Lição 8: Estudo de Caso: Um Protocolo de Autenticação

Luís Soares Barbosa

Sumário

Esta última Lição é integralmente dedicada a um caso de estudo em modelação e análise de processos, incluindo a discussão de satisfação de propriedades formuladas na lógica estudada na Lição 7. O caso em discussão é uma versão simplificada de um protocolo de autenticação muito popular: o Needham-Schroeder Public-Key Protocol.

Ao contrário de que se passou em Lições anteriores, toda a discussão do caso é directamente conduzida sobre uma ferramenta de simulação e análise — o MWB *— com a qual os alunos são supostos adquirir uma razoável familiaridade.*

Disciplina: Métodos de Programação IV (2005-06)
Docente *Luís Soares Barbosa,* Departamento de Informática, Universidade do Minho

---

1 O Protocolo

Enquanto protocolo de *autenticação*, o objectivo do protocolo de Needham-Schroeder, que abreviaremos por NSPK (de *Needham-Schroeder Public-Key Protocol*) [NS78], é estabalecer a identidade de duas entidades participantes numa conversação. Para isso recorre a uma técnica de encriptação por chave pública. A ideia base é que cada entidade possui um par de chaves, K^- e K^+, ditas, respectivamente, *privada* e *pública*, tais que uma mensagem m encriptada com a chave pública pode ser desencriptada com a chave privada, o que na notação usual para definição de protocolos de segurança, se escreve

$$m \;=\; \{\{m\}_{K^+}\}_{K^-} \tag{1}$$

Como o nome indica, as chaves públicas são publicamente conhecidas... No entanto, uma mensagem encriptada com uma chave pública só pode ser desencriptada pelo proprietário da correspondente chave privada.

A versão simplificada do NSPK que analisaremos nesta Lição, usada em [Low96], é descrita da forma seguinte:

$$\begin{array}{rcll} Ana & \longrightarrow & Rui & : \; \{n_A, {K_A}^+\}_{{K_R}^+} \\ Rui & \longrightarrow & Ana & : \; \{n_A, n_R\}_{{K_A}^+} \\ Ana & \longrightarrow & Rui & : \; \{n_R\}_{{K_R}^+} \end{array}$$

O significado intuitivo do protocolo é o seguinte: Ana deseja estabelecer a autenticidade de uma interacção com Rui, para o que inicia o protocolo enviando-lhe uma mensagem encriptada com a chave pública de

Rui. Apenas este, portanto, a pode desencriptar. A mensagem contém um *nonce* n_A gerado aleatoriamente assim com a chave pública de Ana. Ao receber a mensagem esta é desencriptada por Rui, que para isso usa a sua própria chave privada, obtendo, assim, n_A e ${K_A}^+$. Em resposta, Rui envia a Ana n_A e um novo *nonce* n_R, também gerado aleatoriamente, ambos cifrados com a chave pública de Ana. Quando esta recebe esta segunda mensagem pode ter a certeza de que foi enviada por Rui. De facto, só ele poderia ter desencriptado a primeira mensagem e obtido o valor n_A. Finalmente, Ana retorna n_R a Rui encriptada com ${K_R}^+$. É então a vez de Rui ficar convencido de que foi efectivamente Ana quem enviou a mensagem, pois só ela poderia conhecer n_R. Os dois intervenientes adquirem, assim, confiança na identidade um do outro e podem prosseguir a conversação.

O NSPK ficou famoso por ter demorado cerca de 17 anos a ser concebido um ataque que o violasse [BAN90]. Durante esse tempo foi usado em diversas aplicaçoes comerciais e, nomeadamente no protocolo Kerberos de autenticação de redes cliente-servidor. Nesta Lição vamos modelar o NSPK em π-*calculus* e analisar a possibilidade de ser atacado por recurso ao MWB [VM94].

2 Modelando o NSPK

Comecemos por identificar cada um dos prticipantes, Ana e Rui, por um processo ligados por uma canal a. Designamos por S o processo correspondente à sua agregação em paralelo. O código correspondente em MWB é[1]:

```
agent Ana(a,kR,kA,i,c) = \
     (^n) ('i.'a<kR,n,kA>.a(x,y,z).[x=kA][y=n]'c.'a<kR,z>.0)
agent Rui(a,kR,r,c) = \
     (^n) (a(x,y,z).[x=kR]'r.'a<z,y,n>.a(x,y).[x=kR][y=n]'c.0)

agent S(iAR,rRA,cAR,cRA) = \
     (^a,kA,kR) (Ana<a,kR,kA,iAR,cAR> | Rui<a,kR,rRA,cRA>)
```

Apesar da notação usada dever ser, nesta altura do curso, clara para todos, convirá formular algumas notas. Em particular, note-se a forma como se garante a produção de um *nonce* privado através do operador de restrição: até ser comunicado em a, o valor n, que corresponde a n_A, é privado ao processo Ana. A mensagem que envia

```
'a<kR,n,kA>
```

tem 3 parâmetros: o primeiro é a chave pública de Rui, o que, na convenção notacional que adoptamos, significa isso que a mensagem é encriptada com essa chave. Os dois outros parâmetros são os valores que efectivamente a constituem. Após o envio, Ana fica a aguardar a recepção no mesmo canal partilhado a uma mensagem encriptada com a chave pública de Rui e que contenha o *nonce* previamente enviado. Se efectivamente recebe esta informação, o processo que modela Ana sinaliza sucesso num canal c e envia a Rui o *nonce* recebido encriptado com a sua chave pública.

A modelação de Rui segue princípios similares, sinalizando também em c a completação bem sucedida do protocolo.

Do ponto de vista de um observador externo, aquilo que é relevante observar para verificar o funcionamento adequado do protocolo são os eventos que sinalizam o *início* (via canal i) e *fim* (via c) da execução, sendo o progresso interno representado por sequências de acções τ. Usando o MWB podemos simular a evolução da conversação (modelada pelo processo S):

```
MWB> step S(iAR,rRA,cAR,cRA)
```

[1] Recorde que o símbolo \ é usado no MWB para permitir a continuação de uma expressão noutra linha.

```
* Valid responses are:
  a number N >= 0 to select the Nth commitment,
  <CR> to select commitment 0,
  q to quit.
0: |>'iAR.(^~v,~v6,~v7)((^~v8)'~v<~v7,~v8,~v6>.~v(x,y,z)
         .[x=~v6][y=~v8]'cAR.'~v<~v7,z>.0
     | (^n)~v(x,y,z).[x=~v7]'rRA.'~v<z,y,n>.~v(x12,y13).[x12=~v7][y13=n]'cRA.0)
Step>0
0: |>t.(^~v,~v6,~v7,~v8)(~v(x,y,z).[x=~v6][y=~v8]'cAR.'~v<~v7,z>.0
     | (^~v9)'rRA.'~v<~v6,~v8,~v9>.~v(x,y).[x=~v7][y=~v9]'cRA.0)
Step>0
0: |>'rRA.(^~v,~v6,~v7,~v8)(~v(x,y,z).[x=~v6][y=~v8]'cAR.'~v<~v7,z>.0
      | (^~v9)'~v<~v6,~v8,~v9>.~v(x,y).[x=~v7][y=~v9]'cRA.0)
Step>0
0: |>t.(^~v,~v6,~v7)('cAR.'~v<~v6,~v7>.0 | ~v(x,y).[x=~v6][y=~v7]'cRA.0)
Step>0
0: |>'cAR.(^~v,~v6,~v7)('~v<~v6,~v7>.0 | ~v(x,y).[x=~v6][y=~v7]'cRA.0)
Step>0
0: |>t.'cRA.0
Step>0
0: |>'cRA.0
Step>0
No commitments for 0
Quitting.
MWB>
```

Podemos aqui observar que Ana inicia o protocolo, Rui responde e, por fim, ambos se comprometem na conversação. Note-se ainda que o nome `v8` que surge logo na primeira linha do *printout* é a representação interna no MWB para o *nonce* n_A. Quando Ana o envia para Rui, o escopo deste nome alarga-se para incluir o processo que modela Rui. Por essa razão aparece, logo na linha seguinte, como parâmetro da restrição mais externa.

Exercício 1

Esta é uma situação típica em π-*calculus*. Identifique-a com clareza e explique o seu impacto no cálculo.

O passo seguinte na nossa modelação do NSPK consiste em introduzir as versões genéricas de entidades com os papeis de, respectivamente, *activar* o protocolo e *responder*. Os dois processos que os representam, $Init$ e $Resp$, têm definições análogas a Ana e Rui na versão anterior.

```
agent Init(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 Resp(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)
```

Consideremos, agora, uma situação envolvendo três participantes — Ana, Rui e $Mefistofeles$ — todos eles definidos usando os processos genéricos especificados acima instanciados com os canais relevantes. Nesta configuração Ana e Rui continuam com os seus papeis de activação e resposta, respectivamente, enquanto o novo participante $Mefistofeles$ pode actuar ora como iniciador ora como respondente.

```
agent Ana(ar,aj,kR,kA,kJ,iAR,iAJ,cAR,cAJ)  = \
      Init<ar,kA,kR,iAR,cAR> + Init<aj,kA,kJ,iAJ,cAJ>

agent Rui(ar,jr,kR,rRA,rRJ,cRA,cRJ) = \
```

```
      Resp<ar,kR,rRA,cRA> + Resp<jr,kR,rRJ,cRJ>

agent Mefistofeles(aj,jr,kJ,kR,iJR,rJA,cJR,cJA) = \
      Init<jr,kJ,kR,iJR,cJR> + Resp<aj,kJ,rJA,cJA>

agent S(iAR,iAJ,iJR,rRA,rRJ,rJA,cAR,cRA,cAJ,cJA,cRJ,cJR) =\
      (^ ar,aj,jr,kA,kR,kJ) \
      ( Ana<ar,aj,kA,kR,kJ,iAR,iAJ,cAR,cAJ>  \
         |   \
        Rui<ar,jr,kR,rRA,rRJ,cRA,cRJ> \
         |   \
        Mefistofeles<aj,jr,kJ,kR,iJR,rJA,cJR,cJA>  )
```

3 Quando Ataca Mefistófeles

Eis que fala o letrado que em vós há!
Coisa que não toqueis, a léguas está,
O que não percebeis não é real,
O que não calculais como erro vale,
O que não pesais não tem peso para vós,
E o que não cunhais é falso, dizeis.

J. W. Goethe. *Fausto*.
— fala de Mefistófeles no 1º Acto da 2ª Parte.
(trad. de João Barrento, *Relógio d'Água*, 2003)

O Ataque

O ataque ao NSPK descrito em [Low96] pode ser apresentado da forma seguinte, assumindo-se que o intruso é modelado por Mefistofeles:

$$
\begin{aligned}
Ana \longrightarrow \text{Mefistofeles} &: \{n_A, {K_A}^+\}_{K_J{}^+} \\
\text{Mefistofeles}(Ana) \longrightarrow Rui &: \{n_A, {K_A}^+\}_{K_R{}^+} \\
Rui \longrightarrow \text{Mefistofeles}(Ana) &: \{n_A, n_R\}_{K_A{}^+} \\
\text{Mefistofeles} \longrightarrow Ana &: \{n_A, n_R\}_{K_A{}^+} \\
Ana \longrightarrow \text{Mefistofeles} &: \{n_R\}_{K_J{}^+} \\
\text{Mefistofeles} \longrightarrow Rui &: \{n_R\}_{K_R{}^+}
\end{aligned}
$$

Ana inicia o protocolo com o intruso que, fazendo-se passar por Ana envia a mensagem para Rui encriptada com a chave pública deste. Ao desencriptar a mensagem, Rui encontra a chave pública de Ana e fica convencido ter sido esta a enviar-lhe a mensagem. Procedendo de acordo com o protocolo, Rui gera o seu *nonce* que re-envia, com o *nonce* recebido, encriptados com a chave pública de Ana, ao intruso Mefistofeles que, erradamente, pensa ser Ana. Apesar dos seus sortilégios na peça de Goethe, Mefistofeles não tem meios para desencriptar a mensagem e, portanto, simplesmente a faz seguir para Ana. Esta, por sua vez, desencripta a mensagem e encontra um *nonce* n_R que supõe pertencer a Mefistofeles. Continuando a cumprir o protocolo, envia a este esse *nonce* encriptado com a chave pública de Mefistofeles que usara no início do diálogo. Mefistofeles é, agora, capaz de desencriptar a mensagem e aceder a n_R que vai, por fim, enviar a Rui. Este fica, então, convencido de estar a falar com Ana pois, segundo ele, só esta seria conhecedora do *nonce* n_R.

Exercício 2

Note que este ataque é, em verdade, uma intercalação de duas execuções do NSPK. Identifique-as.

Um Modelo para Mefistófeles

A configuração para este ataque é modelada abaixo. Note-se que tanto Ana como Rui voltaram a ser definidos como iniciador e respondente de um único diálogo (enquanto atrás podiam participar nesse papel em dois diálogos distintos), mas a definição de $Mefistofeles$ alterou-se assumindo o papel de intruso que acima caracterizamos. Assim,

```
agent Init(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 Resp(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 Ana(aj,kA,kJ,iAJ,cAJ)  = Init<aj,kA,kJ,iAJ,cAJ>

agent Rui(jr,kR,rRA,cRA) =   Resp<jr,kR,rRA,cRA>

agent Mefistofeles(aj,jr,kJ,kR) = \
      aj(x,y,z).[x=kJ]'jr<kR,y,z>.jr(x,y,z).'aj<x,y,z>.aj(x,y).[x=kJ]'jr<kR,y>.0


agent S(iAJ,rRA,cRA,cAJ) =\
      (^ aj,jr,kA,kR,kJ) \
      ( Ana<aj,kA,kJ,iAJ,cAJ>  \
         |   \
        Rui<jr,kR,rRA,cRA> \
         |   \
        Mefistofeles<aj,jr,kJ,kR>  )
```

Para verificar que o ataque pode mesmo ocorrer vamos recorrer à capacidade do MWB para verificar a validade de propriedades formuladas na lógica de Hennessy-Milner discutida na Lição anterior. Assim,

```
MWB>prove S(iAJ,rRA,cRA,cAJ)\
    <'iAJ><t><t><'rRA><t><t><'cAJ><t><t><'cRA>TT
Model Prover says: YES!
 (21 inferences)
MWB>
```

Exercício 3

Explique informalmente o significado desta fórmula modal e a razão pela qual a sua validade traduz a existência de um ataque ao protocolo.

4 Modelando um Mefistófeles Genérico

A nossa preocupação até este ponto foi modelar um intruso para efectuar um ataque específico e bem conhecido ao NSPK. Um procedimento semelhante pode ser adoptado para analisar outras possíveis configurações de ataque. No modelo seguinte, porém, procuramos proceder de uma forma mais geral modelando em $Mefistofeles$ os diversos comportamentos que lhe permitem intrometer-se numa conversação. Tais comportamentos revelam das seguintes capacidades:

- actuar quer como activador quer como respondente;
- interceptar mensagens e responder-lhes, eventualmente alterando as partes não encriptadas;
- esconder mensagens;
- desencriptar mensagens com chaves que conheça e aceder eventualmente a novos *nonces* e chaves;
- enviar mensagens baseado no conhecimento adiquirido até então.

No modelo seguinte assumimos que os três participantes usam um mesmo canal a para comunicar. O activador, representado pelo processo $Init$, é practicamente igual ao da especificação anterior, mas agora com um comportamento recursivo. Note-se, ainda, que uma vez que o modelo usa um canal comum a todos os participantes, o activador tem o cuidado de re-encaminhar todas as mensagens que não são encriptadas com a sua chave pública.

O modelo do respondente é também similar ao anteriormente considerado, mas também com o requisito de re-encaminhar as mensagens não encriptadas com a sua chave pública. Além disso faz uma validação suplementar na chave recebida para determinar quem se lhe está a dirigir. Este teste não era necessário no modelo anterior onde cada par de participantes partilhava um canal único. Assim,

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

agent Resp(a,k,k1,k2,r1,r2,c1,c2)  = \
     a(x,y,z).( [x=k]AuxRespA<a,k,k1,k2,r1,r2,c1,c2,y,z> \
                + 'a<x,y,z>.Resp<a,k,k1,k2,r1,r2,c1,c2> )

agent AuxRespA(a,k,k1,k2,r1,r2,c1,c2,y,z) = \
      [z=k1]'r1.AuxRespB<a,k,c1,y,z> \
      + \
      [z=k2]'r2.AuxRespB<a,k,c2,y,z>

agent AuxRespB(a,k,c,y,z) = \
      (^ n)  ('a<z,y,n>.AuxRespC<a,k,c,n> )

agent AuxRespC(a,k,c,n) = \
      a(x,y,z).( [x=k]  ([y=n] 'c.0 + 'a<x,y,z>.AuxRespC<a,k,c,n> ) \
                        + \
                        'a<x,y,z>.AuxRespC<a,k,c,n> )
```

Mais interessante é o modelo para $Mefistofeles$. Este pode receber qualquer mensagem cifrada com a sua chave pública e enviar o seu contéudo para qualquer dos outros participantes. Se a mensagem não vem encriptada com a sua chave, re-envia-a no canal comum a. Além disso, $Mefistofeles$ tem agora a capacidade de activar ele próprio o protocolo enviando uma mensagem para o respondente (Rui).

Vejamos, então, a especificação mais geral de uma configuração com um intruso:

```
agent Ana(a,kA,kR,kJ,iAR,iAJ,icAR,icAJ)  = \
      Init<a,kA,kR,iAR,icAR> + Init<a,kA,kJ,iAJ,icAJ>

agent Rui(a,kR,kA,kJ,rAR,rJR,cAR,cJR) = \
      Resp<a,kR,kA,kJ,rAR,rJR,cAR,cJR>

agent Mefistofeles(a,kJ,kA,kR) = \
      ( a(x,y,z).( [x=kJ] AuxMefistofeles<a,kJ,kA,kR,y,z> \
                  + \
                  'a<x,y,z>. Mefistofeles<a,kJ,kA,kR> ) \
        + \
        (^ n)  ('a<kR,n,kJ>. Mefistofeles<a,kJ,kA,kR> ) )

agent AuxMefistofeles(a,kJ,kA,kR,n,k) = \
      'a<kA,n,k>.Mefistofeles<a,kJ,kA,kR>  +  'a<kR,n,k>.Mefistofeles<a,kJ,kA,kR>


agent GeneralS(iAR,iAJ,icAR,icAJ,rAR,rJR,cAR,cJR) =\
      (^ a,kA,kR,kJ) \
      ( Ana<a,kA,kR,kJ,iAR,iAJ,icAR,icAJ> \
        |   \
        Rui<a,kA,kA,kJ,rAR,rJR,cAR,cJR> \
        |   \
        Mefistofeles<a,kJ,kA,kR> )
```

Podemos, agora, verificar que é possível detectar o ataque de há pouco neste modelo mais geral. O procedimento é exactamente o mesmo:

```
MWB> prove GeneralS(iAR,iAJ,icAR,icAJ,rAR,rJR,cAR,cJR)
           <'iAJ><t><t><'rAR><t><t><'icAJ><t><t><'cAR>TT
Model Prover says: YES!
 (83 inferences)
MWB>
```

A propriedade que provamos sobre esta nova configuração é a mesma que utilizamos no primeiro modelo. Será possível escrever e testar uma propriedade mais geral que seja satisfeita em qualquer modelo se houver possibilidade de ataque ao protocolo aí representado?

Intuitivamente podemos dizer que há um ataque ao NSPK se numa sua execução o respondente Rui se comprometer numa sessão iniciada por Ana sem esta se ter previamente comprometido numa sessão com Rui. Podemos exprimir isto por uma propriedade de *animação fraca* onde a sinalização de Ana estar comprometida numa sessão com Rui, *i.e.,* `<'cAR>TT`, apareça numa computação em que nenhum evento `<'icAR>` tenha ocorrido. Assim, podemos testar

```
MWB>prove S(iAR,iAJ,icAR,icAJ,rAR,rJR,cAR,cJR)\
          mu X.( <'cAR>TT | (<t>X | <'iAR>X | <'iAJ>X | <'icAJ>X | <'rAR>X
                 | <'rJR>X | <'cAR>X | <'cJR>X) )
Model Prover says: YES!
 (13420 inferences)
MWB>
```

o que revela que existe efectivamente uma possibilidade de ataque ao NSPK.

Referências

[BAN90] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[Low96] G. Lowe. Braeaking and fixing the Needham-Schroeder public-key protocol using FDR. *Software — Concepts and Tool*, 17:93–102, 1996.

[NS78] R. M. Needham and M. Scroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(2):120–126, 1978.

[VM94] B. Victor and F. Moller. The mobility workbench: A tool for the π-calculus. In *Proceedings of CAV'94: Computer Aided Verification*. Springer Lect. Notes Comp. Sci. (818), 1994.

A Exercícios

Exercício 1

Responda às questões formuladas no texto desta Lição.

Exercício 2

Explique porque razão todo o modelo que satisfizer a propriedade

```
<'iAJ><t><t><'rAR><t><t><'icAJ><t><t><'cAR>TT
```

satisfaz também a propriedade

```
mu X.( <'cAR>TT | (<t>X | <'iAR>X | <'iAJ>X | <'icAJ>X | <'rAR>X
       | <'rJR>X | <'cAR>X | <'cJR>X) )
```

Exercício 3

Considere a seguinte versão corrigida do NSPK proposta em [Low96]:

$$
\begin{aligned}
Ana &\longrightarrow Rui \;:\; \{n_A, {K_A}^+\}_{{K_R}^+} \\
Rui &\longrightarrow Ana \;:\; \{n_A, n_R, {K_R}^+\}_{{K_A}^+} \\
Ana &\longrightarrow Rui \;:\; \{n_R\}_{{K_R}^+}
\end{aligned}
$$

1. Modele esta versão do protocolo no MWB e simule a sua execução.
2. Certifique-se, recorrendo ao MWB, que as fórmulas usadas para detectar a possibilidade de ataque não são satisfeitas neste modelo.
