Modeling and Specification of Authentication Protocol (NSLPK) in OTS/CafeOBJ

Lecture Note 06
CafeOBJ Team for JAIST-FSSV2010
Topics

♦ NSLPK, an authentication protocol, and Agreement Property that NSLPK should enjoy.

♦ A specification of NSLPK.

♦ Formalization of Agreement Property.
An authentication protocol that is a revised version (by Gavin Lowe in 1995) of NSPK published by Roger Needham and Michael Schroeder in 1978.

- $n_x$: a nonce (an unguessable random number) made by a principal $x$.
- $\{m\}_{k(x)}$: a ciphertext obtained by encrypting a message (tuple) $m$ with the principal $x$’s public key.
Agreement Property

Whenever a protocol run is successfully completed by $p$ and $q$,
- the principal with which $p$ is communicating is really $q$, and
- the principal with which $q$ is communicating is really $p$
even if there are malicious principals.

For verification of the property,
- the assumptions used are made clear,
- a transition system (an OTS) of NSLPK is made, and
- the property is formalized.
Lowe’s Attack on NSPK

- 17 years passed since it was created till an attack was found by Lowe on NSPK whose difference from NSLPK is only that a Resp message is \( \{ n_p, n_q \}_k(p) \) but not \( \{ n_p, n_q, q \}_k(p) \).

\[
\begin{align*}
p & \rightarrow i \\
\{ n_p, p \}_k(i) & \\
\{ n_p, n_q \}_k(p) & \leftarrow i \\
\{ n_q \}_k(i) & \\
q & \rightarrow i \\
\{ n_p, p \}_k(q) & \\
\{ n_p, n_q \}_k(p) & \\
\{ n_q \}_k(q) & \\
\{ n_q \}_k(q) & \leftarrow i \\
\{ n_p, n_q \}_k(p) & \\
\end{align*}
\]
Assumptions (1)

- There are an arbitrary number of principals, all of which except for one are trustable.

- Each principal is given a pair of public & private keys; the public key is known by all principals, while the private key only by the principal.

- The cryptosystem used is perfect:
  - Ciphertexts can only be decrypted with the corresponding private keys.
  - Nonces (and private keys and the plaintexts of ciphertexts) cannot be guessed.
The behaviors of malicious principals are formalized as the *intruder* proposed by Dolev&Yao; the intruder does the following:

- to glean as much information as possible from the network,
- to fake messages based on the gleaned information, and
- to imitate a trustable principal.
Making an OTS of NSLPK

♦ Formalization of data such as nonces, ciphertexts, etc.

♦ Formalization of the behaviors of NSLPK.
  • To determine what values are observed.
  • Formalization of sending messages exactly obeying the protocol.
  • Formalization of faking messages based on the gleaned information from the network.
Principals & Random Numbers

♦ Module PRINCIPAL:

mod* PRINCIPAL principal-sort Principal {
  [Principal]
  op intruder : -> Principal
  op _=_ : Principal Principal -> Bool {comm}
  eq (P:Principal = P) = true .
  ceq P1:Principal = P2:Principal if P1 = P2 .
}

♦ Module RANDOM:

mod* RANDOM principal-sort Random {
  [Random]
  op _=_ : Random Random -> Bool {comm}
  eq (R:Random = R) = true .
  ceq R1:Random = R2:Random if R1 = R2 .
}
Nonces

♦ Module NONCE: One constructor is declared.

\[
\text{op n : Principal Principal Random } \to \text{ Nonce } \{ \text{constr} \}
\]

✓ \( n(p, q, r) \) denotes a nonce made by \( p \) for \( q \), where \( r \) makes the nonce unique and unguessable.

  - \( n_p \) in \( \{ n_p, p \} \) is denoted by \( n(p, q, r_1) \).
  - \( n_q \) in \( \{ n_p, n_q, q \} \) is denoted by \( n(q, p, r_2) \).

✓ \( p, q, r \) in \( n(p, q, r) \) are meta-information.

✓ The following operators are prepared:

\[
\begin{align*}
\text{eq creator}(n(C, W, R)) &= C . \\
\text{eq forwhom}(n(C, W, R)) &= W . \\
\text{eq random}(n(C, W, R)) &= R . \\
\text{eq } (N1 = N2) &= (\text{creator}(N1) = \text{creator}(N2) \text{ and } \text{forwhom}(N1) = \text{forwhom}(N2) \text{ and } \text{random}(N1) = \text{random}(N2)) .
\end{align*}
\]

Init: \( p \rightarrow q \{ n_p, p \}_{k(q)} \)
Resp: \( q \rightarrow p \{ n_p, n_q, q \}_{k(p)} \)
Ack: \( p \rightarrow q \{ n_q \}_{k(q)} \)
Module **CIPHER1**: One constructor is declared.

\[
\text{op enc1 : Principal Nonce Principal} \\
\qquad \rightarrow \text{Cipher1 \{constr\}}
\]

✓ \(\text{enc1}(p, n, q)\) denotes a ciphertext \(\{n, q\}_{k(p)}\).

✓ The following operators are provided:

\[
\begin{align*}
\text{eq key}(\text{enc1}(K,N,P)) & = K . \\
\text{eq nonce}(\text{enc1}(K,N,P)) & = N . \\
\text{eq principal}(\text{enc1}(K,N,P)) & = P . \\
\text{eq (E11 = E12)} & = (\text{key}(E11) = \text{key}(E12) \text{ and} \\
& \quad \text{nonce}(E11) = \text{nonce}(E12) \text{ and} \\
& \quad \text{principal}(E11) = \text{principal}(E12)) .
\end{align*}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Init:} & p \rightarrow q \quad \{n_p,p\}_{k(q)} \\
\text{Resp:} & q \rightarrow p \quad \{n_p,n_q,q\}_{k(p)} \\
\text{Ack:} & p \rightarrow q \quad \{n_q\}_{k(q)} \\
\hline
\end{array}
\]
Ciphertexts in Resp Messages

♦ Module CIPHER2: One constructor is declared.

\[ \text{op enc2 : Principal Nonce Nonce Principal } \rightarrow \text{Cipher2 } \{\text{constr}\} \]

✓ \( \text{enc2}(p, n_1, n_2, q) \) denotes a ciphertext \( \{n_1, n_2, q\}_{k(p)} \).

✓ The following operators are provided:

\[
\begin{align*}
\text{eq key(enc2(K,N1,N2,P)) = K } . \\
\text{eq nonce1(enc2(K,N1,N2,P)) = N1 .} \\
\text{eq nonce2(enc2(K,N1,N2,P)) = N2 .} \\
\text{eq principal(enc2(K,N1,N2,P)) = P .} \\
\text{eq (E21 = E22) = (key(E21) = key(E22) and} \\
\text{ nonce1(E21) = nonce1(E22) and} \\
\text{ nonce2(E21) = nonce2(E22) and} \\
\text{ principal(E21) = principal(E22)) .}
\end{align*}
\]

Init: \[ p \rightarrow q \quad \{n_p,p\}_{k(q)} \]

Resp: \[ q \rightarrow p \quad \{n_p,n_q,q\}_{k(p)} \]

Ack: \[ p \rightarrow q \quad \{n_q\}_{k(q)} \]
Ciphertexts in Ack Messages

♦ Module **CIPHER3**: One constructor is declared.

\[
\text{op enc3 : Principal Nonce \rightarrow Cipher3 \{constr\}}
\]

\[\text{enc3}(p, n) \text{ denotes a ciphertext } \{n\}_{k(p)}.\]

✓ The following operators are provided:

\[
\begin{align*}
\text{eq } \text{key}(\text{enc3}(K,N)) &= K. \\
\text{eq } \text{nonce}(\text{enc3}(K,N)) &= N. \\
\text{eq } (E31 = E32) &= (\text{key}(E31) = \text{key}(E32) \text{ and } \\
&\quad \quad \quad \text{nonce}(E31) = \text{nonce}(E32)).
\end{align*}
\]

<table>
<thead>
<tr>
<th>Init: ( p \rightarrow q )</th>
<th>( {n_p,p}_{k(q)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resp: ( q \rightarrow p )</td>
<td>( {n_p,n_q,q}_{k(p)} )</td>
</tr>
<tr>
<td>Ack: ( p \rightarrow q )</td>
<td>( {n_q}_{k(q)} )</td>
</tr>
</tbody>
</table>
Messages (1)

♦ A term denoting a message contains
  • the (seeming) source (sender),
  • the destination (receiver), and
  • the body (ciphertext).

♦ In addition to those data, it also has
  • the actual source (creator).

because messages may be faked by the intruder.

This is meta-information.

♦ Such a term is in the form
  • $m(creator, sender, receiver, ciphertext)$
Module MESSAGE: Three constructors are declared.

[Message1 Message2 Message3 < Message]

- \( m1(p?, p, q, \text{enc1}(...)) \) denotes an Init message.
- \( m2(p?, p, q, \text{enc2}(...)) \) denotes a Resp message.
- \( m3(p?, p, q, \text{enc3}(...)) \) denotes an Ack message.

\[
\begin{array}{l}
\text{Init: } p \rightarrow q \ \{n_p,p\}_{k(q)} \\
\text{Resp: } q \rightarrow p \ \{n_p,n_q,q\}_{k(p)} \\
\text{Ack: } p \rightarrow q \ \{n_q\}_{k(q)}
\end{array}
\]
Messages (3)

✓ The following operators are provided:

\[
\begin{align*}
\text{eq creator}(\text{m1}(C,S,R,E1)) &= C . \\
\text{eq creator}(\text{m2}(C,S,R,E2)) &= C . \\
\text{eq creator}(\text{m3}(C,S,R,E3)) &= C . \\
\text{eq sender}(\text{m1}(C,S,R,E1)) &= S . \\
\text{eq sender}(\text{m2}(C,S,R,E2)) &= S . \\
\text{eq sender}(\text{m3}(C,S,R,E3)) &= S . \\
\text{eq receiver}(\text{m1}(C,S,R,E1)) &= R . \\
\text{eq receiver}(\text{m2}(C,S,R,E2)) &= R . \\
\text{eq receiver}(\text{m3}(C,S,R,E3)) &= R . \\
\text{eq cipher1}(\text{m1}(C,S,R,E1)) &= E1 . \\
\text{eq cipher2}(\text{m2}(C,S,R,E2)) &= E2 . \\
\text{eq cipher3}(\text{m3}(C,S,R,E3)) &= E3 .
\end{align*}
\]

✓ Note that cipher_i (i = 1,2,3) is declared as follows:

\[
\text{op cipher}_i : \text{Message}_i \to \text{Cipher}_i
\]
The following operators are provided (cont.):

\[
\begin{align*}
\text{eq } (M = M) &= \text{true} \ . \\
\text{eq } (M_{11} = M_{12}) &= (\text{creator}(M_{11}) = \text{creator}(M_{12}) \text{ and } \\
&\quad \text{sender}(M_{11}) = \text{sender}(M_{12}) \text{ and } \\
&\quad \text{receiver}(M_{11}) = \text{receiver}(M_{12}) \text{ and } \\
&\quad \text{cipher1}(M_{11}) = \text{cipher1}(M_{12})) \ . \\
\text{eq } (M_{21} = M_{22}) &= (\text{creator}(M_{21}) = \text{creator}(M_{22}) \text{ and } \\
&\quad \text{sender}(M_{21}) = \text{sender}(M_{22}) \text{ and } \\
&\quad \text{receiver}(M_{21}) = \text{receiver}(M_{22}) \text{ and } \\
&\quad \text{cipher2}(M_{21}) = \text{cipher2}(M_{22})) \ . \\
\text{eq } (M_{31} = M_{32}) &= (\text{creator}(M_{31}) = \text{creator}(M_{32}) \text{ and } \\
&\quad \text{sender}(M_{31}) = \text{sender}(M_{32}) \text{ and } \\
&\quad \text{receiver}(M_{31}) = \text{receiver}(M_{32}) \text{ and } \\
&\quad \text{cipher3}(M_{31}) = \text{cipher3}(M_{32})) \ . \\
\text{eq } (M_{11} = M_{21}) &= \text{false} \ . \\
\text{eq } (M_{11} = M_{31}) &= \text{false} \ . \\
\text{eq } (M_{21} = M_{31}) &= \text{false} \ .
\end{align*}
\]
Module SOUP:

```plaintext
mod* SOUP (D :: EQTRIV) principal-sort Soup {
  [Elt.D < Soup]
  op empty : -> Soup {constr}
  op _ _ : Soup Soup -> Soup
    {constr assoc comm id: empty}
  op _\in_ : Elt.D Soup -> Bool
  var S : Soup vars E1 E2 : Elt.D
  eq E1 \in empty = false .
  eq E1 \in (E2 S) = (E1 = E2) or E1 \in S .
}
where EQTRIV is as follows:

mod* EQTRIV principal-sort Elt {
  [Elt]
  op _=_ : Elt Elt -> Bool {comm}
  eq (E:Elt = E) = true .
  eq E1:Elt = E2:Elt if E1 = E2 .
}
```

Soups (Associative & Commutative Collections)
Networks

♦ Formalized as soups of messages.

\[ \text{SOUP(MESSAGE)} \ast \{\text{sort Soup} \rightarrow \text{Network}\} \]

✓ Sending a message is formalized as putting it in the soup.
✓ If the soup contains \( mi(p?, p, q, e_i) \), then \( q \) can receive it.
✓ \( q \) may believe that it originates in \( p \), but it may not be true.
✓ Suppose that messages are never deleted from the soup.
✓ This assumption may make it possible to do something that can never happen in the real world, but covers all possible cases.
Soups of Nonces & Soups of Random Numbers

♦ Soups of nonces:

\[
\text{SOUP(NONCE) \* \{sort Soup -> NonceSoup\}}
\]

\[
\{n_p, p\}_{k(i)} \quad \{n_q\}_{k(i)}
\]

Gleaned nonces

♦ Soups of random numbers:

\[
\text{SOUP(RANDOM) \* \{sort Soup -> RandSoup\}}
\]

\[
\ldots \, n(p, q, r_1) \, \ldots \\
\ldots \, n(q, p, r_2) \, \ldots \\
\]

Used random numbers

JAIST-FSSV2010, March 1-5, 2010, Kanazawa
Observable Values

♦ The three values are made observable:
  • the network (a soup of messages),
  • a soup of random numbers that have been used, and
  • a soup of nonces that have been gleaned by the intruder from the network.

♦ The corresponding operators (called observation or observer operators or functions) are as follows:

  \[
  \begin{align*}
  \text{op network} & : \text{System} \to \text{Network} \\
  \text{op rands} & : \text{System} \to \text{RandSoup} \\
  \text{op nonces} & : \text{System} \to \text{NonceSoup}
  \end{align*}
  \]

where \text{System} is the sort for the set of states, i.e. the state space.
An arbitrary initial state is denoted by the operator:

\[
\text{op init : } \rightarrow \text{ System } \{\text{constr}\}
\]

such that

\[
\begin{align*}
\text{eq network(init)} & = \text{ empty } . \\
\text{eq rands(init)} & = \text{ empty } . \\
\text{eq nonces(init)} & = \text{ empty } .
\end{align*}
\]
Formalization of Sending Messages

Sending Init, Resp and Ack messages according to the protocol is formalized as the operators (called transition operators or functions):

\[
\begin{align*}
\text{op sdm1} & : \text{System Principal Principal Random} \\
& \rightarrow \text{System \{constr\}} \\
\text{op sdm2} & : \text{System Principal Principal Principal} \\
& \text{Random Nonce} \rightarrow \text{System \{constr\}} \\
\text{op sdm3} & : \text{System Principal Principal Principal} \\
& \text{Nonce Nonce} \rightarrow \text{System \{constr\}}
\end{align*}
\]

\( \text{sdm1}(s, p, q, r) \) denotes the successor state of \( s \) when \( p \) sends an Init message to \( q \) in \( s \).

\( \text{sdm2}(s, q^?, p, q, r, n) \) denotes the successor state of \( s \) when \( p \) sends a Resp message to \( q \) in \( s \).

\( \text{sdm3}(s, q^?, p, q, n_1, n_2) \) denotes the successor state of \( s \) when \( p \) sends an Ack message to \( p \) in \( s \).
Sending Init Messages (1)

♦ The effective condition of $sdm_1$ is that a random number is fresh.

$$eq \; c-sdm_1(S,P,Q,R) = not(R \in \text{rands}(S)).$$

♦ When $c-sdm_1(S,P,Q,R)$ holds, in the successor state $sdm_1(S,P,Q,R)$,

\[
m_1(P,P,Q,\text{enc}_1(Q,n(P,Q,R),P))
\]

If $Q$ is intruder,

\[
n(P,Q,R)
\]

Otherwise,

\[
\ldots
\]
Sending Init Messages (2)

The equations for \( sdm1 \):

\[
\text{ceq network}(sdm1(S,P,Q,R)) = m1(P,P,Q,\text{enc1}(Q,n(P,Q,R),P)) \text{ network}(S)\
\text{if c-sdm1}(S,P,Q,R) .
\]

\[
\text{ceq rands}(sdm1(S,P,Q,R)) = R \text{ rands}(S) \text{ if c-sdm1}(S,P,Q,R) .
\]

\[
\text{ceq nonces}(sdm1(S,P,Q,R)) = (\text{if } Q = \text{intruder} \text{ then } n(P,Q,R) \text{ nonces}(S) \text{ else nonces}(S) \text{ fi})
\text{if c-sdm1}(S,P,Q,R) .
\]

\[
\text{ceq sdm1}(S,P,Q,R) = S \text{ if not c-sdm1}(S,P,Q,R) .
\]

Note that we need to have the last one because we have to explicitly declare that nothing changes if \( c-sdm1(S,P,Q,R) \) does not hold.
Sending Resp Messages (1)

- The effective condition of sdm2 is that there exists an Init message in the network and a random number is fresh.

\[ eq \ c-sdm2(S,Q?,P,Q,R,N) \]
\[ = (m1(Q?,Q,P,enc1(P,N,Q)) \in \text{network}(S) \text{ and } \not(R \in \text{rands}(S))) . \]

When \( c-sdm2(S,Q?,P,Q,RN) \) holds, in the successor state \( sdm2(S,Q?,P,Q,R,N) \),

\[ m2(P,P,Q,enc2(Q,N,n(P,Q,R),P)) \]
\[ m1(Q?,Q,P,enc1(P,N,Q)) \]

If \( Q \) is intruder,

\[ n(P,Q,R) \]
\[ \text{N} \]

Otherwise,

\[ \text{...} \]
Sending Resp Messages (2)

♦ The equations for $sdm2$:

\[
\begin{align*}
ceq \quad & \text{network}(sdm2(S,Q?,P,Q,R,N)) \\
& = m2(P,P,Q,enc2(Q,N,n(P,Q,R),P)) \text{ network}(S) \\
& \quad \text{if } c-sdm2(S,Q?,P,Q,R,N) . \\
ceq \quad & \text{rands}(sdm2(S,Q?,P,Q,R,N)) \\
& = R \text{ rands}(S) \text{ if } c-sdm1(S,P,Q,R) . \\
ceq \quad & \text{nonces}(sdm2(S,Q?,P,Q,R,N)) \\
& = (\text{if } Q = \text{intruder then } N \ n(P,Q,R) \ \text{nonces}(S) \\
& \quad \text{else } \text{nonces}(S) \ \text{fi}) \\
& \quad \text{if } c-sdm2(S,Q?,P,Q,R,N) . \\
ceq \quad & \text{sdm2}(S,Q?,P,Q,R,N) \\
& = S \text{ if not } c-sdm2(S,Q?,P,Q,R,N) .
\end{align*}
\]

Init: $p \rightarrow q \ {n_p,p}_{k(q)}$

Resp: $q \rightarrow p \ {n_p,n_q,q}_{k(p)}$

Ack: $p \rightarrow q \ {n_q}_{k(q)}$
Sending Ack Messages (1)

- The effective condition of $\text{sdm3}$ is that there exist an Init message and a Resp message in the network.

$$\text{eq } c-\text{sdm3}(S,Q?,P,Q,N1,N2)$$

$$= m1(P,P,Q,\text{enc1}(Q,N1,P)) \in \text{network}(S) \text{ and }$$

$$m2(Q?,Q,P,\text{enc2}(P,N1,N2,Q)) \in \text{network}(S) .$$

- When $c-\text{sdm3}(S,Q?,P,Q,N1,N2)$ holds, in the successor state $\text{sdm3}(S,Q?,P,Q,N1,N2)$,

$$m3(P,P,Q,\text{enc3}(Q,N2))$$

$$m1(P,P,Q,\text{enc1}(Q,N1,P))$$

$$m2(Q?,Q,P,\text{enc2}(P,N1,N2,Q))$$

... If $Q$ is intruder,

N2

Otherwise,

...
Sending Ack Messages (2)

◆ The equations for $sdm3$:

$$ceq \text{ network}(sdm3(S,Q?,P,Q,N1,N2))$$
$$= m3(P,P,Q,\text{enc3}(Q,N2)) \text{ network}(S)$$
if $c-sdm3(S,Q?,P,Q,N1,N2)$.

$$eq \text{ rands}(sdm3(S,Q?,P,Q,N1,N2)) = \text{ rands}(S) \ .$$

$$ceq \text{ nonces}(sdm3(S,Q?,P,Q,N1,N2))$$
$$= (\text{ if } Q = \text{ intruder} \text{ then } N2 \ \text{ nonces}(S)$$
$$\text{ else } \text{ nonces}(S) \ \text{ fi})$$
if $c-sdm3(S,Q?,P,Q,N1,N2)$.

$$ceq \text{ sdm3}(S,Q?,P,Q,N1,N2)$$
$$= S \text{ if not } c-sdm3(S,Q?,P,Q,N1,N2) \ .$$
The intruder may fake messages based on the nonces and ciphertexts gleaned from the network, which is formalized as the transition operators:

\[
\begin{align*}
\text{op } \text{fkm11} &: \text{System Principal Principal Message1} \\
&\rightarrow \text{System } \{\text{constr}\} \\
\text{op } \text{fkm12} &: \text{System Principal Principal Nonce} \\
&\rightarrow \text{System } \{\text{constr}\} \\
\text{op } \text{fkm21} &: \text{System Principal Principal Message2} \\
&\rightarrow \text{System } \{\text{constr}\} \\
\text{op } \text{fkm22} &: \text{System Principal Principal Nonce Nonce} \\
&\rightarrow \text{System } \{\text{constr}\} \\
\text{op } \text{fkm31} &: \text{System Principal Principal Message3} \\
&\rightarrow \text{System } \{\text{constr}\} \\
\text{op } \text{fkm32} &: \text{System Principal Principal Nonce} \\
&\rightarrow \text{System } \{\text{constr}\}
\end{align*}
\]

\(\text{fkm11}(s, p, q, m_1)\) denotes the successor state of \(s\) when the intruder fakes based on \(m_1\) an Init message, which seems to have been sent by \(p\) to \(q\), in \(s\).

\(\text{...}\)
The effective condition of $\text{fkm11}$ is that there exists an Init message in the network.

$$eq \ c\-\text{fkm11}(S,P,Q,M1) = M1 \ \text{in network}(S) .$$

The equations for $\text{fkm11}$:

$$ceq \ \text{network}(\text{fkm11}(S,P,Q,M1)) = m1(\text{intruder},P,Q,\text{cipher1}(M1)) \ \text{network}(S) \ \text{if} \ c-\text{fkm11}(S,P,Q,M1) .$$
$$eq \ \text{rands}(\text{fkm11}(S,P,Q,M1)) = \text{rands}(S) .$$
$$eq \ \text{nonces}(\text{fkm11}(S,P,Q,M1)) = \text{nonces}(S) .$$
$$ceq \ \text{fkm11}(S,P,Q,M1) = S \ \text{if not} \ c-\text{fkm11}(S,P,Q,M1) .$$
Faking Init Messages (2)

- The effective condition of \texttt{fkm12} is that a nonce is available to the intruder.
  \[
  \text{eq } c\text{-fkm12}(S,P,Q,N) = N \in \text{nonces}(S) .
  \]

- The equations for \texttt{fkm12}:
  \[
  \begin{align*}
  \text{ceq } \text{network}(\text{fkm12}(S,P,Q,N)) &= \text{ml(intruder},P,Q,\text{enc1}(Q,N,P)) \ \text{network}(S) \\
  &\quad \text{if } c\text{-fkm12}(S,P,Q,N) . \\
  \text{eq } \text{rands}(\text{fkm12}(S,P,Q,N)) &= \text{rands}(S) . \\
  \text{eq } \text{nonces}(\text{fkm12}(S,P,Q,N)) &= \text{nonces}(S) . \\
  \text{ceq } \text{fkm12}(S,P,Q,N) &= S \ \text{if not } c\text{-fkm12}(S,P,Q,N) .
  \end{align*}
  \]

\[
\text{ml(intruder},P,Q,\text{enc1}(Q,N,P))
\]

\[
\cdots
\]
Faking Resp Messages (1)

♦ The effective condition of $fkm21$ is that there exists a Resp message in the network.

\[
eq c-fkm21(S,P,Q,M2) = M2 \in \text{network}(S) .
\]

♦ The equations for $fkm21$:

\[
\begin{align*}
\text{ceq network}(fkm21(S,P,Q,M2)) &= m2(\text{intruder},P,Q,\text{cipher2}(M2)) \ \text{network}(S) \\
\text{if c-fkm21}(S,P,Q,M2) . \\
\text{eq rands}(fkm21(S,P,Q,M2)) &= \text{rands}(S) . \\
\text{eq nonces}(fkm21(S,P,Q,M2)) &= \text{nonces}(S) . \\
\text{ceq fkm21}(S,P,Q,M2) &= S \ \text{if not c-fkm21}(S,P,Q,M2) .
\end{align*}
\]
Faking Resp Messages (2)

♦ The effective condition of $\text{fkm22}$ is that two different nonces are available to the intruder.

\[
\text{eq } c-\text{fkm22}(S,P,Q,N_1,N_2) = N_1 \in \text{nonces}(S) \text{ and } N_2 \in \text{nonces}(S) \text{ and not}(N_1 = N_2).
\]

♦ The equations for $\text{fkm22}$:

\[
\begin{align*}
\text{ceq } \text{network}(\text{fkm22}(S,P,Q,N_1,N_2)) &= \text{m2(intruder,P,Q,enc2(Q,N_1,N_2,P)) network}(S) \text{ if } c-\text{fkm22}(S,P,Q,N_1,N_2). \\
\text{eq } \text{rands}(\text{fkm22}(S,P,Q,N_1,N_2)) &= \text{rands}(S). \\
\text{eq } \text{nonces}(\text{fkm22}(S,P,Q,N_1,N_2)) &= \text{nonces}(S). \\
\text{ceq } \text{fkm22}(S,P,Q,N_1,N_2) &= S \text{ if not } c-\text{fkm22}(S,P,Q,N_1,N_2).
\end{align*}
\]
Faking Ack Messages (1)

- The effective condition of $fkm_{31}$ is that there exists an Ack message in the network.
  \[eq \ c-fkm_{31}(S,P,Q,M3) = M3 \in \text{network}(S) .\]

- The equations for $fkm_{31}$:
  \[ceq \ \text{network}(fkm_{31}(S,P,Q,M3))\]
  \[= m3(\text{intruder},P,Q,\text{cipher3}(M3)) \ \text{network}(S)\]
  \[if \ c-fkm_{31}(S,P,Q,M3) .\]
  \[eq \ rands(fkm_{31}(S,P,Q,M3)) = rands(S) .\]
  \[eq \ nonces(fkm_{31}(S,P,Q,M3)) = nonces(S) .\]
  \[ceq fkm_{31}(S,P,Q,M3) = S \ if \ not \ c-fkm_{31}(S,P,Q,M3) .\]

\[m3(\text{intruder},P,Q,\text{cipher3}(M3))\]

\[M2\]

\[\cdots\]
The effective condition of $fkm32$ is that a nonce is available to the intruder.

\[ eq \ c-fkm32(S,P,Q,N) = N \ \text{in} \ \text{nonces}(S) . \]

The equations for $fkm32$:

\begin{align*}
& \text{ceq} \ \text{network}(fkm32(S,P,Q,N)) = m3(\text{intruder}, P, Q, \text{enc3}(Q,N)) \ \text{network}(S) \\
& \text{if} \ c-fkm32(S,P,Q,N) . \\
& \text{eq} \ \text{rands}(fkm32(S,P,Q,N)) = \text{rands}(S) . \\
& \text{eq} \ \text{nonces}(fkm32(S,P,Q,N)) = \text{nonces}(S) . \\
& \text{ceq} \ fkm32(S,P,Q,N) = S \ \text{if} \ \text{not} \ c-fkm32(S,P,Q,N) . \\
\end{align*}
Constructors of State Space

♦ The (reachable) state space (System) is constructed from the constructors init, sdm1, sdm2, sdm3, fkm11, fkm12, fkm21, fkm22, fkm31 and fkm32.

♦ Theorems on System (or invariant properties of the OTS of NSLPK) can be proved by simultaneous structural induction of the reachable state space (System).

\[
\text{NSLPK} \models p_i(\text{init}) \quad \begin{array}{cccc}
\text{NSLPK} \cup \{p_j(s) = \text{true for } j = 1,\ldots,n\} \\
\models \{s, p, q, r\} p_i(\text{sdm1}(s, p, q, r)) \\
\cdots \\
\text{NSLPK} \models (\forall S:\text{System}) p_i(S) \text{ for any } l \in \{1,\ldots,n\}
\end{array}
\]
Formalization of Agreement Property (1)

♦ Whenever a protocol run is successfully completed by $p$ and $q$,
  • the principal with which $p$ is communicating is really $q$, and
  • the principal with which $q$ is communicating is really $p$.

♦ The property can be rephrased based on the specification of NSLPK as follows:
  • Whenever $p$ receives a Resp message that is what $p$ really expects, the Resp message originates in $q$, and
  • whenever $q$ receives an Ack message that is what $q$ really expects, the Ack message originates in $p$. 
Precisely the property is described as follows:

- If $p$ is not the intruder, then whenever $p$ has sent an Init message to $q$ and receives a valid Resp message, which seems to have been sent by $q$, the Resp message originates in $q$, and
- If $q$ is not the intruder, then whenever $q$ has sent a Resp message to $p$ and receives a valid Ack message, which seems to have been sent by $p$, the Ack message originates in $p$.

Note that if $p$ (or $q$) is the intruder, then the intruder can fake a valid Resp (or Ack) message, which does not originate in $q$ (or $p$).

If two nonces $n, n'$ are available to the intruder, then the intruder can fake the messages:

$$m_1(\text{intruder}, \text{intruder}, q, \text{enc}_1(q, n, \text{intruder}))$$

$$m_2(\text{intruder}, q, \text{intruder}, \text{enc}_2(\text{intruder}, n, n', q))$$
Formalization of Agreement Property (3)

♦ That there exists a message in the network means that it has been sent by (or originates in) the creator, and any messages in the network whose receivers are \( p \) can be received by \( p \). So, the property is formalized:

\[
eq \text{inv}_1(S,P,Q,Q?,R,N) = (\neg(P = \text{intruder}) \quad \text{and} \quad \\
 m_1(P,P,Q,\text{enc}_1(Q,n(P,Q,R),P)) \in \text{network}(S) \quad \text{and} \quad \\
m_2(Q?,Q,P,\text{enc}_2(P,n(P,Q,R),N,Q)) \in \text{network}(S) \quad \text{implies} \quad \\
m_2(Q,Q,P,\text{enc}_2(P,n(P,Q,R),N,Q)) \in \text{network}(S)) .
\]

\[
eq \text{inv}_2(S,P,Q,P?,R,N) = (\neg(Q = \text{intruder}) \quad \text{and} \quad \\
m_2(Q,Q,P,\text{enc}_2(P,N,n(Q,P,R),Q)) \in \text{network}(S) \quad \text{and} \quad \\
m_3(P?,P,Q,\text{enc}_3(Q,n(Q,P,R))) \in \text{network}(S) \quad \text{implies} \quad \\
m_3(P,P,Q,\text{enc}_3(Q,n(Q,P,R))) \in \text{network}(S)) .
\]
NSLPK has been used as an example to discuss what to prepare for verification that a system enjoys a property:

• Make the assumptions clear.
• Create a transition system (an OTS) of the system.
  ✓ Formalize data used.
  ✓ Determine what values are observed.
  ✓ Determine what actions of the system are formalized as transitions.

You may want to take into account the property.
• Formalize the property based on the specification of the OTS.