(A).\overline{b2}N_B.b3(N_B).$
  $\overline{b4}(B, \{A, N_B\}_{\mathrm{k}[B,S]}).b5(B, \{A, N_B\}_{\mathrm{k}[B,S]})$

Environment ability

- If two messages are leaked the environment:
  $(A, \{B, M\}_{\mathrm{k}[A,S]}), (\mathrm{k}[A, S], \{B, M\}_{\mathrm{k}[B,S]})$
- The environment can split and decrypt the message:
  $A, \{B, M\}_{\mathrm{k}[A,S]}, \mathrm{k}[A, S], \{B, M\}_{\mathrm{k}[B,S]}, M \ldots$
- The environment can compose and encrypt the message:
  $\{A\}_{\mathrm{k}[A,S]}, (A, M), \{\{B, M\}_{\mathrm{k}[B,S]}\}_{\mathrm{k}[A,S]} \cdots$
- The environment knows some common messages:
  $A, +\mathrm{k}[A], \ldots$
- The environment can produce new messages:
  $I, N_I, \ldots$
- The environment can produce infinite many messages!
  $(S \triangleright M)$

Formal definition of traces

- Action α is a term of $\overline{a}M$ or $a(M)$. An action is ground if the attached message does not have any variables.
  - eg: $b1\ x$, $\overline{a1}(A, \{B, K_{AB}\}_{K_{AS}})$
- Trace s is a string of ground actions such that for each s', s'' and $a(M)$, if $s = s'.a(M).s''$, then $msg(s') \triangleright M$.
  - $\surd$ $b1(A).\overline{b2}N_B.\overline{a1}A.a2\{I\}_{\mathrm{k}[I,S]}$
  - $\times$ $\overline{a1}(A, \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]}).b1(\{A, \mathrm{k}[A, B]\}_{\mathrm{k}[B,S]})$
- Configuration is a pair $\langle s, P\rangle$,in which s is a trace and P is a closed process (All variables are bound).

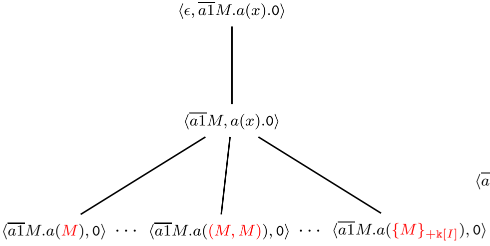
Type

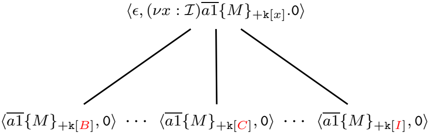
- A type system is proposed such that the type of each variable, message and process can be inferred
  - $\{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]} : \ominus(i * k[i * i])$
  - $b1(x).let\ (y, z) = x\ in\ [z = A].\mathbf{0} : \alpha * i \rightarrow \mathtt{unit}$
  - $x : \alpha * i; \quad y : \alpha; \quad z : i$
- A principal will be stuck if it receives a message whose type can not unify the type of the input variable
  - $b1(\{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]}).let\ (y, z) = \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]}\ in\ [z = A].\mathbf{0}$
- A variable (or a subexpression) with type variable as its type can be unified to any type, so that it can be substituted to any message

Reasons that cause the system to be infinite

Operational semantics

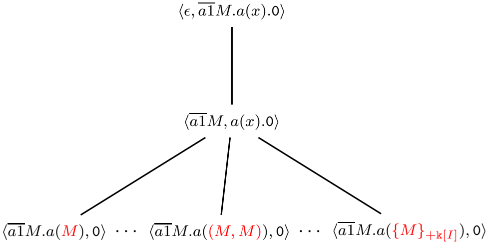
$$
\begin{array}{rl}
(INPUT) & \langle s, a(x).P : \tau_1 \rightarrow \tau_2 \rangle \longrightarrow \langle s.a(M), P\{M/x\} \rangle \\
 & s \triangleright M, \Gamma \vdash M : \tau_1 \\
(OUTPUT) & \langle s, \bar{a}M.P \rangle \longrightarrow \langle s.\bar{a}M, P \rangle \\
(RANGE) & \langle s, (\nu x : \mathcal{A})P \rangle \longrightarrow \langle s, P\{m/x\} \rangle \quad m \in \mathcal{A}
\end{array}
$$

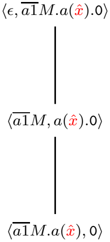




Approach of parametric model

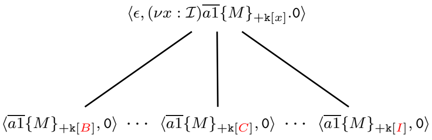
- Each sub-expression with a type variable as its type will be marked with a parametric variable that will not be further instantiated.
- Any message that instantiates the sub-expression will take the same effect to the protocol.

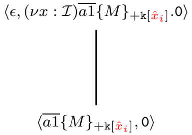




Approach of parametric model (cont.)

- A binder will not be instantiated instantly, it will be instantiated “when needed”(We will explain the “need” later)





$$\begin{array}{rl}
(PINPUT) & \langle \hat{s}, a(\hat{M}).\hat{P} \rangle \longrightarrow_p \langle \hat{s}.a(\hat{M}), \hat{P} \rangle \\
(POUTPUT) & \langle \hat{s}, \overline{a}\hat{M}.\hat{P} \rangle \longrightarrow_p \langle \hat{s}.\overline{a}\hat{M}, \hat{P} \rangle \\
(PRANGLE) & \langle \hat{s}, (\nu \hat{x} : \mathcal{A})\hat{P} \rangle \longrightarrow_p \langle \hat{s}, \hat{P} \rangle
\end{array}$$

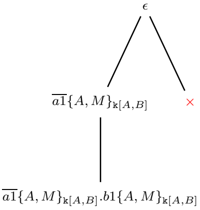
Parametric process and trace

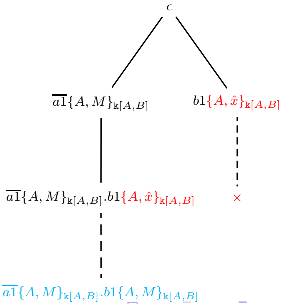
- In a parametric system, parametric traces will be used to represent each possible run of a protocol.
- Each trace in an original system has an abstraction trace in its parametric system.
- A parametric trace may have infinitely many instantiated traces in its original system (named concretizations).
- It may have no concretizations!
  - $\overline{a1}(A, \{B, \mathrm{k}[A, B]\}_{\mathrm{k}[A,S]}).b1(\{A, \hat{x}\}_{\mathrm{k}[B,S]})$

Unchangeable messages

- An unchangeable message (UM) is an encrypted input message such that its key is not leaked in the environment.
- A parametric variable in an unchangeable message cannot be instantiated to arbitrary ground messages. So we must explicitly instantiate it (by unification).
- If a unification is failed, the parametric trace has no concretizations.

$A \longrightarrow B : \{A, M\}_{\mathtt{k}[A,B]}$





Explicit trace

- An explicit trace is a parametric trace that each UM can be deduced by its prefix parametric trace.
- An explicit trace can be obtained by gradually unifying each UM with messages in its prefix parametric trace.

$$\overline{a1}\{A, B, M\}_{\mathtt{k}[A,S]}.s1\{\hat{x}, \hat{y}, \hat{z}\}_{\mathtt{k}[\hat{x},S]}.\overline{s2}(\{\hat{x}, \hat{y}, \hat{z}\}_{\mathtt{k}[\hat{y},\hat{S}]}).b1(\{A, B, \hat{w}\}_{\mathtt{k}[B,S]})$$

$$\downarrow$$

$$\overline{a1}\{A, B, M\}_{\mathtt{k}[A,S]}.s1\{A, B, \hat{z}\}_{\mathtt{k}[A,S]}.\overline{s2}(\{A, B, \hat{z}\}_{\mathtt{k}[B,\hat{S}]}).b1(\{A, B, \hat{z}\}_{\mathtt{k}[B,S]})$$

$$\downarrow$$

$$\overline{a1}\{A, B, M\}_{\mathtt{k}[A,S]}.s1\{A, B, M\}_{\mathtt{k}[A,S]}.\overline{s2}(\{A, B, M\}_{\mathtt{k}[B,\hat{S}]}).b1(\{A, B, M\}_{\mathtt{k}[B,S]})$$

- The number of explicit trace of one parametric trace is finite. Each explicit trace represents a possible run of the protocol.

Deducing to an explicit trace(Woo-Lam)

(More than one unifications)

$$b1(A).\overline{b2}\,N_B.b3(\hat{y}_b).\overline{b4}\,(B, \{A, \hat{y}_b\}_{k[B,S]}).s1(\hat{x}_s, \{\hat{y}_s, \{\hat{z}_s\}_{k[\hat{y}_s,S]}\}_{k[\hat{x}_s,S]}).$$
$$\overline{s2}\,\{\hat{x}_s, \hat{z}_s\}_{k[\hat{y}_s,S]}.b5(\{A, N_B\}_{k[B,S]})$$
$$\hookrightarrow$$
$$b1(A).\overline{b2}\,N_B.b3(N_B).\overline{b4}\,(B, \{A, N_B\}_{k[B,S]}).s1(\hat{x}_s, \{\hat{y}_s, \{\hat{z}_s\}_{k[\hat{y}_s,S]}\}_{k[\hat{x}_s,S]}).$$
$$\overline{s2}\,\{\hat{x}_s, \hat{z}_s\}_{k[\hat{y}_s,S]}.b5(\{A, N_B\}_{k[B,S]}$$

$$b1(A).\overline{b2}\,N_B.b3(\hat{y}_b).\overline{b4}\,(B, \{A, \hat{y}_b\}_{k[B,S]}).s1(\hat{x}_s, \{\hat{y}_s, \{\hat{z}_s\}_{k[\hat{y}_s,S]}\}_{k[\hat{x}_s,S]}).$$
$$\overline{s2}\,\{\hat{x}_s, \hat{z}_s\}_{k[\hat{y}_s,S]}.b5(\{A, N_B\}_{k[B,S]})$$
$$\hookrightarrow$$
$$b1(A).\overline{b2}\,N_B.b3(\hat{y}_b).\overline{b4}\,(B, \{A, \hat{y}_b\}_{k[B,S]}).s1(A, \{B, \{N_B\}_{k[B,S]}\}_{k[A,S]}).$$
$$\overline{s2}\,\{A, N_B\}_{k[B,S]}.b5(\{A, N_B\}_{k[B,S]})$$
$$\hookrightarrow$$
$$\times$$

Authentication properties

- Intuitively, principal A is authenticated to B means if B "thinks" he accepts a message from A, then it really comes from A.
- In the original model, it is defined as: if $\overline{acc}$ occurs in a trace, then $\overline{a3}$ must occurs in the trace before $\overline{acc}$, and both of them are attached with the same message. $(\langle\epsilon, Sys\rangle \models \overline{a3}x \hookleftarrow \overline{acc}x)$
- The definition is equivalent to the same definition defined in explicit traces.

$$
\begin{array}{lll}
A \longrightarrow B: & & A \\
B \longrightarrow A: & & N_B \\
A \longrightarrow B: & & \{N_B\}_{K_{AS}} \\
B \longrightarrow S: & & B, \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
S \longrightarrow B: & & \{A, N_B\}_{K_{BS}}
\end{array}
$$

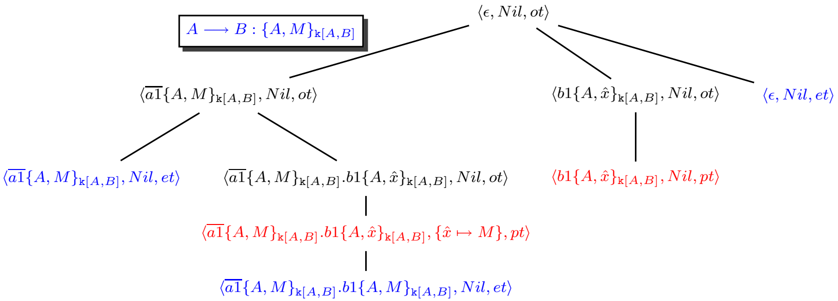
$$
\begin{aligned}
A \triangleq\, & \overline{a1}\, A.a2(x_a).\overline{a3}\,\{x_a\}_{\mathrm{k}[A,S]}.\mathbf{0} \\
B \triangleq\, & b1(x_b).\,\overline{b2}\, N_B.b3(y_b).\overline{b4}\,(B, \{x_b, y_b\}_{\mathrm{k}[B,S]}). \\
& b5(z_b).case\ z_b\ of\ \{u_b\}_{\mathrm{k}[B,S]}\ in \\
& let(w_b, t_b) = u_b\ in\ [w_b = x_b][u_b = N_B] \\
& \overline{acc}\, y_b.\mathbf{0}
\end{aligned}
$$

On-the-fly model checking by Maude

- Two reasons to use Maude:
  - A new parametric trace generation is decided on-the-fly by trying to unify UM(it may fail)
  - It is easily to transfer a specification property to a reachability problem.
- The way of implementation by Maude
  - Each elementary definition and function in the parametric model is implemented to functional modules.
  - A trace generating system is represented in a system module.
  - `search` command is used to find whether the negation of a specification is reachable.

Trace generating system

- A state of the trace generating system is a 3-tuple: $\langle tr, S, k \rangle$, where
    - *tr* is a parametric trace.
    - *S* is a list of substitutions.
    - *k* is a type of *tr*, where $k \in \{ot, et, pt\}$. *ot* represents an original trace, *et* represents an explicit trace and *pt* represents a pending trace.



Transition rules of Woo-Lam protocol

- Initial state: $\langle \epsilon, \{\}, ot \rangle$
- Parametric transition relation:
  - $\langle tr, S, ot \rangle \hookrightarrow \langle tr.\overline{a}1A, S, ot \rangle$ if $\overline{a1} \notin tr$
  - $\langle tr, S, ot \rangle \hookrightarrow \langle tr.b1(\hat{x}), S, ot \rangle$ if $b1 \notin tr$
  - ...
- Reduction relation:
  - $\langle tr, \theta \# S, pt \rangle \hookrightarrow \langle tr\theta, ES(tr\theta), pt \rangle$
    if not $Exp(tr\theta)$
  - $\langle tr, \theta \# S, pt \rangle \hookrightarrow \langle tr, S, pt \rangle$
- Trace type transferred relation:
  - $\langle tr, S, ot \rangle \hookrightarrow \langle tr, ES(tr), pt \rangle$ if not $Exp(tr)$
  - $\langle tr, S, ot \rangle \hookrightarrow \langle tr, \{\}, et \rangle$ if $Exp(tr)$
  - $\langle tr, \theta \# S, pt \rangle \hookrightarrow \langle tr\theta, \{\}, et \rangle$ if $Exp(tr\theta)$

$$
\begin{array}{ll}
A \longrightarrow B: & A \\
B \longrightarrow A: & N_B \\
A \longrightarrow B: & \{N_B\}_{K_{AS}} \\
B \longrightarrow S: & B, \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
S \longrightarrow B: & \{A, N_B\}_{K_{BS}}
\end{array}
$$

Part source code of Woo-Lam protocol

Woo-lam.maude

```
eq init =  < [ Nil ] , NIL , ot >  .
------------------------------------------------------------------------------------------------------------
crl [A_1] : < [ TR1 ], SUBLIST, ot >  => < [ (TR1 . < a(1), o, name(0) >) ], SUBLIST, ot >
                        if not labelinTrace (TR1, a(1)) .
crl [A_2] : < [ TR1 ], SUBLIST, ot >  => < [ TR1 .  < a(2), i, px(0) > .
                    < a(3), o, {px(0)}k[name(0),name(2)] > ], SUBLIST, ot >
                     if  labelinTrace (TR1, a(1)) and  not labelinTrace (TR1, a(2)) .

crl [B_1] : < [ TR1 ], SUBLIST, ot >  => < [ (TR1 . < b(1), i, name(0) > . < b(2), o, name(3) > )], SUBLIST, ot >
                        if not labelinTrace (TR1, b(1)) .
crl [B_3] : < [ TR1 ], SUBLIST, ot >  => < [ (TR1 . < b(3), i, px(1) > .
                < b(4), o, (name(1),{(name(0) , px(1))}k[name(1),name(2)]) >) ], SUBLIST, ot >
                        if labelinTrace (TR1, b(1)) and labelinTrace (TR1, b(2))
                        and not labelinTrace (TR1, b(3)) .
crl [B_5] : < [ TR1 ], SUBLIST, ot >  =>
          < [ (TR1 . < b(5), i, {name(0), name(3)}k[name(1),name(2)] > . < acc, o, px(1) > ) ],SUBLIST, ot >
            if labelinTrace (TR1, b(1)) and labelinTrace (TR1, b(2)) and labelinTrace (TR1, b(3))
                        and labelinTrace (TR1, b(4)) and not labelinTrace (TR1, b(5)) .

crl [S_1] : < [ TR1 ], SUBLIST, ot >  =>
          < [ (TR1 . < s(1), i, (px(2), {px(3),{px(4)}k[px(3),name(2)]}k[px(2),name(2)]) > .
                < s(2), o, {px(3), px(4)}k[px(2),name(2)] > ) ],
                    SUBLIST, ot >   if not labelinTrace (TR1, s(1)) .
************************************************************************************************************
crl [ot_to_ht] : < [ TR1 ] , SUBLIST, ot > =>
          < [ TR1 ]  , getSubstitutionlist( getMessage(analyzingTrace(TR1, nil)),
          elementary( getMessagelist(analyzingTrace(TR1, nil))), NIL), ht >
          if not isExplicitTrace (TR1) .

crl [ot_to_et] : < [ TR1 ] , SUBLIST, ot > => < [ TR1 ], NIL , et >
            if  isExplicitTrace (TR1) .
```

Experimental results

protocols	sessions	lines	states	times(ms)	flaws
NSPK protocol	1	20+330	46	130	detected
fixed NSPK protocol	1	20+330	164	637	secure
Woo-Lam protocol*	1	25+330	168	160	detected
Yahalom protocol	2	36+330	536	1,039	detected
Otway-Ree protocol	2	34+330	2,164	22,316	detected
Woo-lam protocol	2	42+330	105,423	476,507	detected

The tests were preformed on a Pentium 1.4 GHz, 1.5G Memory, Win XP.

A benchmark of analyzing security protocol (by horn logic)

protocols	times(ms)
NSPK protocol	8
fixed NSPK protocol	5
Woo-Lam protocol*	6
Yahalom protocol	16
Otway-Ree protocol	14
Woo-lam protocol	fails

Related work
(Benchmark)

- Based on Horn clauses and resolution, checking the properties in infinite sessions of the protocol.
  - $att(\{m\}_k) \wedge att(k) \rightarrow att(m)$
  - $att(nb) \rightarrow att(\{nb\}_{k_{as}})$ (Woo-Lam protocol)
- It sometimes does not terminate. (NSPK, Woo-Lam)
- A tag system makes system terminating. Security of a tagged protocol does not imply security of its untagged version.
- Related references are:
- Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Logic Programming. CSFW-14, 2001
- Bruno Blanchet and Andreas Podelski. Verification of Cryptographic Protocols: Tagging Enforces Termination. Theoretical Computer Science 333, 2005

Related work
(OFMC, Lazy intruder)

- David Basin, et al. proposed an On-the-fly model checking methods (OFMC).
- They use a high-level language HLPSL to represent a protocol, then translate automatically to a low-level one, IF.
- An intruder's messages are instantiated when necessary (UM is similar).
- An intruder's role is explicitly assigned, thus flexible and efficient (we need to check each role).

```
... ...
Messages
1. A -> B : A, NA
2. B -> S:  B, {|A, NA, NB|}k(B,S)
3. ... ...
Session_instances
[A:a; B:b; S:s]
[A:i; B:b; S:s]
... ...
```

```
state(roleA,step0,sess1,a,b,s,k(a,s)).
state(roleB,step0,sess1,a,b,s,k(b,s)).
state(roleS,step0,sess1,a,b,s,k).
state(roleA,step0,sess2,a,b,s,k(a,s)).
state(roleS,step0,sess2,a,b,s,k).
i_knows(a).i_knows(b).i_knows(s).
i_knows(s).i_knows(i).i_knows(k(i,s)).
```

Related works
(Process calculus)

- Gavin Lowe firstly uses trace analysis on CSP. The intruder is represented as a recursive process. states are restricted by imposing upper-bounds.
- Abadi et al. use some bisimulation to define the security properties. The main problem is that those equivalences are usually undecidable for implementation.
  - $Sys_{imp} \cong Sys_{spec}$
- Their another approach is statical analysis by type system. The attacker model is weaker than Dolev-Yao model, assuming that the intruder is partially trusted.

Related work

(Type system vs. tag system)

- J. Heather et al. show that a tagging system can prevent type flaw attacks.
- A tag is a few bits attached to each message, with different bit patterns allocated to different types
  - $(\mathtt{nonce}, N)$ means N is intended to be a nonce.
- The work infers that the depth of ground messages can be bounded in the search for an attack when the principals are bounded.

$((\mathtt{agent}, \{\mathtt{agent}, \{\mathtt{nonce}\}\mathtt{sk}\}\mathtt{sk}),$

$((\mathtt{agent}, B), (\{\mathtt{agent}, \{\mathtt{nonce}\}\mathtt{sk}\}\mathtt{sk}, \{(\mathtt{agent}, A), \{(\mathtt{nonce}, N_B)\}_{K_{AS}}\}_{K_{BS}})))$

$$
\begin{aligned}
A \longrightarrow B &: \quad A \\
B \longrightarrow A &: \quad N_B \\
A \longrightarrow B &: \quad \{N_B\}_{K_{AS}} \\
B \longrightarrow S &: \quad B, \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
S \longrightarrow B &: \quad \{A, N_B\}_{K_{BS}}
\end{aligned}
$$

$$
\begin{aligned}
I(A) \longrightarrow B &: \quad A \\
B \longrightarrow I(A) &: \quad N_B \\
I(A) \longrightarrow B &: \quad N_B \\
B \longrightarrow I(S) &: \quad B, \{A, N_B\}_{K_{BS}} \\
I(S) \longrightarrow B &: \quad \{A, N_B\}_{K_{BS}}
\end{aligned}
$$

Related work

(Binder vs. Principals Restriction)

- The research of H. Comon-Lundh et al. is based on the Horn clauses, which proved that it is sufficient to only consider a bound number of principals when verifying some security properties.
- Given an attack using *n* agents, we project every honest identity on one single identity and every dishonest identity on one dishonest identity.

- $A \rightarrow B: \quad A, N_a$
  (Yahalom protocol)
- $\texttt{Fresh}(t, s), T(t) \Rightarrow T([st(a, 0, \langle a, b, srv \rangle), s].t)$
  s: session, t: trace
- $\texttt{T}(\texttt{t})$, $\texttt{In}([st(a, 0, \langle a, b, srv \rangle), s], t)$, $\texttt{NotPlayed}(a, 1, s, t) \Rightarrow T([\langle a, n_1(a, s) \rangle, s].[st(a, 1, \langle a, b, srv \rangle), s].t)$
- Solution: Keep a uninstantiated.

Future work

- Develop a parser that transfers an original system to its counterpart Maude source code.
- Try to perform model checking on other security properties such as non-repudiation, fairness, anonymity, etc.
- Extend the calculus to one that can define recursive process so that we can model checking a protocol with infinite sessions.

Thank you!