The Spi Clalculus

Rafał Linek
nr albumu 181272

27 maja 2002

1 Wstęp

Autorami **rachunku SPI** są Martin Abadi i Andrew D. Gordon. **Rachunek SPI** jest zaprojektowany do opisywania i analizowania protokołu ochorny takich jak autentykacja. Protokół jest oparty na kanałach komunikacyjnych z takimi właściwościami jak autentykacja i prywatność. (Są one tworzone i używane na przykład do autentykacji między serwerami i klientami.).
Ponieważ semantyka **rachunku spi** jest precyzyjna a zarazem bardzo „inteligentna" mamy dobrą podstawę do analizowania protokołów. Możemy wyrazić gwarancję ochrony jako równoważność pomiędzy procesami w tym rachunku (np. możemy wyrazić że protokół zachowuje tajny kawałek danych X poprzez powiedzenie, że protokół z X jest równoważny protokołowi X' dla każdego X'). Ta równoważność znaczy równoważność w oczach konkretnego środowiska, który moze być w interakcji z protokołem. Niemniej bardzo trudno tą równoważność udowodnić.

2 Rachunek PI

Rachunek PI jest małym, ale bardzo silnym jezykiem programowania. Może on być podstawą do obliczeń równoległych jak rachunek lambda dla obliczeń sekwentów.
Rachunek PI jest systemem niezależnych, równoległych procesów, które synchronizują przesyłanie wiadomości na nazwanych kanałach. Kanały moga być ograniczone (zastrzeżone) tak, że tylko wybrane procesy mogą się przez nie komunikować. Zaletą **rachunku PI** jest to, że ograniczenie kanałów może się zmieniać w trakcie wykony-wania obliczeń (procesy są ruchliwe). Kiedy proces wysyła wiadomość poprzez ograniczony kanał do procesu nie należącego do tego kanału, to mówimy, że ograniczenie jest łamane. To znaczy, że powiększane jest ograniczenie o proces odbierający na kanale. Tak więc, kanały są elastyczna mozliwością komunikacji.
W **rachunku pi** mamy nieskończony zbiór nazw (m, n , p, q , r) używanych do komunikacji między kanałami i nieskończony zbiór zmiennych (x, y, z).
Zbiór termów definiujemy jako gramatykę:
$L, M, N(termy) ::=$
$n-$nazwy
$(M, N)-$pary
$0-$zero
$suc(M)-$następnik
$x-$zmienne
Zbiór procesów definiujemy jako gramatykę:
$P, Q, R(procesy) ::=$
$\overline{M} < N > .P-$wyjście

$M(x).P-$wejście
$P|Q-$kompozycja
$(vn)P-$ograniczenie
$!P-$replikacja
$(M\ is\ N)P-$dopasowanie
$0-$nil
$let(x, y) = MinP-$rozdzielenie par
$case\ M\ of\ 0 : P\ suc(x) : Q-$wybór liczb

W $(vn)P$ n jest związane w P. W $M(x).P$, zmienna x jest zawiązana w P.
W $let\ (x, y) = M\ in\ P$, zmienne x i y są związane w P.
W $case\ M\ of\ 0 : P\ suc(x) : Q$ zmienna x jest związana z drugą gałezią–Q.
Piszemy $P(M/x)$ do wyrażenia zastąpienia każdego wolnego wystąpienia x w procesie P przez term M.
Dodatkowo używamy $\overline{M} < N >$ dla ozanaczenia $\overline{M} < n > .0$.
Właściwosci:

- Podstawową własnością jest interakcja, termem N jest przesyłany pomiędzy wyjściowym procesem a procesem wejściowym poprzez nazwany kanał m.
    - proces wyjściowy $\overline{m} < N > .P$ jest gotowy na wyjściu kanału m; po udanej interakcji term N jest przesyłany kanałem m i proces P startuje.
    - proces wejsciowy $m(x).P$ jest gotowy na wejściu kanału m; po udanej interakcji w której N jest przesyłany przez kanał m, proces $P(N/x)$ startuje.
- Kompozycja $P|Q$ to równoległe wykonywanie procesów P i Q. Interakcja na kanałach znanych dla obydwu procesów odbywa się niezależnie.
- Ograniczenie $(vn)P$ oznacza, że proces tworzy nową, prywatną nazwę n i zachowuje się jak P.
- Replikacja $!P$ zachowuje się jak nieskończona liczba kopii P działających równolegle.
- Dopasowanie $(M\ is\ N)P$ zachowuje się jak P przy założeniu, że termy M i N są takie same. Wpp zostaje zablokowany i nic nie robi.
- Nil to jest proces 0 nie robi nic.
  Kolejne rzeczy są dodane dla naszych potrzeb (nie są w standardzie pi)
- Rozdzielenie par to proces $let(x, y) = M\ in\ P$ zachowywujący się jak $P(N/x)(L/y)$ gdzie term M jest parą (N, L). Wpp proces jest zablokowany.

- Wybór liczb to proces $case\ M\ of\ 0 : P\ suc(x) : Q$ zachowywujacy się jak P gdy term M jest 0, lub jak $Q(N/x)$ gdy M jest $suc(N)$. Wpp proces jest zablokowany.

Piszemy $P \simeq Q$ aby wyrazić, że zachowanie procesu P i Q są nierozróżnialne. Wewnętrzne struktury P i Q mogą być rózne, ale trzeci proces R nie może rozróżnić równoległego wykonania z P z równoległym wykonaniem z Q.

2.1 Przykład

Są dwa procesy A i B, które dzielą kanał c_{AB}. Tylko A i B mogą wysyłać dane lub słuchać na kanle. W tym protokole A używa c_{AB} do wysyłania pojedyńczej wiadomości M do B.
Nieformalnie:
Wiadomość 1 $A \rightarrow B : M$ na c_{AB}
W **rachunku pi**:

$A(M) \doteq \overline{c_{AB}} < M >$
$B \doteq c_{AB}(x).0$
$Inst(M) \doteq (vc_{AB})(A(M)|B)$

Procesy $A(M)$ i B oznaczają dwie instancje procesów. $Inst(M)$ określa cały protokół. Kanał c_{AB} jest ograniczony, w ten sposób, że tylko A i B mogą z niego korzystać. W tej definicji $A(M)$ i $Inst(M)$ są procesami zparametryzowanymi M.
Bardziej użytecznym przykładem jest, gdy B startuje proces F z wejścia.

$A(M) \doteq \overline{c_{AB}} < M >$
$B \doteq c_{AB}(x).F(x)$
$Inst(M) \doteq (vc_{AB})(A(M)|B)$

Nieformalnie $F(x)$ jest rezultatem zastosowania F do x. Formalniej F jest abstrakcją i $F(x)$ jest instancją tej abstrakcji.

- autentykacja: B zawsze stosuje F do wiadomości M którą A wysyła; atakujący nie może spowodować, żeby B zastosowało F do innej wiadomości;

- sekretność: wiadomość M nie może być czytana podczas transmisji z A do B; gdy F nie ujawnia M, to cały protokół nie ujawnia M.

Własność tajności może być stwierdzona w równoważnosci termów.
Gdy $F(M) \doteq F(M')$ dla każdego M i M', wtedy $Inst(M) \doteq Inst(M')$. To znaczy, że gdy $F(M)$ jest nierozróżnialne z $F(M')$ wtedy protokół z wiadomością M jest

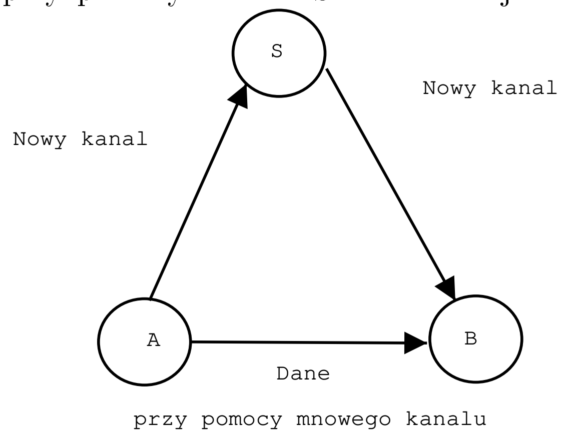
nierozróżnialny z protokołem z waidomością M'.
Autentykację zapisujemy jako równoważność:

$A(M) \doteq \overline{c_{AB}} < M >$
$B_{spec}(M) \doteq c_{AB}(x).F(M)$
$Inst_{spec}(M) \doteq (\nu c_{AB})(A(M)|B_{spec}(M))$

A jest takie jak zwykle. B zaś jest zastapione wersją jako $B_{spec}(M)$. Otrzymuje on wejście z A i wtedy działa jak B z otrzymanym M. Mówimy, że jest on „magiczną” wersją $B_{spec}(M)$, podobnie jak jest to z $Inst_{spec}(M)$. Stąd autentykacja wygląda nastepująco $Inst(M) \doteq Inst_{spec}(M)$ dla każdego M.
Z autentyczności i tajności wynika, że nawet aktywny atakujący bedący w środowisku nie rozróżni dwóch procesów.

2.2 Następny przykład

W tym przykładzie przed wysyłaniem wiadomości tworzony i przesyłany jest kanał przy pomocy serwera S. Protokół jest oparty na protokole „WideMouthedFrog”.



Nieformalnie:
Wiadomość 1 $A \rightarrow S : c_{AB}$ na c_{AS}
Wiadomość 2 $S \rightarrow B : c_{AB}$ na c_{SB}
Wiadomość 3 $A \rightarrow B : M$ na c_{AB}
Kanał c_{AB} jest tworzony przez A. S nie używa kanału tylko go transmituje.

$A(M) \doteq (\nu c_{AB})\overline{c_{AS}} < c_{AB} > .\overline{c_{AB}} < M >$
$S \doteq c_{AS}(x).\overline{c_{SB}} < x >$
$B \doteq c_{SB}.x(y).F(y)$
$Inst(M) \doteq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B)$

Piszemy $F(y)$ mając na myśli, że reprezentuje to co B z wiadomością. (jak w

poprzednim przykładzie). Ograniczenie na kanale c_{AS}, c_{SB} i c_{AB} odzwierciedla oczekiwaną gwarancje prywatnosci na tych kanałach. A generuje nowy kanał c_{AB} i wysyła go poza zasięgiem ograniczenia do B, przez S.
Autentykacja:

$A(M) \doteq (vc_{AB})\overline{c_{AS}}.(v\overline{c_{AB}} < M >$
$S \doteq c_{AS}(x).\overline{c_{SB}} < x >$
$B_{spec}(M) \doteq c_{SB}.x(y).F(M)$
$Inst_{spec}(M) \doteq (vc_{AS})(vc_{sb})(A(M) \mid S \mid B_{spec}(M))$

Warunki autentyczności i tajności są tak samo określone jak w poprzednim przykładzie.

3 Rachunek Spi z dzielonym kluczem

Do **rachunku pi** dodajemy konstrukcje związane z kryptografią.
Zbiór termów jak w **rachuku pi**. Dodatkowo:
M_N-dzielony klucz odkodowywania
Zbiór procesów jak w **rachuku pi**. Dodatkowo:
$case\ L\ of\ x_N\ in\ P-$dzielony klucz kodowania

Zmienna x jest związana z P.

- term M_N reprezentuje zaszyfrowany text termu M kluczem N, używającym dzielonego klucza kryptosystemu takiego jak DES.
- proces $case\ L\ of\ x_N\ in\ P$ usiłuje odszyfrować term L, kluczem N. Gdy L jest zakodowanym tekstem z M_N wtedy proces zachowuje sie jak $P(M/x)$. Wpp blokuje się.

Semantyka **rachunku spi** może być taka jak w **rachunku pi**. Interesującą jest równoważność. Znów piszemy $P \doteq Q$ w rozumieniu, że zachowanie procesów P i Q jest nierozróżnialne. Jednak teraz jest to bardziej skomplikowane.
$P(M) \doteq (vK)\underline{c} < M_K >$
Proces wysyła M przy użyciu nowego klucza K poprzez publiczny kanał c. Klucz nie jest przesyłany. Intuicyjnie $P(M)$ i $P(M')$ są nierozróżnialne dla każdych M i M', ponieważ obserwator nie może odkryć K i stąd nie może nic powiedzieć o M i o M'. Z drugiej strony $P(M)$ i $P(M')$ są zupełnie różne, ponieważ transmitują inna wiadomośc poprzez c.

3.1 Przykład

Dwa procesy A i B dzielą klucz K_{AB}. Jest kanał c_{AB}, który jednak teraz nie jest chroniony. A wysyła wiadomośc M przy użyciu K_{AB} do B przy pomocy kanału c_{AB}.
Nieformalnie:

Wiadomość 1 $A \rightarrow B : M_{K_{AB}}$ na c_{AB}

W **rachunku spi**:

$A(M) \doteq \overline{c_{AB}} < M_{K_{AB}} >$
$B \doteq c_{AB}(x).case\ x\ of\ y_{K_{AB}}\ in\ F(y)$
$Inst(M) \doteq (vK_{AS})(A(M)|B)$

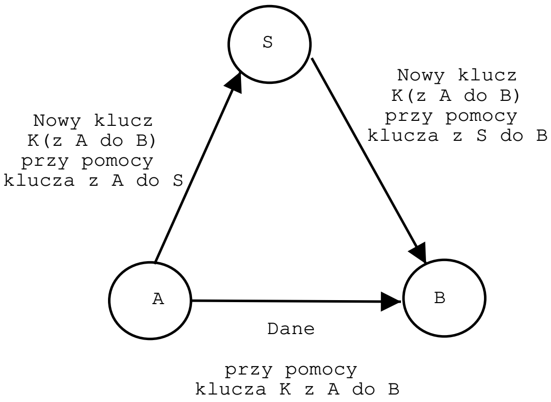
A wysyła $M_{K_{AB}}$ na c_{AB}, podczas gdy B słucha na c_{AB}. Dostawszy wiadomość B odkodowywuje ją przy użyciu K_{AB}. Gdy się to uda B stosuje F do rezultatu. Założenie, że A i B dzielą K_{AB} stwarza ryzyko, że ograniczenie na K_{AB} jest legalne mimo, że K_{AB} nie używa kanału. Z drugiej strony c_{AB} nie jest ograniczone gdyż jest publicznym kanałem. Można wysłać inne wiadomości kodowane innym kluczem, ale w tym przypadku spowoduje to zablokowanie B.
Specyfikacja:

$A(M) \doteq \overline{c_{AB}} < M_{K_{AB}} >$
$B_{spec}(M) \doteq c_{AB}(x).case\ x\ of\ y_{K_{AB}}\ inF(M)$
$Inst_{spec}(M) \doteq (vK_{AS})(A(M)|B_{spec}(M))$

Autentykacja i tajność jak w poprzednich przykładach.

3.2 Przykład z zakładaniem klucza

Znów korzystamy tu z protokołu „Wide Mounthed Frog”. A i B dzielą klucze K_{AS} i K_{SB} związane z serwerem S. Kiedy A i B chcą sie komunikować, tworzy nowy klucz K_{AB} i wysyła do serwera poprzez K_{AS}, który transmituje go do B poprzez K_{SB}. Dopóki cała komunikacja jest chroniona kodowaniem, może być używana przez pub-



liczne kanały.
Nieformalnie:

Wiadomość 1 $A \to S : K_{AB K_{AS}}$ na c_{AS}
Wiadomość 2 $S \to B : K_{AB K_{SB}}$ na c_{SB}
Wiadomość 3 $A \to B : M_{K_{AB}}$ na c_{AB}

W **rachunku spi**:

$A(M) \doteq (vK_{AB})(\overline{c_{AS}} < K_{AB K_{AS}} > .\overline{c_{AB}} < M_{K_{AB}} >)$
$S \doteq c_{AS}(x).case\ x\ of\ y_{K_{AS}}\ in\ \overline{c_{SB}} < y_{K_{SB}} >$
$B \doteq c_{SB}(x).case\ x\ of\ y_{K_{SB}}\ inc_{AB}(z).case\ z\ of\ w_y\ in\ F(w)$
$Inst(M) \doteq (vK_{AS})(vK_{SB})(A(M)\ S \mid B)$

gdzie proces $F(w)$ rpeprezentuje pozostałośc po zachowaniu B po odebraniu wiadomosci w. A genetruje klucz K_{AB} i wysyła go poza zasięgiem B do S.
Specyfikacja:

$A(M) \doteq (vK_{AB})(\overline{c_{AS}} < K_{AB K_{AS}} > .\overline{c_{AB}} < M_{K_{AB}} >)$
$S \doteq c_{AS}(x).case\ x\ of\ y_{K_{AS}}\ in\ \overline{c_{SB}} < y_{K_{SB}} >$
$B_{spec} \doteq c_{SB}(x).case\ x\ of\ y_{K_{SB}}\ inc_{AB}(z).case\ z\ of\ w_y\ in\ F(M)$
$Inst_{spec}(M) \doteq (vK_{AS})(vK_{SB})(A(M)\ S \mid B_{spce}(M))$

Autentykacja i tajnośc jak poprzednio.

3.2.1 Przykład z błędem

Tym razem klika razy będziemy chcieli wysyłać różne rzeczy.
Mamy serwer i n różnych procesów. Używamy terów $suc(0), suc(suc(0)), \ldots$ (w skrócie: $\underline{1}, \underline{2}, \ldots$) jako nazwy procesów. Wejściowe kanały są publiczne i maja nazwy: $c_1, c_2, \ldots, c_n$ i c_s. Serwer zawiera pary kluczy z ich procesami. Proces i używa klucza

K_{iS} do wysłania wiadomości do S. Inny klucz k_{Si} jest do uzyskiwania wiadomości z S dla każdego $0 < i < n + 1$.
Nieformalnie:

Wiadomość 1 $A \to S : A, B, K_{AB_{K_{AS}}}$ na c_S
Wiadomość 2 $S \to B : A, K_{AB_{K_{SB}}}$ na c_B
Wiadomość 3 $A \to B : A, M_{K_{AB}}$ na c_B

A i B przebiegają po wszystkich procesach. Trzeba uważać na nazwy, by nie było niejasności. Dopuszczalne jest wykonywanie równoległe.
Przyjmujemy nastepujacą notacje:

$c(x_1, x_2).P \doteq c(y).let\ (x_1, x_2) = y\ in\ P$
$case\ L\ of\ (x_1, x_2)_N\ in\ let\ (x_1, x_2) = y\ in\ P$

Gdzie zmienne y nie są wolne w P. Dla skończonej rodziny procesów $P_1, p_2, \ldots, P_k$ oznaczamy $\prod_{i\epsilon 1..k}$ k-drozną kompozycję $P_1 \mid \ldots\ \mid P_k$. Poza tym opuszczamy nawiasy $(N, N')_{N''}$ u nas zapiszemy jako $N, N'_{N''}$. Instancja I jest trójką (i, j, M), gdzie i i j są adresami źródła i celu, a M wiadomością.

$Send(i, j, M) \doteq (vK)(\overline{c_S} < (\underline{i}, (\underline{j}, K)_{K_{iS}}) > \mid \overline{c_j} < (\underline{i}, M_K) >)$

Porces wysyłający tworzy klucz K i wysła do serwera z nazwami $\underline{i}$ i $\underline{j}$. Wysyałamy dwie wiadomości równolegle.

$Recv(j) \doteq c_j(y_{cipher}).case\ y_{cipher}\ of\ (x_A, x_{key})_{K_{Sj}}\ in\ c_j(z_A, z_{cipher}).(x_A\ is\ z_A)\ case\ z_{cipher}$ $of\ z_{plain_{x_{key}}}\ in\ F(x_A, \underline{j}, z_{plain})$

Odbierający proces czeka na wiadomosc y_{cipher} z serwera i wyciąga klucz x_{key} z wiadomości. Następnie czeka na wiadomość z_{cipher} z tym kluczem i na końcu stosuje F do nazwy x_a od wysłanego, do jego własnej nazwy $\underline{j}$ i do zawartości z_{plain}. Zmienne x_A i z_A są nazwami procesów wysłających.
Serwer jest taki sam dla każdej instancji.

$S \doteq c_s(x_A, x_{cipher}).\prod_{i\epsilon 1..n}(x_A\ is\ \underline{i})\ case\ x_{cipher}\ of\ (x_B, x_{key})_{K_{iS}}$
$in \prod_{j\$epsilon1..n}(x_B\ is\ \underline{j})\overline{c_j} < (x_A, x_{key})_{K_{Sj}} >$

Zmienna x_A oznacza proces wysyłający, x_B odbierający. x_{key} jest nowym kluczem, a x_{cipher} jest kodująca częscią pierwszej wiadomości protokołu. Programujemy tu n-drożne na x_A i x_B użycie kompozycji procesów.
Cały system ma postać:

$Sys(I_1, \ldots, I_m) \doteq (vK_{iS}^-)(vK_{Sj}^-)(Send(I_1) \mid \ldots \mid Send(I_m) \mid !S \mid !Recv(1) \mid \ldots \mid !Recv(n))$

Gdzie $(vK_{iS}^-)(vK_{Sj}^-)$ oznaczają $(vK_{1S})\ldots(vK_{nS})(vK_{S1})\ldots(vK_{Sn})$. Specyfikacja nie jest trudna do określenia. Więc podam tylko jako podsumowanie autentykacje:
$Sys(I_1, \ldots, I_m) \doteq Sys_{spec}(I_1, \ldots, I_m) dlakadejinstancjiI_1, \ldots I_m$

Niestety ten protokół ma wadę. Rozważmy system $Sys(I, I')$ gdzie $I = (i, j, M)$ i $I' = (i, j, M')$. Atakujący może powtażać wiadomości dla jednej instancji i robić zamieszanie w innych instancjach. Spowoduje to podwójne użycie F i powstaną dwie kopie $F(\underline{i}, \underline{j},\ M)$.

3.3 Przykład naprawiony

Rozwiązniem jest użycie stępli czasowych, które w tym rachunku moga być użyte. Autorzy jednak proponują inne rozwiązanie. Nowy protokół wygląda następująco:

Wiadomość 1 $A \to S : A$ na c_S
Wiadomość 2 $S \to A : N_S$ na c_A
Wiadomość 3 $A \to S : A, (A, A, B, K_{AB}, NS)_{K_{AS}}$ na c_S
Wiadomość 4 $A \to B : *$ na c_B
Wiadomość 5 $B \to S : N_B$ na c_S
Wiadomość 6 $S \to B : (S, AB, K_{AB}, N_{B_{K_{SB}})}$ na c_B
Wiadomość 7 $A \to B : A, M_{K_{AB}}$ na c_B

Wiadomości 1 i 2 są odpowiedziami na wezwanie. Wezwaniem jest N_S w chwili stworzonej przez S. Chwila nie jest sekretna, ale musi być nieprzewidywalna. W wiadomości 3, A mówi, że A i B mogą się komunikować poprzez K_{AB} po uzyskaniu N_S. W wiadomości 4, $*$ reprezentuje dowolną ale ustaloną wiadomośc. S używa $*$ do sygnalizacji, że jest gotowe na chwile wezwania N_S z B. W wiadomości 6, S mówi, że A mówi że A i B moga się komunikować przy uzyciu K_{AB}. Wiadomość 7 przesyła dane. Używamy skrótu:
$dla\ k > 1\ (N_1, \ldots,\ N_k, M_{k+1}) \doteq ((N_1, \ldots, N_k), N_{k+1})$
$i\ podobnie\ piszemy\ let\ (x_1, \ldots,\ x_k) = N\ in\ P,\ c(x_1, \ldots,\ x_k).P\ i\ case\ L\ of\ x_1, \ldots,\ x_{k_N}\ in\ P$

W **rachunku SPI**:

$Send(i, j, M) \doteq \overline{c_s} < \underline{i} > \ \mid c_i(x_{nonce}).(vK)(\overline{c_S} < (\underline{i}, (\underline{i}, \underline{i}, \underline{j}, K, x_{nonce})_{K_{iS}}) > \ \mid$
$\overline{c_j} < (\underline{i}, M_K) >)$
$S \doteq c_S(x_A). \prod_{i \epsilon 1..n}(x_A\ is\ \underline{i})$
$(vN_S)(\overline{c_i}(N_S) \mid c_S(x'_A, x_{cipher}).(x'_A\ is\ \underline{i})\ case\ x_{cipher}\ of\ (y_A, z_A, x_B, x_{key}, x_{nonce})_{K_{iS}}\ in$

$\prod_{j\epsilon 1..n}(y_A\ is\ \underline{i})(z_A\ is\ \underline{i})(x_B\ is\ \underline{j})(x_{nonce}\ is\ N_S)(\overline{C_j} < * > \ |$
$c_S(y_{nonce}).\overline{C_j} < (S, \underline{i}, \underline{j}, x_{key}, y_{nonce})_{K_{iS}} >))$
$Recv(j) \doteq c_j(w).(vN_B)(\overline{c_S} < N_B >) \mid c_j(y_{cipher}).case\ y_{cipher}\ of$
$(x_S, x_A, x_B, x_{key}, y_{nonce})_{K_{Sj}}\ in$
$\prod_{i\epsilon 1..n}(x_S\ is\ S)(x_A\ is\ \underline{i})(x_B\ is\ \underline{j})(y_{nonce}\ is N_B)c_j(z_A, z_{cipher}).(z_A\ is\ x_A)$
$case\ z_{cipher}\ of\ z_{plain_{x_{key}}}\ in\ F(\underline{i}, \underline{j}, z_{plain}))$
$Sys(I_1, \ldots, I_m) \doteq (vK_{Si}^-)(vK_{Sj}^-)(Send(I_1) \mid \ldots \mid Send(I_m) \mid !S \mid !Recv(1) \mid \ldots \mid !Recv(n))$

N_S i N_B reprezentują chwile. Dolne indeksy zmiennych wskazują co zmienna powinna reprezentować (np. x_A, x'_A, y_A, z_A są nazwami procesów wysyłających.
Da się pokazać, że:
Autentykacje zapisujemy jako równoważność:

$Sys(I_1, \ldots, I_m) \doteq Sys_{spec}(I_1, \ldots, I_m)$
dla kazdej instancji $I_1, \ldots, I_m$
Tajność zapisujemy jako równoważność:

$Sys(I_1, \ldots, I_m) \doteq Sys(J_1, \ldots, J_m)$
gdy dla każdej pary $(I_1, J_1), \ldots, (I_m, J_m)$ zachodzi nierównoważność