Topici speciale în logică și securitate I

Protocoale de securitate III

Ioana Leuștean

Master anul II, Sem. I, 2019-2020

Protocolul Needham-Schroeder

$Protocol = Role \rightharpoonup RoleSpec$

$(kn, s) \in RoleSpec$ cu $kn \in \mathcal{P}(RoleTerm)$ și $s \in RoleEvent^*$

$P(R)$ este specificarea rolului R pentru orice $P \in Protocol$ și $R \in Role$.

Limbajul asociat protocolului Needham-Schroeder este
$Role = \{i, r\}$, $Fresh = \{ni, nr\}$, $Func = \emptyset$, $Label = \{1, 2, 3, 4, 5\}$, $Var = \{V, W\}$

$$\begin{aligned} NS(i) = \quad & (\{i, r, ni, sk(i), pk(i), pk(r)\}, \\ & [send_1(i, r, \{\!| ni, i |\!\}_{pk(r)}), \\ & recv_2(r, i, \{\!| ni, V |\!\}_{pk(i)}), \\ & send_3(i, r, \{\!| V |\!\}_{pk(r)}), \\ & claim_4(i, synch)]) \end{aligned}$$

$$\begin{aligned} NS(r) = \quad & (\{i, r, nr, sk(r), pk(r), pk(i)\}, \\ & [recv_1(i, r, \{\!| W, i |\!\}_{pk(r)}), \\ & send_2(r, i, \{\!| W, nr |\!\}_{pk(i)}), \\ & recv_3(i, r, \{\!| nr |\!\}_{pk(r)}), \\ & claim_5(r, synch)]) \end{aligned}$$

Exemplul

În exemplele de mai jos, k și $k2$ sunt chei simetrice.

$$P_1(i) = (\{i, r, k\}, [send_1(i, r, \{\!|\, i, r, V \,|\!\}_k), recv_2(r, i, \{\!|\, V, r \,|\!\}_k)])$$

$$P_2(i) = (\{i, r, k\}, [recv_1(r, i, \{\!|\, i, r, \{\!|\, W \,|\!\}_{k2} \,|\!\}_k), send_2(i, r, \{\!|\, W \,|\!\}_{k2})])$$

Exemplul

În exemplele de mai jos, k și $k2$ sunt chei simetrice.

$$P_1(i) = (\{i, r, k\}, [send_1(i, r, \{\!\!\{ i, r, V \}\!\!\}_k), recv_2(r, i, \{\!\!\{ V, r \}\!\!\}_k)])$$

$$P_2(i) = (\{i, r, k\}, [recv_1(r, i, \{\!\!\{ i, r, \{\!\!\{ W \}\!\!\}_{k2} \}\!\!\}_k), send_2(i, r, \{\!\!\{ W \}\!\!\}_{k2})])$$

Ce este greșit?

Ordinea evenimentelor

Pentru $P \in Protocol$ definim:

- Ordinea evenimentelor unui rol R cu $P(R) = (kn, [\epsilon_1, \ldots, \epsilon_n])$ este

$$\epsilon_1 <_R \cdots <_R \epsilon_n$$

  este o relație de ordine totală.

Ordinea evenimentelor

Pentru $P \in \mathit{Protocol}$ definim:

- Ordinea evenimentelor unui rol R cu $P(R) = (kn, [\epsilon_1, \ldots, \epsilon_n])$ este

  $$\epsilon_1 <_R \cdots <_R \epsilon_n$$

  este o relație de ordine totală.

- Relația de comunicare: $\dashrightarrow \subseteq \mathit{RoleEvent} \times \mathit{RoleEvent}$

  $\epsilon_1 \dashrightarrow \epsilon_2$ ddacă există $l \in \mathit{Label}$, $R, R' \in \mathit{Role}$, $rt_1, rt_2 \in \mathit{RoleTerm}$ astfel încât $\epsilon_1 = \mathit{send}_l(R, R', rt_1)$ și $\epsilon_2 = \mathit{recev}_l(R', R, rt_2)$

Ordinea evenimentelor

Pentru $P \in \mathit{Protocol}$ definim:

- Ordinea evenimentelor unui rol R cu $P(R) = (kn, [\epsilon_1, \ldots, \epsilon_n])$ este

$$\epsilon_1 <_R \cdots <_R \epsilon_n$$

  este o relație de ordine totală.

- Relația de comunicare: $\dashrightarrow \subseteq \mathit{RoleEvent} \times \mathit{RoleEvent}$

  $\epsilon_1 \dashrightarrow \epsilon_2$ ddacă există $l \in \mathit{Label}$, $R, R' \in \mathit{Role}$, $rt_1, rt_2 \in \mathit{RoleTerm}$ astfel încât $\epsilon_1 = \mathit{send}_l(R, R', rt_1)$ și $\epsilon_2 = \mathit{recev}_l(R', R, rt_2)$

- Relația de ordine a protocolului:

$$\prec_P = \left(\dashrightarrow \ \cup \bigcup_{R \in \mathit{Role}} <_R \right)^+$$

$\prec_P \subseteq$ *RoleEvent* $\times$ *RoleEvent*

Exemplu:
Ordinea evenimentelor pentru protocolul Needham-Schroeder

$$\begin{array}{ccc}
send_1(i, r, \{\!|\, ni, i \,|\!\}_{pk(r)}) & \prec_{NS} & recv_1(i, r, \{\!|\, W, i \,|\!\}_{pk(r)}) \\
\curlywedge_{NS} & & \curlywedge_{NS} \\
recv_2(r, i, \{\!|\, ni, V \,|\!\}_{pk(i)}) & \succ_{NS} & send_2(r, i, \{\!|\, W, nr \,|\!\}_{pk(i)}) \\
\curlywedge_{NS} & & \curlywedge_{NS} \\
send_3(i, r, \{\!|\, V \,|\!\}_{pk(r)}) & \prec_{NS} & recv_3(i, r, \{\!|\, nr \,|\!\}_{pk(r)}) \\
\curlywedge_{NS} & & \curlywedge_{NS} \\
claim_4(i, synch) & & claim_5(r, synch)
\end{array}$$

Execuția unui protocol

- Specificarea unui protocol descrie fiecare rol in parte.
- Rolurile sunt niște tipare pentru acțiunile agenților din sistem.
- Atunci cand se execută un protocol, un agent poate orice rol, de oricâte ori, eventual în paralel.
- Atunci când instanțiem un rol trebuie să ținem seama ca variabilele *fresh* să fie unice pe instanțiere.
- Vom identifica o execuție, posibil parțială, a unui rol de către un agent prin identificatori speciali, *RID* (*Run Identifiers*).
- Folosind acești identificatori, definim un alt tip de termeni, *RunTerms*, cu ajutorul cărora definim execuția concretă unui protocol.

Descrierea unei sesiuni *RunTerms*

$$
\begin{aligned}
RunTerm \quad ::= \quad & Var^{\#RID} \mid Fresh^{\#RID} \mid Role^{\#RID} \\
& \mid Agent \\
& \mid Func\,(RunTerm^{*}) \\
& \mid (RunTerm, RunTerm) \\
& \mid \{\!|\, RunTerm \,|\!\}_{RunTerm} \\
& \mid AdversaryFresh \\
& \mid sk(RunTerm) \mid pk(RunTerm) \mid k(RunTerm, RunTerm)
\end{aligned}
$$

- $^{-1}$: $RunTerm \rightarrow RunTerm$
- *AdversaryFresh* termenii de bază generați de adversar.

Sistem de deducție pe termeni

$$\vdash \subseteq \mathcal{P}(RoleTerm \cup RunTerm) \times (RoleTerm \cup RunTerm)$$

$M \vdash t$ modelează ceea ce se pate deduce știind M

$\vdash$ este cea mai mică relație care satisface următoarele proprietăți:

dacă	$t \in M$	atunci	$M \vdash t$
dacă	$M \vdash t_1$ și $M \vdash t_2$	atunci	$M \vdash (t_1, t_2)$
dacă	$M \vdash (t_1, t_2)$	atunci	$M \vdash t1$ și $M \vdash t_2$
dacă	$M \vdash t$ și $M \vdash k$	atunci	$M \vdash \{\!\mid t \mid\!\}_k$
dacă	$M \vdash \{\!\mid t \mid\!\}_k$ și $M \vdash k^{-1}$	atunci	$M \vdash t$
dacă	$M \vdash t_1$ și ... și $M \vdash t_n$	atunci	$M \vdash f(t_1, \ldots, t_n)$

unde $f \in Func$ are aritatea n.

Descrierea unei sesiuni folosind *RunTerms*

- În specificarea unui protocol folosim termeni generici *RoleTerms* care vor fi instanțiati (individualizați) atunci când descriem execuția unei sesiuni

Descrierea unei sesiuni folosind *RunTerms*

- În specificarea unui protocol folosim termeni generici *RoleTerms* care vor fi instanțiati (individualizați) atunci când descriem execuția unei sesiuni

  De exemplu: termenul generic i care desemnează rolul inițiatorului va fi instanțiat cu A (Alice) care desemnează un agent concret;

Descrierea unei sesiuni folosind *RunTerms*

- În specificarea unui protocol folosim termeni generici *RoleTerms* care vor fi instanțiati (individualizați) atunci când descriem execuția unei sesiuni

  De exemplu: termenul generic i care desemnează rolul inițiatorului va fi instanțiat cu A (Alice) care desemnează un agent concret; valoarea fresh generică ni va fi instanțiată cu $ni^{\#1}$, $ni^{\#2}$, ... care reprezină valorile concrete generate în prima sesiunea, în a doua sesiune , etc.

Descrierea unei sesiuni folosind *RunTerms*

- În specificarea unui protocol folosim termeni generici *RoleTerms* care vor fi instanțiati (individualizați) atunci când descriem execuția unei sesiuni

  De exemplu: termenul generic *i* care desemnează rolul inițiatorului va fi instanțiat cu A (Alice) care desemnează un agent concret; valoarea fresh generică *ni* va fi instanțiată cu $ni^{\#1}$, $ni^{\#2}$, ... care reprezină valorile concrete generate în prima sesiunea, în a doua sesiune , etc.

- O *instanțiere* este un triplet

$$(\theta, \rho, \sigma) \in RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$$

Descrierea unei sesiuni folosind *RunTerms*

- În specificarea unui protocol folosim termeni generici *RoleTerms* care vor fi instanțiati (individualizați) atunci când descriem execuția unei sesiuni

  De exemplu: termenul generic *i* care desemnează rolul inițiatorului va fi instanțiat cu *A* (Alice) care desemnează un agent concret; valoarea fresh generică *ni* va fi instanțiată cu $ni^{\#1}$, $ni^{\#2}$, ... care reprezină valorile concrete generate în prima sesiunea, în a doua sesiune , etc.

- O *instanțiere* este un triplet

$$(\theta, \rho, \sigma) \in RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$$

Notații: pentru o funcție *f* vom nota cu *dom*(*f*) și *ran*(*f*) domeniul și, respectiv, imaginea funcției.

Instanțierea

Fie *Inst* mulțimea *instanțierilor*,

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

Pentru $inst = (\theta, \rho, \sigma) \in Inst$ definim

$$\langle inst \rangle : RoleTerm \to RunTerm$$

Pentru simplitate, în loc de $\langle inst \rangle(t)$ vom scrie $inst(t)$.

Instanțierea

Fie *Inst* mulțimea *instanțierilor*,

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

Pentru $inst = (\theta, \rho, \sigma) \in Inst$ definim

$$\langle inst \rangle : RoleTerm \to RunTerm$$

Pentru simplitate, în loc de $\langle inst \rangle(t)$ vom scrie $inst(t)$.

- $inst(n) = n^{\#\theta}$ pentru $n \in Fresh$

Instanțierea

Fie *Inst* mulțimea *instanțierilor*,

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

Pentru $inst = (\theta, \rho, \sigma) \in Inst$ definim

$$\langle inst \rangle : RoleTerm \to RunTerm$$

Pentru simplitate, în loc de $\langle inst \rangle(t)$ vom scrie $inst(t)$.

- $inst(n) = n^{\#\theta}$ pentru $n \in Fresh$
- $inst(R) = \rho(R)$ pentru $R \in Role \cap dom(\rho)$
  $inst(R) = R^{\#\theta}$ pentru $R \in Role \setminus dom(\rho)$

Instanțierea

Fie *Inst* mulțimea *instanțierilor*,

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

Pentru $inst = (\theta, \rho, \sigma) \in Inst$ definim

$$\langle inst \rangle : RoleTerm \to RunTerm$$

Pentru simplitate, în loc de $\langle inst \rangle(t)$ vom scrie $inst(t)$.

- $inst(n) = n^{\#\theta}$ pentru $n \in Fresh$
- $inst(R) = \rho(R)$ pentru $R \in Role \cap dom(\rho)$
  $inst(R) = R^{\#\theta}$ pentru $R \in Role \setminus dom(\rho)$
- $inst(V) = \sigma(V)$ pentru $V \in Var \cap dom(\sigma)$
  $inst(V) = V^{\#\theta}$ pentru $V \in Role \setminus dom(\sigma)$

Instanțierea

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (\theta, \rho, \sigma) \in Inst$

- $inst(f(t_1, \ldots, t_n)) = f(inst(t_1), \ldots, inst(t_n))$

Instanțierea

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (\theta, \rho, \sigma) \in Inst$

- $inst(f(t_1, \ldots, t_n)) = f(inst(t_1), \ldots, inst(t_n))$
- $inst(t_1, t_2) = (inst(t_1), inst(t_2))$

Instanțierea

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (\theta, \rho, \sigma) \in Inst$

- $inst(f(t_1, \ldots, t_n)) = f(inst(t_1), \ldots, inst(t_n))$
- $inst(t_1, t_2) = (inst(t_1), inst(t_2))$
- $inst(\{\!|\, t_1 \,|\!\}_{t_2}) = \{\!|\, inst(t_1) \,|\!\}_{inst(t_2)}$

Instanțierea

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (\theta, \rho, \sigma) \in Inst$

- $inst(f(t_1, \ldots, t_n)) = f(inst(t_1), \ldots, inst(t_n))$
- $inst(t_1, t_2) = (inst(t_1), inst(t_2))$
- $inst(\{\!|\, t_1 \,|\!\}_{t_2}) = \{\!|\, inst(t_1) \,|\!\}_{inst(t_2)}$
- $inst(sk(t)) = sk(inst(t))$, $inst(pk(t)) = pk(inst(t))$

Instanțierea

$Inst = RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (\theta, \rho, \sigma) \in Inst$

- $inst(f(t_1, \ldots, t_n)) = f(inst(t_1), \ldots, inst(t_n))$
- $inst(t_1, t_2) = (inst(t_1), inst(t_2))$
- $inst(\{\!|\, t_1 \,|\!\}_{t_2}) = \{\!|\, inst(t_1) \,|\!\}_{inst(t_2)}$
- $inst(sk(t)) = sk(inst(t)),\ inst(pk(t)) = pk(inst(t))$
- $inst(k(t_1, t_2)) = k(inst(t_1), inst(t_2))$

Exemplu: Instanțierea

$$inst = (\theta, \rho, \sigma) \in RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$$

$$inst = (2, \{i \mapsto B, r \mapsto A\}, \{W \mapsto ni^{\#1}\})$$

Exemplu: Instanțierea

$inst = (\theta, \rho, \sigma) \in RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (2, \{i \mapsto B, r \mapsto A\}, \{W \mapsto ni^{\#1}\})$

$t = \{\!| W, nr, r |\!\}_{pk(i)} \in RoleTerm$

Exemplu: Instanțierea

$inst = (\theta, \rho, \sigma) \in RID \times (Role \rightharpoonup Agent) \times (Var \rightharpoonup RunTerm)$

$inst = (2, \{i \mapsto B, r \mapsto A\}, \{W \mapsto ni^{\#1}\})$

$t = \{\!| W, nr, r |\!\}_{pk(i)} \in RoleTerm$

$inst(\{\!| W, nr, r |\!\}_{pk(i)}) = \{\!| ni^{\#1}, nr^{\#2}, A |\!\}_{pk(B)} \in RunTerm$

Execuții posibile

- $Run = Inst \times RoleEvent^*$ mulțimea *execuțiilor posibile*

Execuții posibile

- $Run = Inst \times RoleEvent^*$ mulțimea *execuțiilor posibile*
- $runsof : Protocol \times Roles \to \mathcal{P}(Run)$

  $$runsof(P,R) = \{(inst, s) \mid \text{există } kn \text{ astfel încât } P(R) = (kn, s)$$
  $$inst = (\theta, \rho, \sigma) \text{ cu } dom(\rho) = roles(s)\}$$

  unde $R \in dom(P)$.

Execuții posibile

- $Run = Inst \times RoleEvent^*$ mulțimea *execuțiilor posibile*
- $runsof : Protocol \times Roles \to \mathcal{P}(Run)$

  $$runsof(P,R) = \{(inst, s) \mid \text{există } kn \text{ astfel încât } P(R) = (kn, s)$$
  $$inst = (\theta, \rho, \sigma) \text{ cu } dom(\rho) = roles(s)\}$$

  unde $R \in dom(P)$.
- Pentru $F \subseteq Run$ definim

  $$runIds(F) = \{\theta \mid \text{există } \rho, \sigma, s \text{ cu } ((\theta, \rho, \sigma), s) \in F\}$$

Semantica operațională: stări

$$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$$

Semantica operațională: stări

$$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$$

$st = \langle\!\langle AKN, F \rangle\!\rangle \in State$

- prima componentă reprezintă cunoașterea adversarului AKN
- a doua componentă reprezintă restul execuției

Semantica operațională: stări

$$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$$

$st = \langle\!\langle AKN, F \rangle\!\rangle \in State$

- prima componentă reprezintă cunoașterea adversarului AKN
- a doua componentă reprezintă restul execuției

Exemple:

$st_1 = \langle\!\langle \{A, B, pk(A), pk(B)\}, \{(\,(2, \{i \mapsto A, r \mapsto B\}, \emptyset), send_1(i, r, \{\!|\, ni \,|\!\}_{pk(r)})\,)\} \rangle\!\rangle$

Semantica operațională: stări

$$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$$

$st = \langle\!\langle AKN, F \rangle\!\rangle \in State$

- prima componentă reprezintă cunoașterea adversarului AKN
- a doua componentă reprezintă restul execuției

Exemple:

$st_1 = \langle\!\langle \{A, B, pk(A), pk(B)\}, \{(\,(2, \{i \mapsto A, r \mapsto B\}, \emptyset), send_1(i, r, \{\!|\, ni \,|\!\}_{pk(r)})\,)\}\rangle\!\rangle$

$st_2 = \langle\!\langle \{A, B, pk(A), pk(B), \{\!|\, ni^{\#2} \,|\!\}_{pk(A)}\}, \emptyset \rangle\!\rangle$

Sistem cu tranziții etichetate (LTS)

$(St, L, \rightarrow st_0)$ LTS, unde

- St este mulțimea stărilor
- L este mulțimea etichetelor
- $\rightarrow \subseteq S \times L \times S$ este relația de tranziție
- $st_0 \in St$ este starea inițială

Sistem cu tranziții etichetate (LTS)

$(St, L, \to st_0)$ LTS, unde

- St este mulțimea stărilor
- L este mulțimea etichetelor
- $\to \subseteq S \times L \times S$ este relația de tranziție
- $st_0 \in St$ este starea inițială

Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ cu $st_i \stackrel{\alpha_{i+1}}{\to} st_{i+1}$

Sistem cu tranziții etichetate (LTS)

$(St, L, \to st_0)$ LTS, unde

- St este mulțimea stărilor
- L este mulțimea etichetelor
- $\to \subseteq S \times L \times S$ este relația de tranziție
- $st_0 \in St$ este starea inițială

Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ cu $st_i \stackrel{\alpha_{i+1}}{\to} st_{i+1}$

Urmă (*trace*): $[\alpha_1, \alpha_2, \ldots, \alpha_n]$

Sistem cu tranziții etichetate (LTS)

$(St, L, \rightarrow st_0)$ LTS, unde

- St este mulțimea stărilor
- L este mulțimea etichetelor
- $\rightarrow \subseteq S \times L \times S$ este relația de tranziție
- $st_0 \in St$ este starea inițială

Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ cu $st_i \stackrel{\alpha_{i+1}}{\rightarrow} st_{i+1}$

Urmă (*trace*): $[\alpha_1, \alpha_2, \ldots, \alpha_n]$

Semantica operațională a unui protocol P va fi modelată printr-un sistem de tranziții etichetat
$(State, RunEvent, \rightarrow, st_0(P))$

Semantica operațională

$$(State, RunEvent, \to, st_0(P))$$

$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$

Semantica operațională

$$(State, RunEvent, \rightarrow, st_0(P))$$

$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$

- $st_0(P) = \langle\langle AKN_0(P), \emptyset \rangle\rangle$ unde $AKN_0(P)$ este cunoașterea inițială a adversarului

Semantica operațională

$$(State, RunEvent, \rightarrow, st_0(P))$$

$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$

- $st_0(P) = \langle\langle AKN_0(P), \emptyset \rangle\rangle$ unde $AKN_0(P)$ este cunoașterea inițială a adversarului

  În modelul Dolev-Yao, adversarul cunoaște informația publică despre rețea, agenții, poate genera valori *fresh* (diferite de ale agenților) și are acces la cunoștințele inițiale ale agenților corupți.

Semantica operațională

$$(State, RunEvent, \rightarrow, st_0(P))$$

$State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$

- $st_0(P) = \langle\!\langle AKN_0(P), \emptyset \rangle\!\rangle$ unde $AKN_0(P)$ este cunoașterea inițială a adversarului

  În modelul Dolev-Yao, adversarul cunoaște informația publică despre rețea, agenții, poate genera valori *fresh* (diferite de ale agenților) și are acces la cunoștințele inițiale ale agenților corupți.

  Exemplu:

  $AKN_0(NS) = AdversaryFresh \cup Agent \cup \{pk(A) \mid A \in Agent\}$

Etichetele sistemului de tranziții: *RunEvent*

În *specificarea protocolului* am folosit:

$$\begin{array}{lcl} RoleEvent_R & ::= & send_{Label}(R, Role, RoleTerm) \\ & & \mid recv_{Label}(Role, R, RoleTerm) \\ & & \mid claim_{Label}(R, Claim[, RoleTerm]) \end{array}$$

Etichetele sistemului de tranziții: *RunEvent*

În *specificarea protocolului* am folosit:

$$
\begin{aligned}
RoleEvent_R \quad ::= \quad & send_{Label}(R, Role, RoleTerm)\\
& \mid recv_{Label}(Role, R, RoleTerm)\\
& \mid claim_{Label}(R, Claim[, RoleTerm])
\end{aligned}
$$

$$RoleEvent = \bigcup_{R \in Role} RoleEvent_R$$

Etichetele sistemului de tranziții: *RunEvent*

În *specificarea protocolului* am folosit:

$$\begin{aligned} RoleEvent_R \quad ::= \quad & send_{Label}(R, Role, RoleTerm) \\ & \mid recv_{Label}(Role, R, RoleTerm) \\ & \mid claim_{Label}(R, Claim[, RoleTerm]) \end{aligned}$$

$$RoleEvent = \bigcup_{R \in Role} RoleEvent_R$$

Pentru a descrie *execuția protocolului* definim:

$$RunEvent = Inst \times (RoleEvent \cup \{create(R) \mid R \in Role\}$$

- $create(R)$ marchează inițierea unei execuții pentru un rol din protocol.

Semantica operațională

$$(State, RunEvent, \rightarrow, st_0(P))$$

- $State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$ unde $Run = Inst \times RoleEvent^*$
- $st_0(P) = \langle\langle AKN_0(P), \emptyset \rangle\rangle$ unde $AKN_0(P)$ este cunoașterea inițială a adversarului

Semantica operațională

$$(State, RunEvent, \rightarrow, st_0(P))$$

- $State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$ unde $Run = Inst \times RoleEvent^*$
- $st_0(P) = \langle\!\langle AKN_0(P), \emptyset \rangle\!\rangle$ unde $AKN_0(P)$ este cunoașterea inițială a adversarului
- $RunEvent = Inst \times (RoleEvent \cup \{create(R) \mid R \in Role\}$

Semantica operațională

$$(State, RunEvent, \rightarrow, st_0(P))$$

- $State = \mathcal{P}(RunTerm) \times \mathcal{P}(Run)$ unde $Run = Inst \times RoleEvent^*$
- $st_0(P) = \langle\langle AKN_0(P), \emptyset \rangle\rangle$ unde $AKN_0(P)$ este cunoașterea inițială a adversarului
- $RunEvent = Inst \times (RoleEvent \cup \{create(R) \mid R \in Role\}$
- Sistemul de tranziții are patru reguli, corespunzătoare evenimentelor: *create*, *send*, *recv*, *claim*

Sistemul de tranziții etichetate

$$(State, RunEvent, \rightarrow, st_0(P))$$

- Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ unde $\alpha_i \in RunEvent$ și $st_i = \langle\langle AKN_i, F_i \rangle\rangle$

Sistemul de tranziții etichetate

$$(State, RunEvent, \rightarrow, st_0(P))$$

- Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ unde $\alpha_i \in RunEvent$ și $st_i = \langle\!\langle AKN_i, F_i \rangle\!\rangle$
- Urmă (*trace*): $[\alpha_1, \alpha_2, \ldots, \alpha_n]$

Sistemul de tranziții etichetate

$$(State, RunEvent, \rightarrow, st_0(P))$$

- Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ unde $\alpha_i \in RunEvent$ și $st_i = \langle\langle AKN_i, F_i \rangle\rangle$
- Urmă (*trace*): $[\alpha_1, \alpha_2, \ldots, \alpha_n]$
- Știind starea inițială, putem defini execuția protocolului folosind urmele.

Sistemul de tranziții etichetate

$$(State, RunEvent, \rightarrow, st_0(P))$$

- Execuție: $[st_0, \alpha_1, st_1, \alpha_2, \ldots, \alpha_n, st_n]$ unde $\alpha_i \in RunEvent$ și $st_i = \langle\langle AKN_i, F_i \rangle\rangle$

- Urmă (*trace*): $[\alpha_1, \alpha_2, \ldots, \alpha_n]$

- Știind starea inițială, putem defini execuția protocolului folosind urmele.

  Pentru un protocol P, definim $traces(P)$ ca fiind mulțimea urmelor sistemului de tranziții etichetate care definește semantica operațională a lui P.

Exemplu: urmă (trace) pentru protocolul Needham-Schroeder

$((1, \rho, \emptyset), \mathit{create}(i))$

$((1, \rho, \emptyset), \mathit{send}_1(i, r, \{\!| ni, i |\!\}_{pk(r)}))$

Exemplu: urmă (trace) pentru protocolul Needham-Schroeder

$((1, \rho, \emptyset), \mathit{create}(i))$

$((1, \rho, \emptyset), \mathit{send}_1(i, r, \{\!| ni, i |\!\}_{pk(r)}))$

$((2, \rho, \emptyset), \mathit{create}(r))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), \mathit{recv}_1(i, r, \{\!| W, i |\!\}_{pk(r)}))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), \mathit{send}_2(r, i, \{\!| W, nr |\!\}_{pk(i)}))$

Exemplu: urmă (trace) pentru protocolul Needham-Schroeder

$((1, \rho, \emptyset), create(i))$

$((1, \rho, \emptyset), send_1(i, r, \{\!|\, ni, i \,|\!\}_{pk(r)}))$

$((2, \rho, \emptyset), create(r))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), recv_1(i, r, \{\!|\, W, i \,|\!\}_{pk(r)}))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), send_2(r, i, \{\!|\, W, nr \,|\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), recv_2(r, i, \{\!|\, ni, V \,|\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), send_3(i, r, \{\!|\, V \,|\!\}_{pk(r)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), claim_4(i, synch))$

Exemplu: urmă (trace) pentru protocolul Needham-Schroeder

$((1, \rho, \emptyset), create(i))$

$((1, \rho, \emptyset), send_1(i, r, \{\!|\, ni, i \,|\!\}_{pk(r)}))$

$((2, \rho, \emptyset), create(r))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), recv_1(i, r, \{\!|\, W, i \,|\!\}_{pk(r)}))$

$((2, \rho, \{W \mapsto ni^{\#1}\}), send_2(r, i, \{\!|\, W, nr \,|\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), recv_2(r, i, \{\!|\, ni, V \,|\!\}_{pk(i)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), send_3(i, r, \{\!|\, V \,|\!\}_{pk(r)}))$

$((1, \rho, \{V \mapsto nr^{\#2}\}), claim_4(i, synch))$

$((2, \rho, \emptyset), recv_3(i, r, \{\!|\, nr \,|\!\}_{pk(r)}))$

$((2, \rho, \emptyset), claim_5(r, synch))$

Semantica operațională: tranziții

$$(State, RunEvent, \rightarrow, st_0(P))$$

- Sistemul de tranziții are patru reguli, corespunzătoare evenimentelor: *create*, *send*, *recv*, *claim*

Semantica operațională: tranziții

$$(State, RunEvent, \rightarrow, st_0(P))$$

- Sistemul de tranziții are patru reguli, corespunzătoare evenimentelor: *create*, *send*, *recv*, *claim*

$$[create_P]\ \frac{R \in dom(P)\ \ ((\theta,\rho,\emptyset),s) \in runsof(P,R)\ \ \theta \notin runsIDs(F)}{\langle\!\langle AKN, F\rangle\!\rangle \xrightarrow{((\theta,\rho,\emptyset),create(R))} \langle\!\langle AKN, F \cup \{((\theta,\rho,\emptyset),s)\}\rangle\!\rangle}$$

Semantica operațională: tranziții

$$[send]\ \frac{e = send_l(R_1, R_2, m)\ \ (inst, [e]\cdot s) \in F}{\langle\!\langle AKN, F\rangle\!\rangle \xrightarrow{(inst,e)} \langle\!\langle AKN \cup \{inst(m)\}, F \setminus \{(inst, [e]\cdot s)\} \cup \{(inst, s)\}\rangle\!\rangle}$$

Semantica operațională: tranziții

$$[send]\ \frac{e = send_l(R_1, R_2, m)\ \ (inst, [e] \cdot s) \in F}{\langle\!\langle AKN, F \rangle\!\rangle \xrightarrow{(inst, e)} \langle\!\langle AKN \cup \{inst(m)\}, F \setminus \{(inst, [e] \cdot s)\} \cup \{(inst, s)\} \rangle\!\rangle}$$

$$[claim]\ \frac{e = claim_l(R, c, t)\ \ (inst, [e] \cdot s) \in F}{\langle\!\langle AKN, F \rangle\!\rangle \xrightarrow{(inst, e)} \langle\!\langle AKN, F \setminus \{(inst, [e] \cdot s)\} \cup \{(inst, s)\} \rangle\!\rangle}$$

Semantica operațională: tranziții

$$[recv]\ \frac{e = recv_l(R_1, R_2, pt)\quad AKN \vdash m\quad (inst, [e] \cdot s) \in F\quad Match(inst, pt, m, inst')}{\langle\!\langle AKN, F \rangle\!\rangle \xrightarrow{(inst', e)} \langle\!\langle AKN, F \setminus \{(inst, [e] \cdot s)\} \cup \{(inst', s)\} \rangle\!\rangle}$$

Semantica operațională: tranziții

$$[recv]\ \frac{e = recv_l(R_1, R_2, pt)\quad AKN \vdash m\quad (inst, [e]\cdot s) \in F\quad Match(inst, pt, m, inst')}{\langle\!\langle AKN, F\rangle\!\rangle \xrightarrow{(inst', e)} \langle\!\langle AKN, F \setminus \{(inst, [e]\cdot s)\} \cup \{(inst', s)\}\rangle\!\rangle}$$

- $Match(inst, pt, m, inst')$ este adevărată ddacă
  - $inst = (\theta, \rho, \sigma)$, $inst' = (\theta', \rho', \sigma')$
  - $inst(pt) = m$, $pt \in RoleTerm$, $m \in RunTerm$
  - $dom(\sigma') = dom(\sigma) \cup vars(pt)$
  - $\sigma'(v) \in type(v)$ oricare $v \in dom(\sigma')$,

unde $vars(pt)$ este mulțimea variabilelor din Var care apar în pt, iar $type(v)$ depinde de modelarea agenților.

Exemplu: *Match*

Definim $type(V) \in \{S_1, S_2, S_3, S_4, S_5\}$ unde

$$
\begin{array}{lcl}
S_1 & ::= & Agent \\
S_2 & ::= & Func\,(RunTerm^*) \\
S_3 & ::= & Fresh \mid AdversaryFresh \\
S_4 & ::= & sk(RunTerm) \mid pk(RunTerm) \\
S_5 & ::= & k(RunTerm, RunTerm)
\end{array}
$$

Exemplu: *Match*

Definim $type(V) \in \{S_1, S_2, S_3, S_4, S_5\}$ unde

$$\begin{array}{lcl} S_1 & ::= & Agent \\ S_2 & ::= & Func\,(RunTerm^*) \\ S_3 & ::= & Fresh|\,AdversaryFresh \\ S_4 & ::= & sk(RunTerm)\,|\,pk(RunTerm) \\ S_5 & ::= & k(RunTerm, RunTerm) \end{array}$$

Dacă $type(X) = S_5$ atunci

- $Match((1, \rho, \emptyset), X, nr^{\#2}, (1, \rho, \{X \mapsto nr^{\#2}\})$

Exemplu: *Match*

Definim $type(V) \in \{S_1, S_2, S_3, S_4, S_5\}$ unde

$$\begin{array}{lll} S_1 & ::= & Agent \\ S_2 & ::= & Func\,(RunTerm^*) \\ S_3 & ::= & Fresh |\, AdversaryFresh \\ S_4 & ::= & sk(RunTerm) \,|\, pk(RunTerm) \\ S_5 & ::= & k(RunTerm, RunTerm) \end{array}$$

Dacă $type(X) = S_5$ atunci

- $Match((1, \rho, \emptyset), X, nr^{\#2}, (1, \rho, \{X \mapsto nr^{\#2}\})$
- $\neg\ Match((1, \rho, \emptyset), nr, nr^{\#2}, inst')$

Exemplu: *Match*

Definim $type(V) \in \{S_1, S_2, S_3, S_4, S_5\}$ unde

$$\begin{array}{lcl} S_1 & ::= & Agent \\ S_2 & ::= & Func\,(RunTerm^*) \\ S_3 & ::= & Fresh \mid AdversaryFresh \\ S_4 & ::= & sk(RunTerm) \mid pk(RunTerm) \\ S_5 & ::= & k(RunTerm, RunTerm) \end{array}$$

Dacă $type(X) = S_5$ atunci

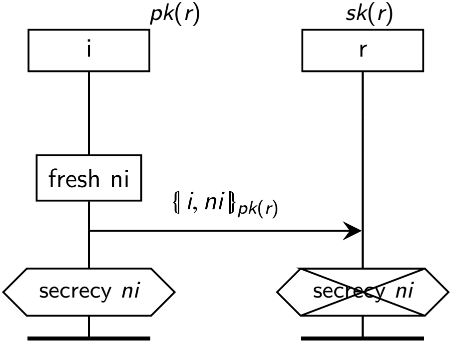
- $Match((1, \rho, \emptyset), X, nr^{\#2}, (1, \rho, \{X \mapsto nr^{\#2}\})$
- $\neg\ Match((1, \rho, \emptyset), nr, nr^{\#2}, inst')$
- $\neg\ Match((1, \rho, \emptyset), X, (nr^{\#1}, nr^{\#2}), inst')$

Proprietăți de securitate

Mesaje secrete (*secrecy*)

Mesaje secrete (*secrecy*)

Exemplu: protocolul OSS (One-Sided Secrecy)



Agenți onești și agenți corupți

- În continuare vom presupune că agenții sunt împărțiți în *corupți* și *onești*.

  $Agent = Agent_H \cup Agent_C$

Agenți onești și agenți corupți

- În continuare vom presupune că agenții sunt împărțiți în *corupți* și *onești*.

  $Agent = Agent_H \cup Agent_C$

- Pentru $(\theta, \rho, \sigma) \in Inst$ definim

  $$honest(\theta, \rho, \sigma) \text{ ddacă } ran(\rho) \subseteq Agent_H$$

  (rolurile sunt instanțiate cu agenți onești).

Agenți onești și agenți corupți

- În continuare vom presupune că agenții sunt împărțiți în *corupți* și *onești*.

  $Agent = Agent_H \cup Agent_C$

- Pentru $(\theta, \rho, \sigma) \in Inst$ definim

  $$honest(\theta, \rho, \sigma) \text{ ddacă } ran(\rho) \subseteq Agent_H$$

  (rolurile sunt instanțiate cu agenți onești).

- Agenții corupți împărtășesc cunoștințe cu adversarul:

  $$AKN_0(P) \supseteq \bigcup_{\substack{R \in Role \\ \rho(R) \in Agent_C}} \{(\theta, \rho, \sigma)(rt) \mid rt \in KN_0(R), rt \text{ nu are subtermeni } fresh\}$$

Proprietăți de securitate: *secrecy*

Fie P un protocol și R un rol.

- Vom spune că proprietatea $\gamma = \mathit{claim}_l(R, \mathit{secret}, \mathit{rt})$ este corectă dacă

  pentru orice $t \in \mathit{traces}(P)$ și oricare $((\theta, \rho, \sigma), \gamma) \in t$

  $$\mathit{honest}((\theta, \rho, \sigma)) \text{ implică } \mathit{AKT}(t) \nvdash (\theta, \rho, \sigma)(\mathit{rt})$$

Proprietăți de securitate: *secrecy*

Fie P un protocol și R un rol.

- Vom spune că proprietatea $\gamma = claim_l(R, secret, rt)$ este corectă dacă

  pentru orice $t \in traces(P)$ și oricare $((\theta, \rho, \sigma), \gamma) \in t$

  $$honest((\theta, \rho, \sigma)) \text{ implică } AKT(t) \nvdash (\theta, \rho, \sigma)(rt)$$

  Informal: pentru toate urmele de execuție care conțin γ, daca agenții sunt onești atunci termenul care se dorește a fi secret nu este deductibil din cunoștinele adversarului.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!| i, ni |\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!| i, W |\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Fie $\theta \in RID$ și $\rho = \{i \mapsto A, r \mapsto B\}$ cu $A, B \in Agent_H$.

Considerăm următoarea urmă:

$$t = [((\theta, \rho, \emptyset), create(r)), ((\theta, \rho, \{W \mapsto ne\}), recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)})) ((\theta, \rho, \{W \mapsto ne\}), claim_3(r, secret, W))]$$

unde $ne \in AdversaryFresh \subseteq AKN_0(P)$.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!\!\{ i, ni \}\!\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!\!\{ i, W \}\!\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Fie $\theta \in RID$ și $\rho = \{i \mapsto A, r \mapsto B\}$ cu $A, B \in Agent_H$.

Considerăm următoarea urmă:

$$\begin{aligned} t = \ & [((\theta, \rho, \emptyset), create(r)), \\ & ((\theta, \rho, \{W \mapsto ne\}), recv_1(i, r, \{\!\!\{ i, W \}\!\!\}_{pk(r)})) \\ & ((\theta, \rho, \{W \mapsto ne\}), claim_3(r, secret, W))] \end{aligned}$$

unde $ne \in AdversaryFresh \subseteq AKN_0(P)$.

Ce observați?

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Fie $\theta \in RID$ și $\rho = \{i \mapsto A, r \mapsto B\}$ cu $A, B \in Agent_H$.

Considerăm următoarea urmă:

$$\begin{aligned} t = \; & [((\theta, \rho, \emptyset), create(r)), \\ & ((\theta, \rho, \{W \mapsto ne\}), recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)})) \\ & ((\theta, \rho, \{W \mapsto ne\}), claim_3(r, secret, W))] \end{aligned}$$

unde $ne \in AdversaryFresh \subseteq AKN_0(P)$.

Dacă $\gamma = claim_3(r, secret, W)$ atunci $((\theta, \rho, \{W \mapsto ne\}), \gamma) \in t$ și $honest(\theta, \rho, \{W \mapsto ne\})$ dar $AKN(t) \vdash ne = (\theta, \rho, \{W \mapsto ne\})(W)$

Exemplu: protocolul OSS

$$OSS(i) = \begin{array}{l}(\{i, r, ni, pk(r)\}, \\ [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), \\ \ claim_2(i, secret, ni)])\end{array} \qquad OSS(r) = \begin{array}{l}(\{i, r, sk(r)\}, \\ [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), \\ \ claim_3(r, secret, W)])\end{array}$$

Fie $\theta \in RID$ și $\rho = \{i \mapsto A, r \mapsto B\}$ cu $A, B \in Agent_H$.

Considerăm următoarea urmă:

$$t = \begin{array}{l}[((\theta, \rho, \emptyset), create(r)), \\ ((\theta, \rho, \{W \mapsto ne\}), recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)})) \\ \ ((\theta, \rho, \{W \mapsto ne\}), claim_3(r, secret, W))]\end{array}$$

unde $ne \in AdversaryFresh \subseteq AKN_0(P)$.

Dacă $\gamma = claim_3(r, secret, W)$ atunci $((\theta, \rho, \{W \mapsto ne\}), \gamma) \in t$ și $honest(\theta, \rho, \{W \mapsto ne\})$ dar $AKN(t) \vdash ne = (\theta, \rho, \{W \mapsto ne\})(W)$

În consecință, proprietatea *secrecy pentru rolul r* nu este corectă.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Fie $\theta \in RID$ și $\rho = \{i \mapsto A, r \mapsto B\}$ cu $A, B \in Agent_H$.

Considerăm următoarea urmă:

$$\begin{aligned} t = \; & [((\theta, \rho, \emptyset), create(r)), \\ & ((\theta, \rho, \{W \mapsto ne\}), recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)})) \\ & ((\theta, \rho, \{W \mapsto ne\}), claim_3(r, secret, W))] \end{aligned}$$

unde $ne \in AdversaryFresh \subseteq AKN_0(P)$.

Dacă $\gamma = claim_3(r, secret, W)$ atunci $((\theta, \rho, \{W \mapsto ne\}), \gamma) \in t$ și $honest(\theta, \rho, \{W \mapsto ne\})$ dar $AKN(t) \vdash ne = (\theta, \rho, \{W \mapsto ne\})(W)$

În consecință, proprietatea *secrecy pentru rolul r* nu este corectă.

Rețineți caracterul *local* al raționamentului!

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Exemplu: protocolul OSS

$$OSS(i) = \begin{array}{l}(\{i, r, ni, pk(r)\},\\ [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}),\\ \ claim_2(i, secret, ni)])\end{array} \qquad OSS(r) = \begin{array}{l}(\{i, r, sk(r)\},\\ [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}),\\ \ claim_3(r, secret, W)])\end{array}$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!\!\{ i, ni \}\!\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!\!\{ i, W \}\!\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Atunci există cel mic k astfel încât $AKN_k \nvdash inst(ni)$ și $AKN_{k+1} \vdash inst(ni)$.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Atunci există cel mic k astfel încât $AKN_k \nvdash inst(ni)$ și $AKN_{k+1} \vdash inst(ni)$.
Observăm că sigura regulă care îmbogățe cunoștințele adversarului este *send*, deci

pentru a trece de la pasul k la pasul $k+1$ a fost aplicată regula *send*.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!| i, ni |\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!| i, W |\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Atunci există cel mic k astfel încât $AKN_k \nvdash inst(ni)$ și $AKN_{k+1} \vdash inst(ni)$.
Observăm că sigura regulă care îmbogățe cunoștințele adversarului este *send*, deci

pentru a trece de la pasul k la pasul $k+1$ a fost aplicată regula *send*.

Rezultă că $AKN_{k+1} = AKN_k \cup \{inst(\{\!| i, ni |\!\}_{pk(r)})\} = AKN_k \cup \{\{\!| \rho(i), ni^{\#\theta} |\!\}_{pk(\rho(r))}\}$.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!| i, ni |\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!| i, W |\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Atunci există cel mic k astfel încât $AKN_k \nvdash inst(ni)$ și $AKN_{k+1} \vdash inst(ni)$.
Observăm că sigura regulă care îmbogățe cunoștințele adversarului este *send*, deci

pentru a trece de la pasul k la pasul $k+1$ a fost aplicată regula *send*.

Rezultă că $AKN_{k+1} = AKN_k \cup \{inst(\{\!| i, ni |\!\}_{pk(r)})\} = AKN_k \cup \{\{\!| \rho(i), ni^{\#\theta} |\!\}_{pk(\rho(r))}\}$.

În consecință $AKN_k \cup \{\{\!| \rho(i), ni^{\#\theta} |\!\}_{pk(\rho(r))}\} \vdash ni^{\#\theta}$, dar acest lucru nu este posibil deoarece $\rho(r)$ este onest.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!|\, i, ni \,|\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!|\, i, W \,|\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Atunci există cel mic k astfel încât $AKN_k \nvdash inst(ni)$ și $AKN_{k+1} \vdash inst(ni)$.
Observăm că sigura regulă care îmbogățe cunoștințele adversarului este *send*, deci

pentru a trece de la pasul k la pasul $k+1$ a fost aplicată regula *send*.

Rezultă că $AKN_{k+1} = AKN_k \cup \{inst(\{\!|\, i, ni \,|\!\}_{pk(r)})\} = AKN_k \cup \{\{\!|\, \rho(i), ni^{\#\theta} \,|\!\}_{pk(\rho(r))}\}$.

În consecință $AKN_k \cup \{\{\!|\, \rho(i), ni^{\#\theta} \,|\!\}_{pk(\rho(r))}\} \vdash ni^{\#\theta}$, dar acest lucru nu este posibil deoarece $\rho(r)$ este onest.

Am obținut o contradicție, deci (*) este falsă.

Exemplu: protocolul OSS

$$OSS(i) = (\{i, r, ni, pk(r)\}, [send_1(i, r, \{\!| i, ni |\!\}_{pk(r)}), claim_2(i, secret, ni)])$$

$$OSS(r) = (\{i, r, sk(r)\}, [recv_1(i, r, \{\!| i, W |\!\}_{pk(r)}), claim_3(r, secret, W)])$$

Demonstrăm că proprietatea $\zeta = claim_2(i, secret, ni)$ este corectă.

Fie $t \in traces(OSS)$ și $inst = (\theta, \rho, \sigma) \in Inst$ astfel încât $(inst, \zeta) \in t$ și $honest(inst)$.

(*) Presupunem că $AKN(t) \vdash inst(ni) = ni^{\#\theta}$.

Atunci există cel mic k astfel încât $AKN_k \nvdash inst(ni)$ și $AKN_{k+1} \vdash inst(ni)$.
Observăm că sigura regulă care îmbogăţe cunoștinţele adversarului este *send*, deci

pentru a trece de la pasul k la pasul $k+1$ a fost aplicată regula *send*.

Rezultă că $AKN_{k+1} = AKN_k \cup \{inst(\{\!| i, ni |\!\}_{pk(r)})\} = AKN_k \cup \{\{\!| \rho(i), ni^{\#\theta} |\!\}_{pk(\rho(r))}\}$.

În consecinţă $AKN_k \cup \{\{\!| \rho(i), ni^{\#\theta} |\!\}_{pk(\rho(r))}\} \vdash ni^{\#\theta}$, dar acest lucru nu este posibil deoarece $\rho(r)$ este onest.

Am obţinut o contradicţie, deci (*) este falsă.

În consecinţă, proprietatea *secrecy pentru rolul r* este corectă.

Bibliografie

- Cremers, C. J. F. (2006). Scyther : semantics and verification of security protocols Eindhoven: Technische Universiteit Eindhoven DOI: 10.6100/IR614943
- Cremers C. and Mauw S. Operational Semantics and Verification of Security Protocols. Springer, 2012.