Special topics in Security and Applied Logic

Specification and verification of security protocols

Master Year II, Sem. I, 2025-2026

Ioana Leuștean
FMI, UB

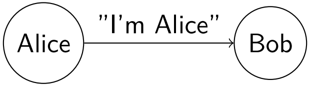
What is a security protocol?

- A security protocol is a set of rules and conventions defining an exchange of messages between two or more agents, with security-relevant goals, as:
  - establishing communication keys
  - agent authentication
  - ensuring secrecy

Security protocols are three-line programs
that people still manage to get wrong.- Roger Needham

- Even if the cryptography is perfect, a flawed protocol can affect the communication.

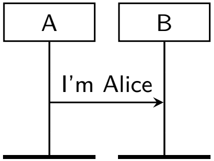
Modelling protocols



Alice-Bob notation

$A \longrightarrow B$: "I'm Alice"

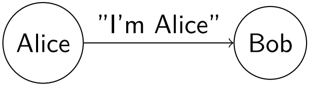
Message sequence charts MSC



General setting

- A security protocol describes few behaviours, which are called *roles* (initiator, responder, server . . .). A protocol is complete and unambigous.
- A *session* is a complete execution of a protocol.
- Several protocol sessions may be executed concurrently, an agent can execute any role, possibly in parallel.

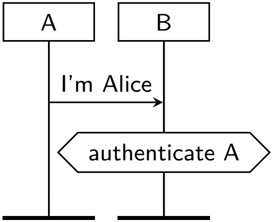
Modelling protocols



Alice-Bob notation

$A \longrightarrow B$: "I'm Alice"

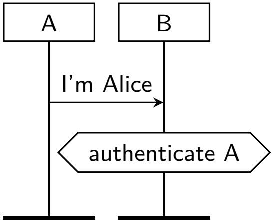
Message sequence charts MSC



For each role, the vertical axes denotes the order in which the events are executed. The hexagon contains the purpose of the protocol.

Modelling protocols

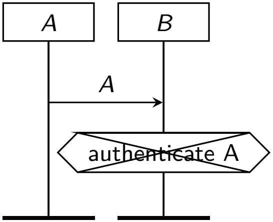
Message sequence charts MSC



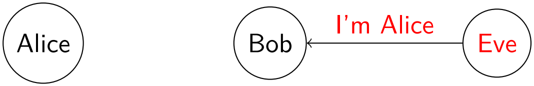
Asssume that *authenticate A* means: *B knows that he speaks with A*.

Is this protocol correct?

Modelling protocols



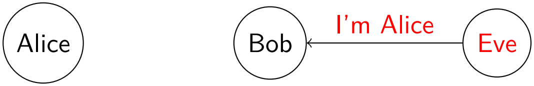
It is possible that Alice is impersonated by Eve!



The intruder Eve impersonates Alice!

$E(A) \longrightarrow B : A$

Modelling protocols



The intruder Eve impersonates Alice!

$E(A) \longrightarrow B : A$

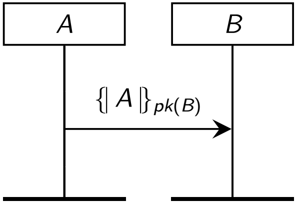
The hostile intruder controls the enviroment where the protocol is executed (the network):

- (s)he can intercept messages,
- (s)he can impersonate any honest agent.

“““““““““““‘

Modelling protocols

$A \longrightarrow B : \{\!|\, A \,|\!\}_{pk(B)}$



- Assume the message is encrypted with $pk(B)$, the public key of B.
- The attacker can intercept $\{\!|\, A \,|\!\}_{pk(B)}$ but (s)he cannot decrypt it!

The *black box* (or *perfect cryptography* assumption):
attackers can encrypt or decrypt messages only if they have the right keys.

The adversary model

The *Dolev-Yao* model:

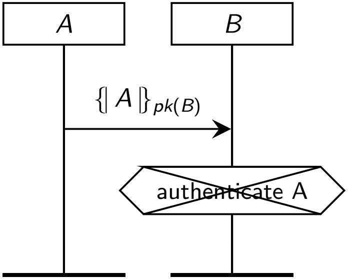
- The adversary has *complete information* over the protocol:
  - s(he) controls the communication channels,
  - s(he) can intercept messages,
  - s(he) has unlimited memory,
  - s(he) can impersonate agents,
  - s(he) can compose messages,
  - s(he) can play the role of an honest agent,

  ...

- but the cryptography *is perfect*:
  the adversary can decrypt a message only if s(he) knows the encryption key.

D. Dolev and A.C. Yao, *On the security of public key protocols*, IEEE Transactions on Information Theory, IT-29 (2): 198–208, 1983.

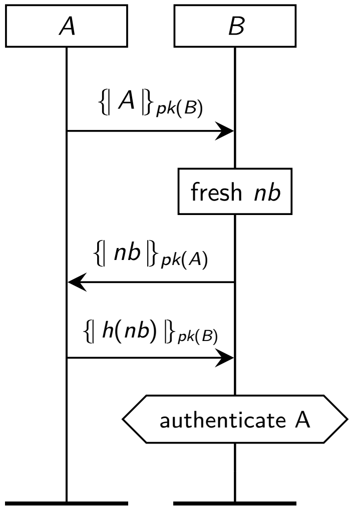
Modelling protocols



B cannot be sure that he speaks with A!

How do we solve the problem?

Nonce



- We use a *fresh value*, or a *nonce* (number used once).
- The atatcker can intercept the message $\{\!| nb |\!\}_{k(A,B)}$ but (s)he cannot extract *nb* because s(he) does not have the key.
- The attacker cannot send the answer expected by B.

Example:

Consider the following protocol:

$A \longrightarrow B : A, B, \{\!| A, na |\!\}_{pk(S)}$

$B \longrightarrow S : \{\!| A, na |\!\}_{pk(S)}, \{\!| B, na |\!\}_{pk(S)}$

What is wrong?

B cannot find na since it does not have the private key $sk(S)$, so B cannot compose the second message.

Example

Consider the following protocol:

$A \longrightarrow B : \{\!| A, B, na |\!\}_{k(A,B)}$

$B \longrightarrow A : \{\!| na, nb |\!\}_{k(A,B)}$

where $k(A,B)$ is a symmetric key.

In the following attack E impersonates B:

$A \longrightarrow E(B) : \{\!| A, B, na |\!\}_{k(A,B)}$

$E(B) \longrightarrow A : \{\!| A, B, na |\!\}_{k(A,B)}$

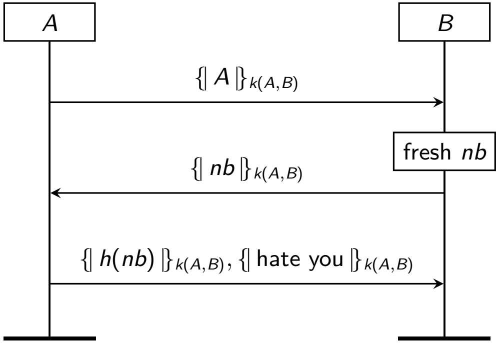
What do you notice?

The attack is *pointless* since A will immediately detect the fact that the received message is composed in a different way. We are not concerned with this kind of attacks!

An *attack* is an exchange of messages between the honest agents and the attacker such that the honest agents *do not detect an anomaly*.

Modelling protocols

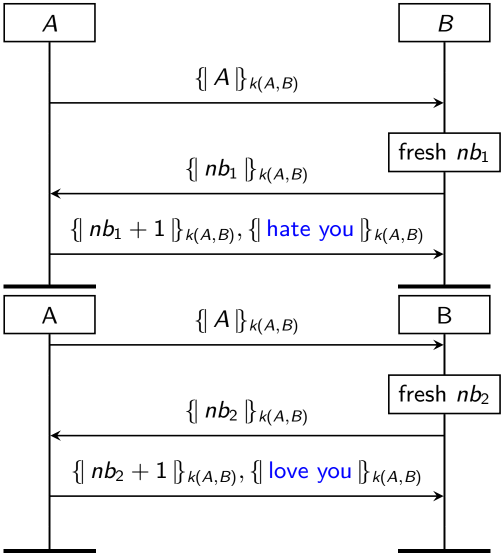
Consider the following protocol:



A
B
$\{\!|\, A \,|\!\}_{k(A,B)}$
fresh nb
$\{\!|\, nb \,|\!\}_{k(A,B)}$
$\{\!|\, h(nb) \,|\!\}_{k(A,B)}, \{\!|\, \text{hate you} \,|\!\}_{k(A,B)}$

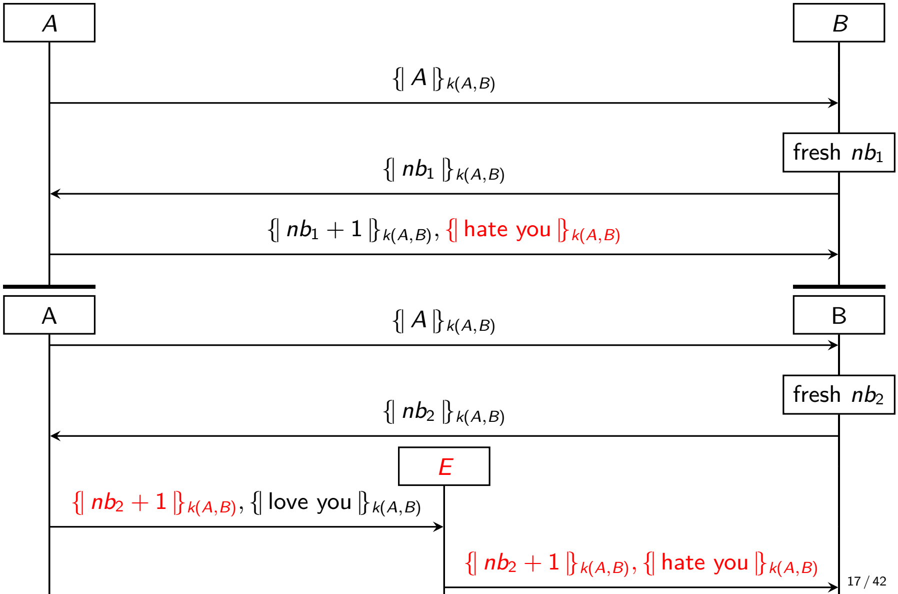
In this session, Bob should reach the conclusion that Alice hates him.

Modelling protocols: 2 sessions



A
B
$\{\!| A |\!\}_{k(A,B)}$
fresh nb_1
$\{\!| nb_1 |\!\}_{k(A,B)}$
$\{\!| nb_1 + 1 |\!\}_{k(A,B)}, \{\!|$ hate you $|\!\}_{k(A,B)}$
A
B
$\{\!| A |\!\}_{k(A,B)}$
fresh nb_2
$\{\!| nb_2 |\!\}_{k(A,B)}$
$\{\!| nb_2 + 1 |\!\}_{k(A,B)}, \{\!|$ love you $|\!\}_{k(A,B)}$

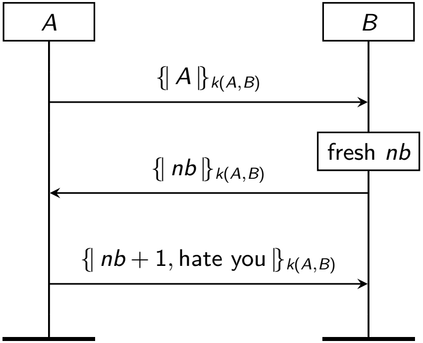
A replay attack



A
B
$\{\!|\, A \,|\!\}_{k(A,B)}$
fresh nb_1
$\{\!|\, nb_1 \,|\!\}_{k(A,B)}$
$\{\!|\, nb_1 + 1 \,|\!\}_{k(A,B)}, \{\!|\, \text{hate you} \,|\!\}_{k(A,B)}$
A
B
$\{\!|\, A \,|\!\}_{k(A,B)}$
fresh nb_2
$\{\!|\, nb_2 \,|\!\}_{k(A,B)}$
E
$\{\!|\, nb_2 + 1 \,|\!\}_{k(A,B)}, \{\!|\, \text{love you} \,|\!\}_{k(A,B)}$
$\{\!|\, nb_2 + 1 \,|\!\}_{k(A,B)}, \{\!|\, \text{hate you} \,|\!\}_{k(A,B)}$

Modelling protocols

- În the previous example, the attacker composes a new message from old pieces that (s)he memorized.
- A better version would be:


A
B
$\{\!| A |\!\}_{k(A,B)}$
fresh nb
$\{\!| nb |\!\}_{k(A,B)}$
$\{\!| nb + 1, \text{hate you} |\!\}_{k(A,B)}$

The Needham-Schroeder protocol

- In many protocols the symmetric key is a secret established using a public key protocol.

Needham-Scroeder Public Key Protocol (NSPK)

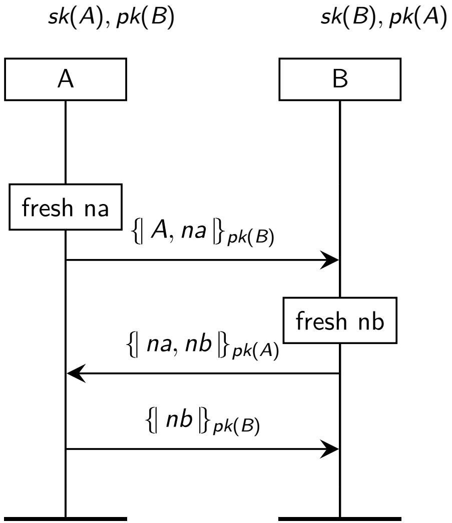
$$\begin{aligned} A \longrightarrow B: &\quad \{\!| A, na |\!\}_{pk(B)} \\ B \longrightarrow A: &\quad \{\!| na, nb |\!\}_{pk(A)} \\ A \longrightarrow B: &\quad \{\!| nb |\!\}_{pk(B)} \end{aligned}$$

The purpose is the mutual authentication of the two participants. After a successful execution:

- Alice and Bob should be ensured they have been communicating each other (all the messages have been sent and received by the communication partner),
- *na* and *nb* are known only by Alice and Bob (the protocols guarantees the confidentiality of the nonces *na* and *nb*).

R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, 21, pp.393-399, 1978

Needham-Scroeder Public Key Protocol (1978)


$sk(A), pk(B)$
$sk(B), pk(A)$
A
B
fresh na
$\{\!|\, A, na \,|\!\}_{pk(B)}$
fresh nb
$\{\!|\, na, nb \,|\!\}_{pk(A)}$
$\{\!|\, nb \,|\!\}_{pk(B)}$

NSPK

Needham-Screoeder Public Key Protocol (NSPK)

$$\begin{array}{lcr}
A \xrightarrow{\{\!|\, A, na \,|\!\}_{pk(B)}} & \qquad & \xrightarrow{\{\!|\, x, y \,|\!\}_{pk(B)}} B \\
A \xleftarrow{\{\!|\, na, z \,|\!\}_{pk(A)}} & \qquad & \xleftarrow{\{\!|\, y, nb \,|\!\}_{pk(x)}} B \\
A \xrightarrow{\{\!|\, z \,|\!\}_{pk(B)}} & \qquad & \xrightarrow{\{\!|\, nb \,|\!\}_{pk(B)}} B
\end{array}$$

NSPK

The "man in the middle" attack

$$
\begin{array}{lcccc}
A & \xrightarrow{\{\!|\, A,na \,|\!\}_{pk(E)}} & E & & \\
 & & E(A) & \xrightarrow{\{\!|\, A,na \,|\!\}_{pk(B)}} & B \\
 & & E(A) & \xleftarrow{\{\!|\, na,nb \,|\!\}_{pk(A)}} & B \\
A & \xleftarrow{\{\!|\, na,nb \,|\!\}_{pk(A)}} & E & & \\
A & \xrightarrow{\{\!|\, nb \,|\!\}_{pk(E)}} & E & & \\
 & & E(A) & \xrightarrow{\{\!|\, nb \,|\!\}_{pk(B)}} & B
\end{array}
$$

The "man in the middle" attack on NSPK

$$\begin{tikzcd}
A \arrow[r, "\{\mid A, na \mid\}_{pk(E)}"] & E & \\
 & E(A) \arrow[r, "\{\mid A, na \mid\}_{pk(B)}"] & B \\
 & E(A) & B \arrow[l, "\{\mid na, nb \mid\}_{pk(A)}"'] \\
A & E \arrow[l, "\{\mid na, nb \mid\}_{pk(A)}"'] & \\
A \arrow[r, "\{\mid nb \mid\}_{pk(E)}"] & E & \\
 & E(A) \arrow[r, "\{\mid nb \mid\}_{pk(B)}"] & B
\end{tikzcd}$$

- A initiates a session with the (dishonest) E,
- E impersonates A when communicating with B,
- B belives that he successfully communicated with A but in fact he communicated with E. The attack violates authentication from B's point of view, as well as the secrecy of nb (E knows nb).

The Needham-Schroeder-Lowe protocol (1996)

Fixing NSPK such that the "man in the middle" attack is prevented [Lowe 1996]

$$\begin{array}{ll} A \longrightarrow B: & \{\!|\, A, na \,|\!\}_{pk(B)} \\ B \longrightarrow A: & \{\!|\, na, nb, B \,|\!\}_{pk(A)} \\ A \longrightarrow B: & \{\!|\, nb \,|\!\}_{pk(B)} \end{array}$$

- A should receive $\{\!|\, na, nb, E \,|\!\}_{pk(A)}$, but E cannot find nb.
- E can send $\{\!|\, na, nb, B \,|\!\}_{pk(A)}$ to A, but this message does not respect the pattern A waits for.

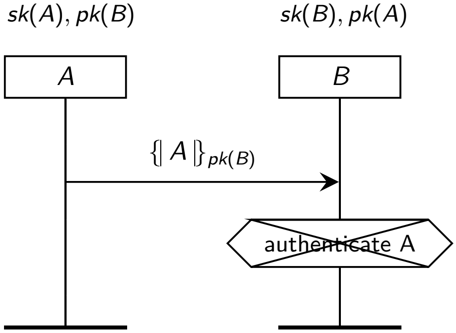
Formal analysis of security protocols

- In formal analysis we define and analyze a protocol within a consistent mathematical theory.
- One studies abstract versions of real protocols (for example the (real, computer-network authentication) protocol Kerberos is based on the (academic) protocol Needham-Scroeder.
- Verification based on:
  - logical systems, for example BAN logic,
  - operational semantics,
  - model-checking tools: Proverif, Scyther, Tamarin, ...
  - ...

Example: Simple Protocol

$A \longrightarrow B : \{\!|\, A \,|\!\}_{pk(B)}$

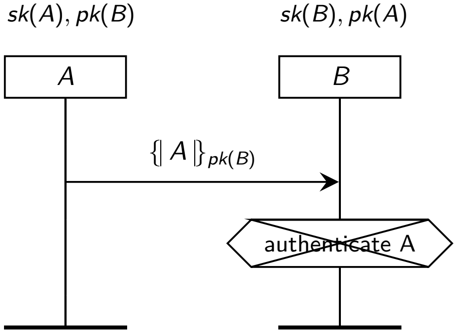
We want to check authentication formally!



Example: Simple Protocol

What does authentication mean?

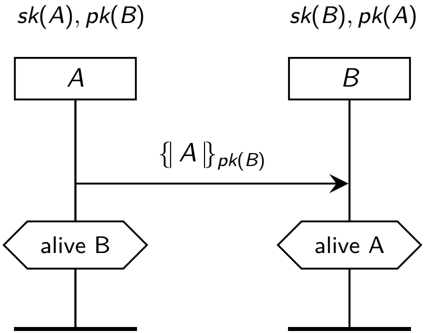
> "In its most basic form, authentication is a simple statement about the existence of communication partner.
> [...]
> *Aliveness* is a form of authentication that aims to establish that an intended communication partner has executed some event."



[OVSP] Cremers C. and Mauw S. Operational Semantics and Verification of Security Protocols. Springer, 2012.

Scyther: Simple Protocol

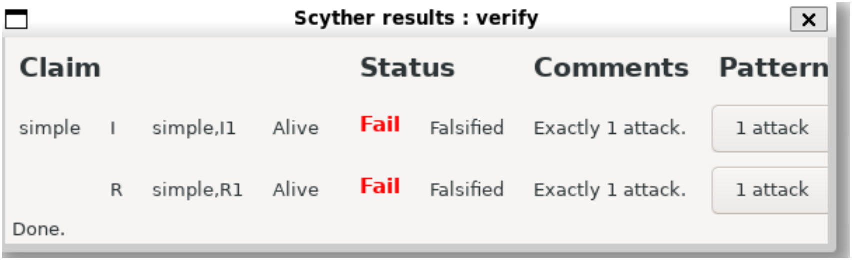
```
protocol simple(I,R)
{
role I
   {
     send_1(I,R, {I}pk(R) );

     claim(I,Alive);
    }
role R
    {
      recv_1(I,R, {I}pk(R));

      claim(R,Alive);
     }
}
```



The claims are locally analyzed!

Scyther: Simple Protocol

```
protocol simple(I,R)
{role I
   {send_1(I,R, {I}pk(R) );
    claim(I,Alive);}
role R
    {recv_1(I,R, {I}pk(R));
     claim(R,Alive);
     }}
```

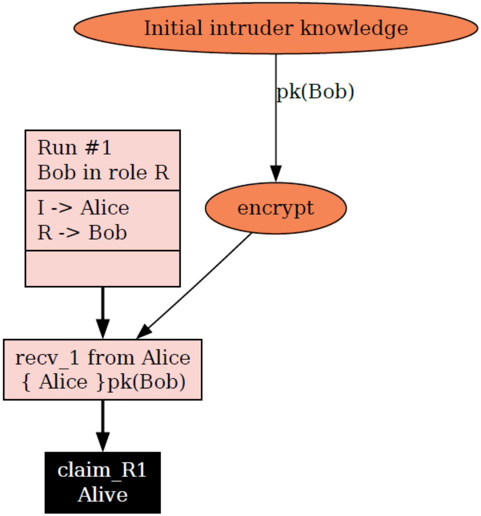


Scyther results : verify

Claim				Status		Comments	Pattern
simple	I	simple,I1	Alive	Fail	Falsified	Exactly 1 attack.	1 attack
	R	simple,R1	Alive	Fail	Falsified	Exactly 1 attack.	1 attack

Done.

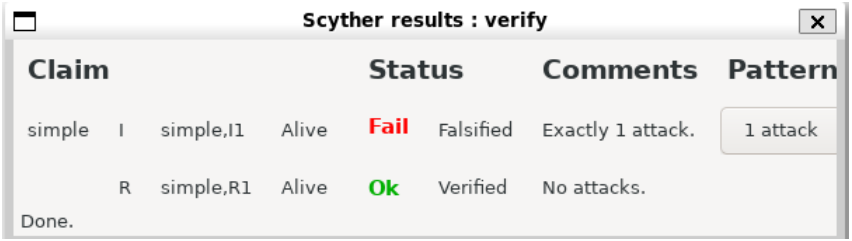
Scyther: Simple Protocol



[Id 2] Protocol simple, role R, claim type Alive

Scyther: Simple Protocol

```
protocol simple(I,R)
{role I
   {send_1(I,R, {I}sk(I) );
    claim(I,Alive);}
role R
    {recv_1(I,R, {I}sk(I));
     claim(R,Alive);
     }}
```



Scyther results : verify

Claim				Status		Comments	Pattern
simple	I	simple,I1	Alive	Fail	Falsified	Exactly 1 attack.	1 attack
	R	simple,R1	Alive	Ok	Verified	No attacks.	

Done.

Operational semantics - references

- 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.

ProVerif

"The ProVerif tool takes protocols written in a variant of **the applied pi calculus** as input together with a security property to verify. The protocol is then **automatically translated into a set of first-order Horn clauses** and the properties are translated into derivability queries."

Véronique Cortier, Steve Kremer. Formal Models and Techniques for Analyzing Security Protocols: A Tutorial. Foundations and Trends in Programming Languages, Now Publishers, 2014, 1 (3), pp.117. https://hal.archives-ouvertes.fr/hal-01090874

```
process
    new skA;
    let pkA = pk(skA) in
        let hostA = host(pkA) in out(c, pkA);
    new skB;
    let pkB = pk(skB) in out(c, pkB);
    ((!processA) | (!processB))
```

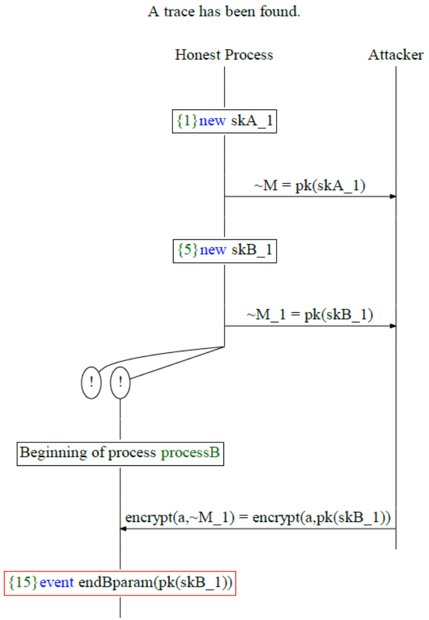
ProVerif: π-calculus for the Simple Protocol

```
query ev:endBparam(x) ==> ev:beginBparam(x).

let processA =
   in(c, pk2);
   event beginBparam(pk2);
   out(c, encrypt(host(pk2), pk2)).

let processB =
    in(c, km);
    let hostA = decrypt(km,skB) in
              event endBparam(pk(skB)).
```

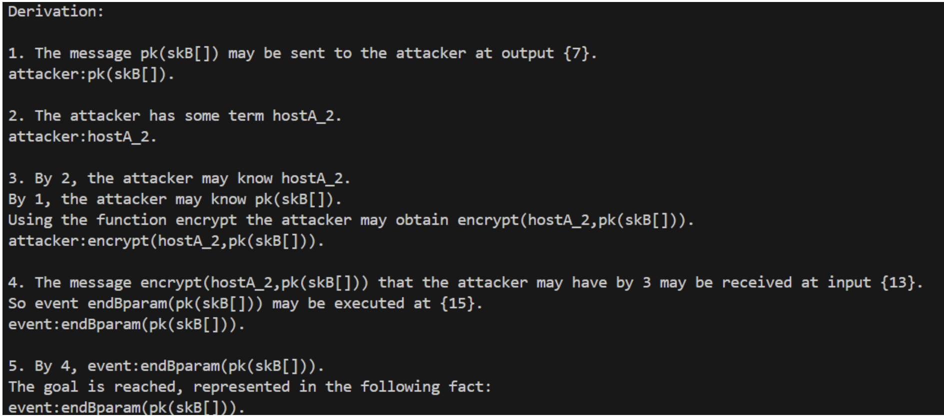
ProVerif: π-calculus for the Simple Protocol

```
query ev:endBparam(x) ==> ev:beginBparam(x).

let processA =

let processB =

process
    new skA;
    let pkA = pk(skA) in
        let hostA = host(pkA) in out(c, pkA);
    new skB;
    let pkB = pk(skB) in out(c, pkB);
    ((!processA) | (!processB))

⋮
A trace has been found.
RESULT: ev:endBparam(x) ==> ev:beginBparam(x) is false.
```

ProVerif: Simple Protocol



ProVerif: Simple Protocol



```
Derivation:

1. The message pk(skB[]) may be sent to the attacker at output {7}.
attacker:pk(skB[]).

2. The attacker has some term hostA_2.
attacker:hostA_2.

3. By 2, the attacker may know hostA_2.
By 1, the attacker may know pk(skB[]).
Using the function encrypt the attacker may obtain encrypt(hostA_2,pk(skB[])).
attacker:encrypt(hostA_2,pk(skB[])).

4. The message encrypt(hostA_2,pk(skB[])) that the attacker may have by 3 may be received at input {13}.
So event endBparam(pk(skB[])) may be executed at {15}.
event:endBparam(pk(skB[])).

5. By 4, event:endBparam(pk(skB[])).
The goal is reached, represented in the following fact:
event:endBparam(pk(skB[])).
```

ProVerif - references

- https://bblanche.gitlabpages.inria.fr/proverif/
- Véronique Cortier, Steve Kremer. Formal Models and Techniques for Analyzing Security Protocols: A Tutorial. Foundations and Trends in Programming Languages, Now Publishers, 2014, 1 (3), pp.117. https://hal.archives-ouvertes.fr/hal-01090874
- Bruno Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. Foundations and Trends® in Privacy and Security , Now publishers inc, 2016, 1 (1-2), pp.1 - 135.https://hal.inria.fr/hal-01423760/

BAN Logic: A Logic of Authentication

"A simple logic has allowed us to describe the beliefs of trustworthy parties involved in authentication protocols and the evolution of these beliefs as a consequence of communication."

Burrows, M., Abadi. M., & Needham, R. (1990) *A Logic of Authentication. ACM Transactions on Computer Systems*, Vol. 8, No. 1: 18–36. 1990.

BAN logic has particular operators denoting agents actions and beliefs:

- $P \mid\equiv X$: Agent P **believes** that X.
- $P \triangleleft X$: Agent P (**receives**) **sees** message X.
- $P \mid\sim X$: Agent P once **said** that X.

The Simple Protocol: assumptions

The simple protocol step

$A \longrightarrow B : \{\!| X |\!\}_{sk(A)}$

is represented (idealized) in BAN logic by

$A \longrightarrow B : \{X\}_{K_A^{-1}}$

and further translated into the following assumption:

$B \triangleleft \{X\}_{K_A^{-1}}$ (B receives the message X encrypted with the secret key of A)

We can also assume that

$B \mid\equiv \overset{K_A}{\mapsto} A$ (B beliefs that K_A is the public key of A)

BAN Logic: deduction rules

Message meaning rules for public keys:

$$MM-PK \;\; \frac{P \mid\equiv \mapsto^{K} Q, P \triangleleft \{X\}_{K^{-1}}}{P \mid\equiv (Q \mid\sim X)}$$

$$MM-SK \;\; \frac{P \textbf{ believes } \mapsto^{K} Q, P \textbf{ sees } \{X\}_{K^{-1}}}{P \textbf{ believes } (Q \textbf{ said } X)}$$

Using this rule, from our assumptions $B \triangleleft \{X\}_{K_A^{-1}}$ and $B \mid\equiv \mapsto^{K_A} A$ we infer that $B \mid\equiv (A \mid\sim X)$ (B believes that A said X), which means that A made an action.

Note that for the simple step $A \longrightarrow B : \{\!| X |\!\}_{pk(B)}$ we are **not able** to make a similar deduction.

Thank you!