

The Needham-Schroeder Public-Key Protocol –How to Break It

Jens Chr. Godskesen
IT University of Copenhagen

Overview

- The Needham-Schroeder Public-Key Protocol
- Modelling the protocol
- Establishing the attack
- A general model

What is a security protocol?

A *protocol* is a set of rules of how to communicate in order to obtain a certain goal. Take for instance HTTP.

If the goal is to enforce security the communicating protocols are often denoted security protocols. HTTPS is a security protocol.

The Needham-Schroeder Public-Key *authentication* protocol (NSPK) is our case of study.

It was in use for about 17 years before it was eventually broken.

Encryption/Decryption

Encryption is used to protect a message m from being read by others it may be encrypted using a parametric key K in which case we write:

$$\{m\}_K$$

If the key is *symmetric* we may decrypt with the same key, hence

$$m = \{\{m\}_K\}_K$$

A pair of *asymmetric* keys consists of a *private* and a *public* key, K^- and K^+ respectively. The two keys are each others inverses, ie.

$$m = \{\{m\}_{K^-}\}_{K^+} = \{\{m\}_{K^+}\}_{K^-}$$

The NSPK protocol

A simplified version of the NSPK protocol can be described as follows using standard notation for security protocol definition:

$$\begin{aligned} Alice &\longrightarrow Bob &: & \{n_A, K_A^+\}_{K_B^+} \\ Bob &\longrightarrow Alice &: & \{n_A, n_B\}_{K_A^+} \\ Alice &\longrightarrow Bob &: & \{n_B\}_{K_B^+} \end{aligned}$$

Extending CCS

Extend CCS by letting output actions now become on the form:

$$\text{'act}\langle a_1, a_2, \dots, a_k \rangle.P$$

intuitively meaning that along the channel act the values a_1, a_2, \dots, a_k can be send after which the process behaves as P .

Dually CCS is extended with input actions on the form:

$$\text{act}(x_1, x_2, \dots, x_k).Q$$

Intuitively $\text{act}(x_1, x_2, \dots, x_k)$ is a binder of the free occurrences of the variables x_1, x_2, \dots, x_k in Q .

Extending CCS

If we take the parallel composition

$$\text{'act}\langle a_1, a_2, \dots, a_k \rangle.P \mid \text{act}(x_1, x_2, \dots, x_k).Q$$

then there is an internal transition to

$$P \mid Q[a_1/x_1, a_2/x_2, \dots, a_k/x_k]$$

Alice and Bob

```
agent Alice(a,kB,kA,i,c) =
    (^ n)('i.'a<kB,n,kA>.a(x,y,z).[x=kA][y=n]'c.'a<kB,z>.0)
agent Bob(a,kB,r,c) =
    (^ n)(a(x,y,z).[x=kB]'r.'a<z,y,n>.a(x,y).[x=kB][y=n]'c.0)
agent Sys(iAB,rBA,cAB,cBA) = (^ a,kA,kB)( Alice<a,kB,kA,iAB,cAB>
    | Bob<a,kB,rBA,cBA>)
```

Running the protocol

MWB>step Sys(iAB,rBA,cAB,cBA)

* Valid responses are:

a number $N \geq 0$ to select the N th commitment,

<CR> to select commitment 0,

q to quit.

0: |>'iAB.(\tilde{v} , \tilde{v}_6 , \tilde{v}_7)((\tilde{v}_8) ' \tilde{v} < \tilde{v}_7 , \tilde{v}_8 , \tilde{v}_6 >. $\tilde{v}(x,y,z)$.[$x=\tilde{v}_6$]... (A)

Step>0

0: |>t.(\tilde{v} , \tilde{v}_6 , \tilde{v}_7 , \tilde{v}_8)($\tilde{v}(x,y,z)$.[$x=\tilde{v}_6$][$y=\tilde{v}_8$]'cAB.' \tilde{v} < \tilde{v}_7 , z >.0 ... (B)

Step>0

0: |>'rBA.(\tilde{v} , \tilde{v}_6 , \tilde{v}_7 , \tilde{v}_8)($\tilde{v}(x,y,z)$.[$x=\tilde{v}_6$][$y=\tilde{v}_8$]'cAB.' \tilde{v} < \tilde{v}_7 , z >.0 ...

Step>0

0: |>t.(\tilde{v} , \tilde{v}_6 , \tilde{v}_7)('cAB.' \tilde{v} < \tilde{v}_6 , \tilde{v}_7 >.0 | $\tilde{v}(x,y)$.[$x=\tilde{v}_6$][$y=\tilde{v}_7$]'cBA.0)

Step>0

0: |>'cAB.(\tilde{v} , \tilde{v}_6 , \tilde{v}_7)(' \tilde{v} < \tilde{v}_6 , \tilde{v}_7 >.0 | $\tilde{v}(x,y)$.[$x=\tilde{v}_6$][$y=\tilde{v}_7$]'cBA.0)

Step>0

0: |>t.'cBA.0

Step>0

0: |>'cBA.0

Step>0

No commitments for 0

Quitting.

Scope extrusion

When a locally restricted name is send to another process as e.g. in

$$(\hat{a})((\hat{n})(a\langle n \rangle.P) \mid a(x).Q)$$

then the internal transition results in

$$(\hat{a}, n)(P \mid Q[n/x])$$

Adding more parties

```
agent Initiator(a,kX,kY,i,c) =
    (^ n)('i.'a<kY,n,kX>.a(x,y,z).[x=kX][y=n]'c.'a<kY,z>.0)
agent Responder(a,k,r,c) =
    (^ n)(a(x,y,z).[x=k]'r.'a<z,y,n>.a(x,y).[x=k][y=n]'c.0)
agent Alice(ab,ae,kA,kB,kE,iAB,iAE,cAB,cAE) =
    Initiator<ab,kA,kB,iAB,cAB> + Initiator<ae,kA,kE,iAE,cAE>
agent Bob(ab,eb,kB,rBA,rBE,cBA,cBE) =
    Responder(ab,kB,rBA,cBA) + Responder(eb,kB,rBE,cBE)
agent Eve(ae,eb,kE,kB,iEB,rEA,cEB,cEA) =
    Initiator<eb,kE,kB,iEB,cEB> + Responder(ae,kE,rEA,cEA)
agent Sys(iAB,iAE,iEB,rBA,rBE,rEA,cAB,cBA,cAE,cEA,cBE,cEB) =
    (^ ab,ae,eb,kA,kB,kE) ( Alice<ab,ae,kA,kB,kE,iAB,iAE,cAB,cAE>
        | Bob<ab,eb,kB,rBA,rBE,cBA,cBE>
        | Eve<ae,eb,kE,kB,iEB,rEA,cEB,cEA> )
```

The Attack

Letting Eve now be the intruder the attack goes as follows:

$$\begin{aligned} Alice &\longrightarrow Eve && : \{n_A, K_A^+\}_{K_E^+} \\ Eve(Alice) &\longrightarrow Bob && : \{n_A, K_A^+\}_{K_B^+} \\ Bob &\longrightarrow Eve(Alice) && : \{n_A, n_B\}_{K_A^+} \\ Eve &\longrightarrow Alice && : \{n_A, n_B\}_{K_A^+} \\ Alice &\longrightarrow Eve && : \{n_B\}_{K_E^+} \\ Eve &\longrightarrow Bob && : \{n_B\}_{K_B^+} \end{aligned}$$

Exercise Explain the two interleavings of the NSPK protocol constituting the attack.

Modeling the intruder

```
agent Initiator(a,kX,kY,i,c) =
    (^ n)('i.'a<kY,n,kX>.a(x,y,z).[x=kX][y=n]'c.'a<kY,z>.0)
agent Responder(a,k,r,c) =
    (^ n)(a(x,y,z).[x=k]'r.'a<z,y,n>.a(x,y).[x=k][y=n]'c.0)
agent Alice(ae,kA,kE,iAE,cAE) = Initiator(ae,kA,kE,iAE,cAE)
agent Bob(eb,kB,rBA,cBA) = Responder(eb,kB,rBA,cBA)
agent Eve(ae,eb,kE,kB) =
    ae(x,y,z).[x=kE]'eb<kB,y,z>.eb(x,y,z).'ae<x,y,z>.ae(x,y).[x=kE]'eb<kB,y>.0
agent Sys(iAE,rBA,cBA,cAE) =
    (^ ae,eb,kA,kB,kE)( Alice<ae,kA,kE,iAE,cAE>
                        | Bob<eb,kB,rBA,cBA>
                        | Eve<ae,eb,kE,kB>)
```

The attack is indeed possible in that the system satisfies the following HML formula:

```
MWB>prove Sys(iAE,rBA,cBA,cAE)
    <'iAE><t><t><'rBA><t><t><'cAE><t><t><'cBA>TT
```

```
Model Prover says: YES!
(21 inferences)
```

How to make a general intruder?

An intruder may be beyond initiating and responding to runs of the protocol:

- listen to any message being part of a protocol run,
- replay any message seen possibly changing unencrypted parts of the message,
- steal messages and hide them from other parties,
- decrypt messages encrypted by the keys he knows, and thereby probably learn new nonces and keys, and
- send messages based on the knowledge (of nonces and keys) obtained so far.

The model with a general intruder

```
agent Initiator(a,kX,kY,i,ic) =
    (^ n)('i.'a<kY,n,kX>.Initiator1<a,kX,kY,i,ic,n>)
agent Initiator1(a,kX,kY,i,ic,n) =
    a(x,y,z).([x=kX]([y=n]'ic.'a<kY,z,z>.0
                + 'a<x,y,z>.Initiator1<a,kX,kY,i,ic,n>)
                + 'a<x,y,z>.Initiator1<a,kX,kY,i,ic,n>))

agent Responder(a,k,k1,k2,r1,r2,c1,c2) =
    a(x,y,z).([x=k]Responder1<a,k,k1,k2,r1,r2,c1,c2,y,z>
                + 'a<x,y,z>.Responder<a,k,k1,k2,r1,r2,c1,c2>))
agent Responder1(a,k,k1,k2,r1,r2,c1,c2,y,z) =
    [z=k1]'r1.Responder2<a,k,c1,y,z>
    + [z=k2]'r2.Responder2<a,k,c2,y,z>
agent Responder2(a,k,c,y,z) = (^ n)('a<z,y,n>.Responder3<a,k,c,n>)
agent Responder3(a,k,c,n) =
    a(x,y,z).([x=k]([y=n]'c.0 + 'a<x,y,z>.Responder3<a,k,c,n>)
                + 'a<x,y,z>.Responder3<a,k,c,n>))
```

The model with a general intruder

```
agent Alice(a,kA,kB,kE,iAB,iAE,icAB,icAE) =
    Initiator<a,kA,kB,iAB,icAB> + Initiator<a,kA,kE,iAE,icAE>
agent Bob(a,kB,kA,kE,rAB,rEB,cAB,cEB) =
    Responder(a,kB,kA,kE,rAB,rEB,cAB,cEB)
agent Eve(a,kE,kA,kB) = (a(x,y,z).([x=kE]Eve1<a,kE,kA,kB,y,z>
    + 'a<x,y,z>.Eve<a,kE,kA,kB>))
    + (^ n)('a<kB,n,kE>.Eve<a,kE,kA,kB>))
agent Eve1(a,kE,kA,kB,n,k) = 'a<kA,n,k>.Eve<a,kE,kA,kB>
    + 'a<kB,n,k>.Eve<a,kE,kA,kB>

agent Sys(iAB,iAE,icAB,icAE,rAB,rEB,cAB,cEB) =
    (^ a,kA,kB,kE)( Alice<a,kA,kB,kE,iAB,iAE,icAB,icAE>
        | Bob<a,kB,kA,kE,rAB,rEB,cAB,cEB>
        | Eve<a,kE,kA,kB> )
```

All parties share and communicate over the same channel.

A channel always take a fixed number of parameters.

Revealing the attack

The attack is still possible:

```
MWB>prove Sys(iAB,iAE,icAB,icAE,rAB,rEB,cAB,cEB)
      <'iAE><t><t><'rAB><t><t><'icAE><t><t><'cAB>TT
```

Model Prover says: YES!

(83 inferences)

A more general property for the attack

Intuitively we would say that there has been an attack if there is an execution of the protocol where Bob ends committing to be running a session with Alice without ever Alice prior in that same execution has committed to be running a session with Bob.

That property is as a kind of weak liveness property where eventually $\langle 'cAB \rangle_{TT}$, holds but only along a path where no event $'icAB$, i.e. is carried out.

A more general property for the attack

```
MWB>prove Sys(iAB,iAE,icAB,icAE,rAB,rEB,cAB,cEB)
      mu X.( <'cAB>TT |
              ( <t>X | <'iAB>X | <'iAE>X | <'icAE>X
                | <'rAB>X | <'rEB>X | <'cAB>X | <'cEB>X ))
Model Prover says: YES!
(31483 inferences)
```

Notice, that the event 'icAB is not allowed to be used for ensuring progress, this is precisely the means to ensure that Alice never along that run has committed to be running a session with Bob.

Exercise Explain why any model satisfying

$$\langle 'iAE \rangle \langle t \rangle \langle t \rangle \langle 'rAB \rangle \langle t \rangle \langle t \rangle \langle 'icAE \rangle \langle t \rangle \langle t \rangle \langle 'cAB \rangle TT$$

also satisfy the property above.

A Corrected Version of NSPK

The corrected version of the NSPK protocol they define by:

$$\begin{aligned} Alice \longrightarrow Bob & : \{n_A, K_A^+\}_{K_B^+} \\ Bob \longrightarrow Alice & : \{n_A, n_B, K_B^+\}_{K_A^+} \\ Alice \longrightarrow Bob & : \{n_B\}_{K_B^+} \end{aligned}$$

Exercise Explain why the attack found above cannot be carried out any longer.