RAPPORT TECHNIQUE EVA

EVA test base

Date	:	November 20, 2001
Authors	:	Dominique Bolignano (Trusted Logic) Francesca Fiorenza (Trusted Logic) Florent Jacquemard (Trusted Logic) Daniel Le Métayer (Trusted Logic)
Title	:	EVA test base
Report number / Version	:	4/ 1.17

TRUSTED LOGIC S.A.
5 rue du Bailliage
78000 Versailles, France
www.trusted-logic.fr

Laboratoire Spécification Vérification
CNRS UMR 8643, ENS Cachan
61, avenue du président-Wilson
94235 Cachan Cedex, France
www.lsv.ens-cachan.fr

Laboratoire Verimag
CNRS UMR 5104,
Univ. Joseph Fourier, INPG
2 av. de Vignate,
38610 Gières, France
www-verimag.imag.fr

Notes : Compatibility with others EVA documents and tools.

This version 1.17 of the EVA technical report number 4 is compatible with:

- the EVA technical report 1, version 3.17 (*Langage de spécification de protocoles cryptographiques de EVA : syntaxe concrète*);
- the EVA technical report 2, version 2.3 (*Langage de spécification de protocoles cryptographiques de EVA: syntaxe abstraite et sémantique*);
- the translator version 2;
- the test base version 2.

Contents

0 Introduction

0.1 The test base

0.1.1 Input files

The EVA test base contains a set of protocol specifications, in the syntax described in the EVA report [JLM01]. Every protocol specification is contained in a text file with the suffix `.eva`.

The base is organized in directories, one directory for each protocol. Distinct versions of the same protocol belong to distinct directories. Each directory contains exactly one `.eva` file specifying the protocol of the directory (we call it protocol file) and possibly several `.eva` files describing known attacks (we call them attack files). Attack files are described in more detail below.

The policy for the file names in the test base is the following:

- identifiers contain alphanumericals `[a-z]` or `[A-Z]` or `[0-9]` or `_`
- the names for protocol files have the form: *protocol file identifier*`.eva`
- the names for attack files have the form: *protocol file identifier*-*attack identifier*`.eva`

The *attack identifier* is generally the name of the authors of the paper in which the attack was first described.

0.1.2 Attacks

We associate to some known attacks on protocols the corresponding EVA specifications which should lead to an error detection when processed with a model checker. Such specifications are described in attack `.eva` files. Each of these specifications contains the protocol messages and declarations, which are the same as in the protocol file in the same directory. They contain also the properties negated by the attack and some information about the possible sessions of the protocol, to narrow the search for the attacks scenarios.

0.1.3 Properties

At the bottom of each protocol file, we provide properties defining the goals of the protocol, as deduced from the reference documents.

We only give properties that are not a direct consequence of the semantics of protocols (as defined in the EVA reports [JLM01],[GL01a] and [GL01b]). For instance, `agreement(A,B,X,X)` may be equivalent to `aliveness(A,B)` if, when `A` and `B` finish a protocol execution, semantics of the protocol imposes that the two values of `X` in `A` and `B` respective environments are equal (otherwise the message would have been rejected). However, there exist some attacks which contradict `agreement` and not `aliveness`, see for instance Section 15.

0.1.4 Axioms

The following axioms for the encryption and decryption of messages are assumed:

$$\begin{array}{rcl} \{\{x\}_y\}_y & = & x \\ \{\{x\}_{PK(y)}\}_{SK(y)} & = & x \\ \{\{x\}_{SK(y)}\}_{PK(y)} & = & x \end{array}$$

In the last two axioms, PK and SK are declared in a `keypair`.

Some additional axioms on the constructors may be given in the protocol specifications. However, only two (classical) axioms are used in the EVA protocol base:

1. The exponential in a finite group, for the Diffie Helman key schema (Diffie Helman § 20, Horn Preneel § 24).

$$\begin{array}{rcl} kap(kap(g,x),y) & = & kap(kap(g,y),x) \\ kas(kas(g,x,p),y,p) & = & kas(kas(g,y,p),x,p) \end{array}$$

   The interpretation of these constructors is $kas(g,x,p) = kap(g,x) = g^x \, mod \, p$ (p is assumed given in kap).

2. Xor (Gong § 13)

$$(x \oplus y) \oplus x = y, (y \oplus x) \oplus x = y$$

0.2 The document

In this document, we describe each protocol of the test base (except for the SSL protocol which is not included for space considerations) using the following presentation:

- Cryptosystem: the cryptographic techniques used in the protocol (including symmetric key cryptography, public key cryptography, certificates one way functions, trusted servers...).
- Description: a short description of the purpose of the protocol
- Reference: the document where the protocol was first defined
- The specification of the protocol
- Attack: (optional) the description, references and specification of some known attacks on the protocol (see 0.1.2)
- Analyses: (optional) the references of some papers reporting a successful verification of the protocol.

The last two items are provided only when appropriate (i.e. when there are some known attacks or analyses).

1 Yahalom

Cryptosystem. Symmetric keys and trusted server.

Description. The purpose of the Yahalom protocol is double: distribution of a fresh symmetric shared key by a trusted server and mutual authentication.

The following version of the Yahalom protocol is the one presented in [CJ97].

Protocol 1 `Yahalom/Yahalom.eva`

```
Yahalom
  A, B, S :       principal
  Na, Nb :        number
  Kas, Kbs, Kab : Key
  A knows A, B, S, Kas
  B knows B, A, S, Kbs
  S knows S, A, B, Kas, Kbs
  {
   1. A -> B : A, Na
   2. B -> S : B, { A, Na, Nb }_Kbs
   3. S -> A : { B, Kab, Na, Nb }_Kas, { A, Kab }_Kbs
   4. A -> B : { A, Kab }_Kbs, { Nb }_Kab
  }
  claim Secret(Kab)
        Agreement(A,B,Nb,Nb)
        Agreement(A,B,Kab,Kab)
```

Attack. Replay attack with type error, see [CJ97].

$$\begin{array}{llcll} 1 & I(A) & \rightarrow & B: & A, N_a \\ 2 & B & \rightarrow & I(S): & B, \{A, N_a, N_b\}_{K_{bs}} \\ 3 & & & & \mathit{Omitted} \\ 4 & I(A) & \rightarrow & B: & \{A, N_a, N_b\}_{K_{bs}}, \{N_b\}_{N_a, N_b} \end{array}$$

With a weak type checking, the principal B may consider the pair $\langle N_a, N_b \rangle$ as the new session key K_{ab}.

Analyses. [Pau01]

Attack 1 `Yahalom/Yahalom-CJ.eva`

```
Yahalom
  A, B, S :       principal
  Na, Nb :        number
  Kas, Kbs, Kab : Key
  A knows A, B, S, Kas
  B knows B, A, S, Kbs
  S knows S, A, B, Kas, Kbs
  {
   1. A -> B : A, Na
   2. B -> S : B, { A, Na, Nb }_Kbs
   3. S -> A : { B, Kab, Na, Nb }_Kas, { A, Kab }_Kbs
   4. A -> B : { A, Kab }_Kbs, { Nb }_Kab
  }
 session A = a, B = b, S = s
 claim Aliveness(A, B)
```

2 Yahalom-Paulson

Cryptosystem. Symmetric keys and trusted server.

Description. Paulson's ([Pau01]) modified version of the Yahalom protocol, following the suggestion of [BAN89].

Protocol 2 `YahalomPaulson/YahalomPaulson.eva`

```
Yahalom_Paulson
 A, B, S : principal
 Na, Nb  : number
 Kas, Kbs, Kab  : Key
 A knows  A, B, S, Kas, Kab
 B knows  B, A, S, Kbs, Kab
 S knows  S, A, B, Kas, Kbs
 {
  1. A -> B :  A, Na
  2. B -> S :  B, Nb, { A, Na }_Kbs
  3. S -> A :  Nb, { B, Kab, Na }_Kas, { A, B, Kab, Nb }_Kbs
  4. A -> B :  { A, B, Kab, Nb }_Kbs, { Nb }_Kab
 }
 claim Secret(Kab)
       Agreement(A,B,Nb,Nb)
       Agreement(A,B,Kab,Kab)
```

Analyse. [Pau01]

3 Yahalom-BAN

Cryptosystem. Symmetric keys and trusted server.

Description. A modified form of the Yahalom protocol studied in [BAN89].

Protocol 3 YahalomBAN/YahalomBAN.eva

```
Yahalom_BAN
 A, B, S :  principal
 Na, Nb  :  number
 Kas, Kbs, Kab : Key
 A knows   A, B, S, Kas, Kab
 B knows   B, A, S, Kbs, Kab
 S knows   S, A, B, Kas, Kbs
 {
  1. A -> B :  A, Na
  2. B -> S :  B, Nb, { A, Na }_Kbs
  3. S -> A :  Nb, { B, Kab, Na }_Kas, { A, Kab, Nb }_Kbs
  4. A -> B :  { A, Kab }_Kbs, { Nb }_Kab
 }
claim Secret(Kab)
      Secret(Nb)
      Agreement(A,B,Nb,Nb)
      Agreement(A,B,Kab,Kab)
```

Attack 1. There an attack in [Pau01], for which it is assumed that the intruder has managed to crack an old certificate $\{A, Kab\}_{Kbs}$ of B, extracting Kab.

Note that message 4 differs from the regular run of the protocol.

1	$I(A)$	$\rightarrow$	$B:$	A, Na
2	B	$\rightarrow$	$I(S):$	$B, Nb, \{A, Na\}Kbs$
3				*Omitted*
4	$I(A)$	$\rightarrow$	$B:$	$\{A, Kab\}_{Kbs}, \{Nb\}_{Kab}$

Attack 2. Replay attack with interleaving and type error in [Syv94].

$a.1$	A	$\rightarrow$	$I(B):$	A, N_a
$b.1$	$I(A)$	$\rightarrow$	$B:$	A, N_a
$b.2$	B	$\rightarrow$	$I(S):$	$B, N_b, \{A, N_a\}_{Kbs}$
$c.1$	$I(A)$	$\rightarrow$	$B:$	$A, \langle N_a, N_b\rangle$
$c.2$	B	$\rightarrow$	$I(S):$	$B, N_b', \{A, N_a, N_b\}_{Kbs}$
$a.3$				*Omitted*
$a.4$	$I(A)$	$\rightarrow$	$B:$	$\{A, N_a(= Kab), N_b\}Kbs, N_{b N_a}$

Attack 2 `YahalomBAN/YahalomBAN-Paulson.eva`

```
Yahalom_BAN_Paulson
 A, B, S :  principal
 Na, Nb  :  number
 Kas, Kbs, Kab : Key
 A knows   A, B, S, Kas, Kab
 B knows   B, A, S, Kbs, Kab
 S knows   S, A, B, Kas, Kbs
 intruder knows  Kab, { A, Kab}_Kbs
 {
  1. A -> B :  A, Na
  2. B -> S :  B, Nb, { A, Na }_Kbs
  3. S -> A :  Nb, { B, Kab, Na }_Kas, { A, Kab, Nb }_Kbs
  4. A -> B :  { A, Kab }_Kbs, { Nb }_Kab
 }
 session A = a, B = b, S = s
 claim Secret(Nb)
       Aliveness(A,B)
```

Attack 3 `YahalomBAN/YahalomBAN-Syverson.eva`

```
Yahalom_BAN_Syverson
 A, B, S :  principal
 Na, Nb  :  number
 Kas, Kbs, Kab : Key
 A knows   A, B, S, Kas, Kab
 B knows   B, A, S, Kbs, Kab
 S knows   S, A, B, Kas, Kbs
 {
  1. A -> B :  A, Na
  2. B -> S :  B, Nb, { A, Na }_Kbs
  3. S -> A :  Nb, { B, Kab, Na }_Kas, { A, Kab, Nb }_Kbs
  4. A -> B :  { A, Kab, Nb }_Kbs, { Nb }_Kab
 }

 session A = a, B = b, S = s
 session A = a, B = b, S = s
 session A = a, B = b, S = s

 claim Aliveness(A,B)
```

4 Yahalom-Lowe

Cryptosystem. Symmetric keys and trusted server.

Description. Another version of the Yahalom protocol presented in [Low98] to illustrate their verification technique. This modification on the original version is supposed to prevent the attacks otherwise possible (see § 1).

Reference: [Low98]

Protocol 4 `YahalomLowe/YahalomLowe.eva`

```
Yahalom_Lowe
 A, B, S :       principal
 Na, Nb :        number
 Kas, Kbs, Kab : Key
 A knows         A, B, S, Kas, Kab
 B knows         B, A, S, Kbs, Kab
 S knows         S, A, B, Kas, Kbs
 {
  1. A -> B : A, Na
  2. B -> S : { A, Na, Nb }_Kbs
  3. S -> A : { B, Kab, Na, Nb }_Kas
  4. S -> B : { A, Kab }_Kbs
  5. A -> B : { A, B, S, Nb }_Kab
 }
 session A = a, B = b, S = s
 claim Secret(Kab)
       Secret(Nb)
       Agreement(A,B,Nb,Nb)
       Agreement(A,B,Kab,Kab)
```

Analyses. In [Low98], no attacks were detected using CASPER/FDR, for a small system with a single instance for each of A, B, and S.

5 Neumann Stubblebine

Cryptosystem. Symmetric keys and trusted server.

Description. Session key exchange (messages 1-4) and repeated authentication (messages 5-7). The session key exchange part is inspired by the Yahalom pretocol (see § 1), with the addition of timestamps.

Reference: [NS93]

Protocol 5 `NeumannStubblebine/NeumannStubblebine.eva`

```
Neumann_Stubblebine
 A, B, S :          principal
 Na, Ma, Nb, Mb : number
 Kas, Kbs, Kab :    key
 Ta, Tb :           Time
 A knows            A, B, S, Kas
 B knows            B, A, S, Kbs
 S knows            S, A, B, Kas, Kbs
 {
  1.  A -> B : A, Na
  2.  B -> S : B, { A, Na, Tb }_Kbs, Nb
  3.  S -> A : { B, Na, Kab, Tb }_Kas, { A, Kab, Tb }_Kbs, Nb
  4.  A -> B : { A, Kab, Tb }_Kbs, { Nb }_Kab
  5.  A -> B : Ma, { A, Kab, Tb }_Kbs
  6.  B -> A : Mb, { Ma }_Kab
  7.  A -> B : { Mb }_Kab
 }
claim Secret(Kab)
      Agreement(A,B,Kab,Kab)
      Agreement(A,B,Nb,Nb)
      Agreement(A,B,Ma,Ma)
      Agreement(A,B,Mb,Mb)
```

Attack 1. From [HLL+95], see also § 3 above for the first 4 messages.

$$
\begin{array}{llcll}
1 & I(A) & \rightarrow & B: & A, N_a \\
2 & B & \rightarrow & I(S): & B, \{A, N_a, T_b\}_{K_{bs}}, N_b \\
3 & & & & \textit{Omitted} \\
4 & I(A) & \rightarrow & B: & \{A, N_a(=K_{ab}), T_b\}_{K_{bs}}, \{N_b\}_{N_a} \\
5 & I(A) & \rightarrow & B: & N'_a, \{A, N_a(=K_{ab}), Tb\}_{K_{bs}} \\
6 & B & \rightarrow & I(A): & N'_b, \{N'_a\}_{N_a} \\
7 & I(A) & \rightarrow & B: & \{N'_b\}_{N_a}
\end{array}
$$

Attack 2. From [HLL+95]. Attack on the repeated authentication part, assuming K_{ab} has been recorded in a previous legitimate run of the protocol.

$$
\begin{array}{llcll}
a.5 & I(A) & \rightarrow & B: & N'_a, \{A, K_{ab}, Tb\}_{K_{bs}} \\
a.6 & B & \rightarrow & I(A): & N'_b, \{N'_a\}_{K_{ab}} \\
b.5 & I(A) & \rightarrow & B: & N'_b, \{A, K_{ab}, Tb\}_{K_{bs}} \\
b.6 & B & \rightarrow & I(A): & N''_b, \{N'_b\}_{K_{ab}} \\
a.7 & I(A) & \rightarrow & B: & \{N'_b\}_{K_{ab}}
\end{array}
$$

Attack 3. From [Wei99].

$$
\begin{array}{llcll}
a.2 & I(B) & \rightarrow & S: & B, \{A, K_{ab}^0, T_b\}_{K_{bs}}, N_b \\
a.3 & S & \rightarrow & I(A): & \{B, N_a, K_{ab}^1, T_b\}_{K_{as}}, \{A, K_{ab}^1, T_b\}_{K_{bs}}, N_b \\
b.2 & I(B) & \rightarrow & S): & B, \{A, K_{ab}^1, T_b\}_{K_{bs}}, N_b \\
b.3 & S & \rightarrow & I(A): & \{B, N_a, K_{ab}^2, T_b\}_{K_{as}}, \{A, K_{ab}^2, T_b\}_{K_{bs}}, N_b \\
\vdots & & & &
\end{array}
$$

The intruder can get as many ciphers $\{A, K_{ab}^i, T_b\}_{K_{bs}}$ as needed to start a known plaintext attack in order to break K_{bs}.

Analyses. [Wei99]: Analysis of a modified form of the protocol where the server S does not accept a nonce as a key (using type check).

6 Needham Schroeder Symmetric Key Protocol

Cryptosystem. Symmetric keys and trusted server.

Description. Distribution of a shared symmetric key `Kab` by a trusted server `S` (messages `1-3`) and mutual authentification (messages `4`,`5`).

Reference: [NS78a].

Protocol 6 `NeedhamSchroederSymetricKey/NSSK.eva`

```
NSSK
 A, B, S :       principal
 Na, Nb :        number
 Kas, Kbs, Kab : key
 dec(number) :   number
 A knows         A, B, Kas, dec
 B knows         B, Kbs, dec
 S knows         A, B, Kas, Kbs
 {
  1. A -> S : A, B, Na
  2. S -> A : { Na, B, Kab, { Kab, A }_Kbs }_Kas
  3. A -> B : { Kab,A }_Kbs
  4. B -> A : { Nb }_Kab
  5. A -> B : { dec(Nb) }_Kab
 }
 claim Secret(Kab)
       Agreement(C,A,Kab,Kab)
```

Attack. Denning Sacco [DS81]. Assume that I has recorded the session a and that K_{ab} is compromised.

$$
\begin{array}{lrcll}
a.1 & A & \rightarrow & S: & A, B, N_a \\
a.2 & S & \rightarrow & A: & \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \\
a.3 & A & \rightarrow & I(B): & \{K_{ab}, A\}_{K_{bs}} \text{ and } K_{ab} \text{ is compromised} \\
b.3 & I(A) & \rightarrow & B: & \{K_{ab}, A\}_{K_{bs}} \\
b.4 & B & \rightarrow & I(A): & \{N_b\}_{K_{ab}} \\
b.5 & I(A) & \rightarrow & B: & \{dec(N_b)\}_{K_{ab}}
\end{array}
$$

and B thinks he shared the secret key K_{ab} with A.

7 Kerberos V5

Cryptosystem. Symmetric keys and trusted servers.

Description. This is a protocol for the distribution of a symmetric key (in a *ticket*), for communication between a client and a server, with authentication. It is based on based on the Needham Schroeder symmetric key protocol (Section 6) and uses timestamps and nonces to correct the flaw of Denning Sacco. The principals are:

C : client

S : a server (C wants to communicate with S)

U : user on behalf of which A and S communicate

G : ticket granting server

A : key distribution center (trusted server)

Reference: [NT94]

Protocol 7 KerberosV5/kerberosV5.eva

```
KerberosV5
 A, G, C, S, U :                      principal
 N1, N2 :                             number
 L1, L2 :                             number
 T1start, T1expire, T2start, T2expire : time
 Kcg, Kcs, Kag, Ku :                  key
 C knows                              C, G, A, S, U, Kcg, Kcs, Ku
 A knows                              A, G, Kag, Ku
 S knows                              S, C, Kcs
 G knows                              G, A, C, Kcg, Kag
 {
  1. C -> A : U, G, L1, N1
  2. A -> C : U, { U, C, G, Kcg, T1start, T1expire }_Kag,
                 { G, Kcg, T1start, T1expire }_Ku
  3. C -> G : S, L2, N2, { U, C, G, Kcg, T1start, T1expire }_Kag,
                         { C, T1start, T1expire }_Kcg
  4. G -> C : U, { U, C, S, Kcs, T2start, T2expire }_Kgs,
                 { S, Kcs, T2start, T2expire, N2 }_Kcg
  5. C -> S : { U, C, S, Kcs, T2start, T2expire }_Kgs,
              { C, T2start, T2expire }_Kcs
  6. S -> C : { T2start, T2expire }_Kcs
 }
claim Secret(Kcs)
      Agreement(C,A,Kcs,Kcs)
      Agreement(C,A,T2start,T2start)
      Agreement(C,A,T2expire,T2expire)
```

Analyses.

[NT94]

[SMB90] modelization with Abstract State Machines (stepwise refinements), and (manual) proof of correctness.

8 Woo and Lam Π^f

Cryptosystem. Symmetric keys and trusted server.

Description. One way authentication protocol with trusted server and symmetric keys.

Reference: [WL94] The definition of Woo and Lam of the correction (page 2 of [WL94]) of this

Protocol 8 WooLam_pif/WooLam_pif.eva

```
WooLam_Pif
 A, B, S :  principal
 Nb :       number
 Kas, Kbs : key
 A knows    A, Kas
 B knows    B, Kbs
 S knows    S, Kas, Kbs
 {
  1. A -> B :  A
  2. B -> A :  Nb
  3. A -> B :  { A,B,Nb }_Kas
  4. B -> S :  { A, B, Nb, { A, B, Nb }_Kas }_Kbs
  5. S -> B :  { A, B, Nb }_Kbs
 }
 claim Aliveness(A, B)
```

protocol corresponds exactly to the property `Aliveness` in EVA:

> "whenever a responder finishes the execution of the protocol, this initiator of the protocol execution is in fact the principal claimed in step 1."

Attack. No known attack.

Analyses. [WL94].

9 Woo and Lam Π

Cryptosystem. Symmetric keys and trusted server.

Description. One way authentication protocol, which is a flawed simplification of the above Π^f protocol.

Reference: [WL94]

Protocol 9 WooLam_pi/WooLam_pi.eva

```
WooLam_Pi
 A, B, S :  principal
 Nb :       number
 Kas, Kbs : key
 A knows    A, Kas
 B knows    B, Kbs
 S knows    S, Kas, Kbs
 {
  1. A -> B : A
  2. B -> A : Nb
  3. A -> B : { Nb }_Kas
  4. B -> S : { A, { Nb }_Kas }_Kbs
  5. S -> B : { Nb }_Kbs
 }
 claim Aliveness(A, B)
```

Attack. ref. [WL94], [CJ97]. Parallel session attack, and type flaw wich makes S consider $I, \{N_b\}_{K_{is}}$ as a nonce N'_b.

$$\begin{array}{lrcll}
a.1 & I(A) & \rightarrow & B: & A \\
a.2 & B & \rightarrow & I(A): & N_b \\
a.3 & I(A) & \rightarrow & B: & G \\
a.4 & B & \rightarrow & I(S): & \{A, G\}_{K_{bs}} \\
b.1 & B & \rightarrow & I(R): & B \\
b.2 & I(R) & \rightarrow & B: & I, \{N_b\}_{K_{is}} \\
b.3 & B & \rightarrow & I(R): & \{I, \{N_b\}_{K_{is}}\}_{K_{bs}} \\
b.4 & I(B) & \rightarrow & S: & \{I, \{N_b\}_{K_{is}}\}_{K_{bs}} \\
b.5 & S & \rightarrow & I(B): & \{N_b\}_{K_{bs}} \\
a.5 & I(S) & \rightarrow & B: & \{N_b\}_{K_{bs}}
\end{array}$$

Attack 4 `WooLam_pi/WooLam_pi-WooLam.eva`

```
WooLam_Pi_WooLam
 A, B, S :  principal
 Nb :       number
 Kas, Kbs : key
 A knows    A, Kas
 B knows    B, Kbs
 S knows    S, Kas, Kbs
 {
  1. A -> B : A
  2. B -> A : Nb
  3. A -> B : { Nb }_Kas
  4. B -> S : { A, { Nb }_Kas }_Kbs
  5. S -> B : { Nb }_Kbs
 }
session A = a, B = b, S = s
session A = r, B = b, S = s
claim Aliveness(A, B)
      Aliveness(S, B)
```

10 Woo and Lam Mutual Authentication protocol

Cryptosystem. Symmetric keys and trusted server.

Description. Mutual authentication with trusted server and symmetric keys.

Reference: [WL94]

Protocol 10 WooLam_mutual/WooLam_mutualAuthentication.eva

```
WooLam_mutualAuthentification
 P, Q, S :       principal
 Kps, Kqs, Kpq : key
 N1, N2 :        number
 P knows         P, Kps
 Q knows         Q, Kqs
 S knows         S, Kps, Kqs
 {
  1. P -> Q : P, N1
  2. Q -> P : Q, N2
  3. P -> Q : { P, Q, N1, N2 }_Kps
  4. Q -> S : { P, Q, N1, N2 }_Kps, { P, Q, N1, N2 }_Kqs
  5. S -> Q : { Q, N1, N2, Kpq }_Kps, { P, N1, N2, Kpq }_Kqs
  6. Q -> P : { Q, N1, N2, Kpq }_Kps, { N1, N2 }_Kpq
  7. P -> Q : { N2 }_Kpq
 }
 claim Aliveness(A, B)
```

Attack 1. Parallel session replay attack, ref. [CJ97].

$$
\begin{array}{lrcll}
a.1 & P & \rightarrow & I: & P, I \\
b.1 & I & \rightarrow & P: & I, N_1 \\
b.2 & P & \rightarrow & I: & P, N_2 \\
a.2 & I & \rightarrow & P: & I, N_2 \\
a.3 & P & \rightarrow & I: & \{P, I, N_1, N_2\}_{K_{ps}} \\
a.4 & I & \rightarrow & S: & \{P, I, N_1, N_2\}_{K_{ps}}, \{P, I, N_1, N_2\}_{K_{is}} \\
a.5 & S & \rightarrow & I: & \{I, N_1, N_2, K_{pi}\}_{K_{ps}}, \{P, N_1, N_2, K_{pi}\}_{K_{is}} \\
a.6 & I & \rightarrow & P: & \{I, N_1, N_2, K_{pi}\}_{K_{ps}}, \{N_1, N_2\}_{K_{pi}} \\
a.7 & P & \rightarrow & I: & \{N_2\}_{K_{pi}} \\
3.b & I & \rightarrow & P: & \{I, P, N_1, N_2\}_{K_{is}} \\
b.4 & P & \rightarrow & I(S): & \{I, P, N_1, N_2\}_{K_{is}}, \{I, P, N_1, N_2\}_{K_{ps}} \\
b.5 & I(S) & \rightarrow & P: & \{I, N_1, N_2, K_{pi}\}_{K_{is}}, \{I, N_1, N_2, K_{pi}\}_{K_{ps}} \\
b.6 & P & \rightarrow & I: & \{P, N_1, N_2, K_{pi}\}_{K_{is}}, \{N_1, N_2\}_{K_{pi}} \\
b.7 & I & \rightarrow & P: & \{N_2\}_{K_{pi}}
\end{array}
$$

Attack 5 `WooLam_mutual/WooLam_mutualAuthentication-CJ.eva`

```
WooLam_mutualAuthentification_CJ
 P, Q, S :        principal
 Kps, Kqs, Kpq : key
 N1, N2 :         number
 P knows          P, Kps
 Q knows          Q, Kqs
 S knows          S, Kps, Kqs
 {
  1. P -> Q : P, N1
  2. Q -> P : Q, N2
  3. P -> Q : { P, Q, N1, N2 }_Kps
  4. Q -> S : { P, Q, N1, N2 }_Kps, { P, Q, N1, N2 }_Kqs
  5. S -> Q : { Q, N1, N2, Kpq }_Kps, { P, N1, N2, Kpq }_Kqs
  6. Q -> P : { Q, N1, N2, Kpq }_Kps, { N1, N2 }_Kpq
  7. P -> Q : { N2 }_Kpq
 }
session A = I, B = B, S = I
 claim Aliveness(A, B)
```

Attack 2. [Low96]

$$\begin{array}{llcll}
a.1 & I(P) & \rightarrow & Q: & P, Q \\
a.2 & Q & \rightarrow & I(P): & Q, N_2 \\
a.3 & I(P) & \rightarrow & Q: & bit-string \\
a.4 & Q & \rightarrow & I(S): & bit-string, \{P, Q, Q, N_2\}_{K_{ps}} \\
b.1 & I(P) & \rightarrow & Q: & P, N_2 \\
b.2 & Q & \rightarrow & I(P): & Q, N_3 \\
b.3 & I(P) & \rightarrow & Q: & bit-string' \\
b.4 & Q & \rightarrow & I(S): & bit-string', \{P, Q, N_2, N_3\}_{K_{ps}} \\
a.5 & I(S) & \rightarrow & Q: & bit-string'', \{P, Q, N_2, N_3\}_{K_{ps}} \\
a.6 & Q & \rightarrow & I(P): & bit-string'', \{Q, N_2\}_{N_3} \\
a.7 & I(P) & \rightarrow & Q: & \{N_2\}_{N_3}
\end{array}$$

Attack 3. Parallel session replay attack in ref. [CJ]. (not available).

Attack 6 `WooLam_mutual/WooLam_mutualAuthentication-Lowe.eva`

```
WooLam_mutualAuthentification_Lowe
 P, Q, S :        principal
 Kps, Kqs, Kpq : key
 N1, N2 :         number
 P knows          P, Kps
 Q knows          Q, Kqs
 S knows          S, Kps, Kqs
 {
  1. P -> Q : P, N1
  2. Q -> P : Q, N2
  3. P -> Q : { P, Q, N1, N2 }_Kps
  4. Q -> S : { P, Q, N1, N2 }_Kps, { P, Q, N1, N2 }_Kqs
  5. S -> Q : { Q, N1, N2, Kpq }_Kps, { P, N1, N2, Kpq }_Kqs
  6. Q -> P : { Q, N1, N2, Kpq }_Kps, { N1, N2 }_Kpq
  7. P -> Q : { N2 }_Kpq
 }
 session P = I, Q = q, S = s
 session P = p, Q = I, S = s
 claim Agreement(P, Q, Kpq, Kpq)
```

11 Kao Chow Repeated Authentication Protocol (1)

Cryptosystem. Symmetric keys and trusted server.

Description. An authentication protocol which is supposed to be less vulnerable to attacks that Neuman Stubblebine's protocol.

Reference: [KC95]

Protocol 11 KaoChow_1/KaoChow_1.eva

```
Kao_Chow_1
 A, B, S :       principal
 Na, Nb :        number
 Kab, Kbs, Kas : key
 A knows         A, B, Kas
 B knows         B, Kbs
 S knows         S, Kas, Kbs
 {
  1. A -> S : A, B, Na
  2. S -> B : { A, B, Na, Kab }_Kas, { A, B, Na, Kab }_Kbs
  3. B -> A : { A, B, Na, Kab }_Kas, { Na }_Kab, Nb
  4. A -> B : { Nb }_Kab
 }
 claim Secret(Kab)
       Agreement(A,B,Kab,Kab)
       Agreement(A,B,Nb,Nb)
```

Attack. Possible when a session key is compromised (as in Denning Sacco attack on Nedham Schroeder)

12 Kao Chow Repeated Authentication Protocol (2)

Description. New version of the protocol of Section 11 to fix the compromised key problem, with an additional session key for handshake.

Protocol 12 KaoChow_2/KaoChow_2.eva

```
Kao_Chow_2
 A, B, S :           principal
 Na, Nb :            number
 Kab, Kbs, Kas, Kt : key
 A knows             A, B, Kas
 B knows             B, Kbs
 S knows             S, Kas, Kbs
 {
  1. A -> S : A, B, Na
  2. S -> B : { A, B, Na, Kab, Kt }_Kas, { A, B, Na, Kab, Kt }_Kbs
  3. B -> A : { A, B, Na, Kab }_Kas, { Na, Kab }_Kt, Nb
  4. A -> B : { Na, Kab }_Kt
 }
 claim Secret(Kab)
       Secret(Kt)
       Agreement(A,B,Kab,Kab)
       Agreement(A,B,Nb,Nb)
```

Reference: [KC95].

13 Gong Mutual Authentication Protocol

Cryptosystem. One-way functions, xor, trusted server (no keys).

Description. Mutual authentication of two principals `A` and `B` with the help of a server `S`. The principals `A` and `B` share respectively the secrets `Pa` and `Pb` with the `S`. A secret symmetric key `k` is also exchanged during the authentication.

This protocol does not use encryption but two (publicly known) one way functions `f` and `g`.

$$\begin{array}{lllll} \texttt{f}: & \texttt{number} \times \texttt{number} \times \texttt{number} \times \texttt{number} & \to & \texttt{number} \times \texttt{number} \times \texttt{number} \\ \texttt{g}: & \texttt{number} \times \texttt{number} \times \texttt{number} \times \texttt{number} & \to & \texttt{number} \end{array}$$

The values computed by `f` are indeed triples built with `<_,_>`. The third message of the protocol is in the original version [Gon89]:

```
3. S -> B : Ns, xor(f(Ns, Nb, A, Pb), <k,ha,hb>), g(k,ha,hb,Pb)
```

where `<k,ha,hb>` = `f(Ns, Na, B, Pa)`. The recipient `B` can compute `f(Ns, Nb, A, Pb)` to extract the secret values `k`, `ha` and `hb` from the message `3` (`g(k,ha,hb,Pb)` is a checksum). This requires the property of distributivity of `xor` on `<,>`, that are expressed in our syntax as an `axiom` (see [JLM01]). However, such an axiom would be ignored by the semantics [GL01a]. We preferred to use in the protocol specification three subfunctions `f1`,`f2`,`f3` computing the respective components of `f`.

Note that one axiom for `xor` in the specification is necessary though.

Reference: [Gon89]

Attack. No flaw known, according to [CJ97].

Protocol 13 `Gong/Gong.eva`

```
Gong
 A, B, S :                            principal
 Na, Nb, Ns, k :                      number
 Pa, Pb :                             number
 f1(number,number,number,number) :   number hash
 f2(number,number,number,number) :   number hash
 f3(number,number,number,number) :   number hash
 g(number, number, number, number) : number hash
 xor(number,number) :                number hash
 k, ha, hb :                          number
 A knows A, B, Pa
 B knows B, A, Pb
 S knows S, Pa, Pb
 alias k =  f1(Ns,Na,B,Pa)
 alias ha = f2(Ns,Na,B,Pa)
 alias hb = f3(Ns,Na,B,Pa)
 axiom xor(xor(x,y),x) = y

 {
  1. A -> B : A, B, Na
  2. B -> S : A, B, Na, Nb
  3. S -> B : Ns, xor(f1(Ns, Nb, A, Pb), k),
                  xor(f2(Ns, Nb, A, Pb), ha),
                  xor(f2(Ns, Nb, A, Pb), hb),
                  g(k, ha, hb, Pb)
  4. B -> A : Ns, hb
  5. A -> B : ha
 }
claim Secret(k)
      Agreement(A,B,k,k)
```

14 TMN

Cryptosystem. Symmetric keys and public keys and trusted server (only the public key of the server is used).

Description. Protocol for the exchange of a symmetric session key `Kb` between principals `A` and `B` via a trusted server `S`. This protocol uses both symmetric key and public key encryption:

- a symmetric session key `Ka` is created by `A` and transmitted to the server `S` (messages `1`) and it is used to encrypt the new session key `Kb` is the last message.
- communications between principals `A` and `B` and the server `S` use a public key cryptosystem. Actually, only the public key `PK(S)` of the trusted server `S` is used.

The last message can be modelized in two ways, either with an `xor` operator or with symmetric key encryption. We have chosen the second option.

Reference. [TMN89], see also [LR97].

Protocol 14 `TMN/TMN.eva`

```
TMN
 A, B, S : principal
 Ka, Kb  : key
 keypair PK, SK (key)
 A knows B,S,Ks, PK(S)
 B knows S,PK(S)
 S knows PK(S),SK(S)
 {
  1.  A -> S : B,{ Ka }_PK(S)
  2.  S -> B : A
  3.  B -> S : A,{ Kb }_PK(S)
  4.  S -> A : B,{ Kb }_Ka
 }
 claim Secret(Kb)
       Agreement(A, B, Kb, Kb)
```

Attack 1. [LR97]. Authentication and secrecy failure: the intruder I impersonates A, and uses a session key K_i of his choice to learn the established session key K_b in the last message.

$$
\begin{array}{lrcll}
1 & I(A) & \rightarrow & S: & B, \{K_i\}_{PK(S)} \\
2 & S & \rightarrow & B: & A \\
3 & B & \rightarrow & S: & A, \{K_b\}_{PK(S)} \\
4 & S & \rightarrow & I(A): & B, \{K_b\}_{K_i}
\end{array}
$$

Note that this is attack a very simple attack without parallel session or replay.

Attack 2. [LR97]. Authentication failure: the intruder I impersonates B and establishes a new session key K_i of his choice.

$$
\begin{array}{lrcll}
1. & A & \rightarrow & S: & B, \{K_a\}_{PK(S)} \\
2. & S & \rightarrow & I(B): & A \\
3. & I(B) & \rightarrow & S: & A, \{K_i\}_{PK(S)} \\
4. & S & \rightarrow & A: & B, \{K_i\}_{K_a}
\end{array}
$$

Attack 7 `TMN/TMN-Lowe1.eva`

```
TMN
 A, B, S : principal
 Ka, Kb  : key
 keypair PK, SK (key)
 A knows B,S,Ks, PK(S)
 B knows S,PK(S)
 S knows PK(S),SK(S)
 intruder knows B, S, PK(S)
 {
  1.  A -> S : B,{ Ka }_PK(S)
  2.  S -> B : A
  3.  B -> S : A,{ Kb }_PK(S)
  4.  S -> A : B,{ Kb }_Ka
 }
 claim Aliveness(A, B)
       Secret(Kb)
```

This attack demonstrates actually more than an authentication flaw, because the established session key is known to the intruder. With a fifth message:

```
5. A -> B : { X }_Kb
```

we would have a violation of the secrecy of `X`.

Attack 3. [LR97]. Parallel session and replay attack combining the above attacks 1 and 2. Secrecy and authentication failure: at the end of the second session, the intruder knows the established session key K_b.

$$
\begin{array}{llcll}
a.1 & I(A) & \rightarrow & S: & B, \{K_i\}_{PK(S)} \\
a.2 & S & \rightarrow & B: & A \\
a.3 & B & \rightarrow & S: & A, \{K_b\}_{PK(S)} \\
a.4 & S & \rightarrow & I(A): & B, \{K_b\}_{K_i} \\
b.1 & A & \rightarrow & S: & B, \{K_a\}_{PK(S)} \\
b.2 & S & \rightarrow & I(B): & A \\
b.3 & I(B) & \rightarrow & S: & A, \{K_b\}_{PK(S)} \\
b.4 & S & \rightarrow & I(A): & B, \{K_b\}_{K_a}
\end{array}
$$

Note that after this attack, A and B are going to communicate with the compromised session key K_b. This was not the case with attacks 1 and 2, because during these attacks, the authentication had been performed only with one honest principal.

15 CCITT X.509 (1)

Cryptosystem. Public keys and signature.

Description. A 1 message protocol from the recommendations of the CCITT for the CCITT.X.509 standard (1987), presented in [AN96]. The timestamp `Ta` and nonce `Na` are not used here. `Xa` and `Ya` are the data transmitted, the privacy of `Ya` is ensured by its encryption with the public key of `B` and the origin of `Xa` and `Ya` is ensured by the encryption with the private key of `A`.

Protocol 15 `CCITTX.509_1/CCITTX.509_1.eva`

```
CCITT509_1
 A, B :          principal
 PK(principal) : key
 SK(principal) : key
 Na :            number
 Ta :            timestamp
 Ya :            userdata
 Xa :            userdata
 A knows A, PK(A), SK(A), PK(B)
 B knows B, PK(B), SK(B), PK(A)
 {
  1. A -> B : A, { Ta, Na, B, Xa, { Ya }_PK(B) }_SK(A)
 }
 claim Agreement(A, B, Xa, Xa)
       Agreement(A, B, Ya, Ya)
       Agreement(A, B, A, A)
```

Attack. [AN96]. The protocol is supposed to ensure the recipient B of the message that the data X_a and Y_a originate from A. However, this is not the case as the following attack shows.

$$\begin{array}{llcll} a.1 & A & \rightarrow & I(B): & A, \{T_a, N_a, B, X_a, \{Y_a\}_{PK(B)}\}_{SK(A)} \\ b.1 & I & \rightarrow & B: & I, \{T_a, N_a, B, X_a, \{Y_a\}_{PK(B)}\}_{SK(I)} \end{array}$$

This attack illustrates the difference between the properties `Agreement` and `Aliveness`. Indeed, in the above scenario, `Aliveness(A,B)` is not negated, whereas `Agreement(A, B, A, A)` is.

Analyses. [BAN89].

Attack 8 `CCITTX.509_1/CCITTX.509_1-AbadiNeedham.eva`

```
CCITTX.509_1-AbadiNeedham
 A, B :          principal
 PK(principal) : key
 SK(principal) : key
 Na :            number
 Ta :            timestamp
 Ya :            userdata
 Xa :            userdata
 A knows A, PK(A), SK(A), PK(B)
 B knows B, PK(B), SK(B), PK(A)
 {
  1. A -> B : A, { Ta, Na, B, Xa, { Ya }_PK(B) }_SK(A)
 }
 claim Aliveness(A, B)
```

16 CCITT X.509 (1c)

Cryptosystem. Public keys and signature.

Description. The solution proposed in [AN96] to correct the above flaw in the CCITT X.509 one message protocol is to sign the secret data `Ya` before it is encrypted.

Protocol 16 `CCITTX.509_1c/CCITTX.509_1c.eva`

```
CCITT509_1c
 A, B :          principal
 PK(principal) : key
 SK(principal) : key
 Na :            number
 Ta :            timestamp
 Ya :            userdata
 Xa :            userdata
 A knows A, PK(A), SK(A), PK(B)
 B knows B, PK(B), SK(B), PK(A)
 {
  1. A -> B : A, { Ta, Na, B, Xa, { Ya, { h(Ya) }_SK(A) }_PK(B) }_SK(A)
 }
claim Agreement(A, B, Xa, Xa)
      Agreement(A, B, Ya, Ya)
      Agreement(A, B, A, A)
```

17 CCITT X.509 (3)

Cryptosystem. Public keys and signature.

Description. 3 messages protocol in the recommendations of the CCITT for the CCITT.X.509 standard (1987), presented and analysed in [BAN89].

Protocol 17 `CCITTX.509/CCITTX.509.eva`

```
CCITT509
 A, B :          principal
 PK(principal) : key
 SK(principal) : key
 Na, Nb :        number
 Ta, Tb :        timestamp
 Ya, Yb :        userdata
 Xa, Xb :        userdata
 A knows A, PK(A), SK(A), PK(B)
 B knows B, PK(B), SK(B), PK(A)
 {
  1. A -> B : A, { Ta, Na, B, Xa, { Ya }_PK(B) }_SK(A)
  2. B -> A : B, { Tb, Nb, A, Na, Xb, { Yb }_PK(A) }_SK(B)
  3. A -> B : A, { Nb }_SK(A)
 }
claim Agreement(A, B, Ya, Ya)
      Agreement(A, B, Na, Na)
      Agreement(A, B, Nb, Nb)
```

Attack. This parallel session attack presented in [BAN89] works if B does not check the timestamp T_a in the first message.

$$
\begin{array}{lrcll}
a.1 & A & \rightarrow & I(B): & A, \{T_a, N_a, B, X_a, \{Y_a\}_{PK(B)}\}_{SK(A)} \\
a.1 & I(A) & \rightarrow & B: & A, \{T_a, N_a, B, X_a, \{Y_a\}_{PK(B)}\}_{SK(A)} \\
a.2 & B & \rightarrow & I(A): & B, \{T_b, N_b, A, N_a, X_b, \{Y_b\}_{PK(A)}\}_{SK(B)} \\
b.1 & A & \rightarrow & I: & A, \{T'_a, N'_a, C, X'_a, \{Y'_a\}_{PK(I)}\}_{SK(A)} \\
b.2 & I & \rightarrow & A: & I, \{T_i, N_b, A, N'_a, X_i, \{Y_i\}_{PK(A)}\}_{SK(I)} \\
b.3 & A & \rightarrow & I: & A, \{N_b\}_{SK(A)} \\
b.3 & I(A) & \rightarrow & B: & A, \{N_b\}_{SK(A)}
\end{array}
$$

Another attack can be found in [lM90].

Analyses. [BAN89].

Attack 9 `CCITTX.509/CCITTX.509-BAN.eva`

```
CCITT509
 A, B :          principal
 PK(principal) : key
 SK(principal) : key
 Na, Nb :        number
 Ta, Tb :        timestamp
 Ya, Yb :        userdata
 Xa, Xb :        userdata
 A knows A, PK(A), SK(A), PK(B)
 B knows B, PK(B), SK(B), PK(A)
 {
  1. A -> B : A, { Ta, Na, B, Xa, { Ya }_PK(B) }_SK(A)
  2. B -> A : B, { Tb, Nb, A, Na, Xb, { Yb }_PK(A) }_SK(B)
  3. A -> B : A, { Nb }_SK(A)
 }
 session A = a, B = b
 session A = a, B = I
 claim Aliveness(A, B)
```

18 Needham Schroeder Public Key Protocol

Cryptosystem. Public keys and certificates (with Certificate Authority).

Description. Mutual authentication of two principals in a public key environment. The distribution of certificates for public keys is ensured by a trusted server S. One can also consider the version without the messages concerning S, i.e. when the two principals A and B are assumed to know the public keys of each other.

Reference: [NS78b]

Protocol 18 NeedhamSchroederPublicKey/NSPK.eva

```
NSPK
 A, B, S :       principal
 PK(principal) : key
 SK(principal) : key
 Na, Nb :        number
 A knows A, B, S, PK(A), PK(S)
 B knows B, S, PK(S)
 S knows S, A, B, PK
 {
  1. A -> S : A, B
  2. S -> A : { PK(B), B }_SK(S)
  3. A -> B : { Na, A }_PK(B)
  4. B -> S : B, A
  5. S -> B : { PK(A), A }_SK(S)
  6. B -> A : { Na, Nb }_PK(A)
  7. A -> B : { Nb }_PK(B)
 }
 claim Agreement(A, B, Na, Na)
       Agreement(A, B, Nb, Nb)
```

Attack. [Low95]. Parallel session attack (freshness flaw of N_b). Messages 1,2,4,5, concerning only the distribution of the public keys, are omitted in this description of the attack.

$$\begin{array}{lrcll} a.3 & A & \rightarrow & I: & \{N_a, A\}_{PK(I)} \\ b.3 & I(A) & \rightarrow & B: & \{N_a, A\}_{PK(B)} \\ b.6 & B & \rightarrow & I(A): & \{N_a, N_b\}_{PK(A)} \\ a.6 & I & \rightarrow & A: & \{N_a, N_b\}_{PK(A)} \\ a.7 & A & \rightarrow & I: & \{N_b\}_{PK(I)} \\ b.7 & I(A) & \rightarrow & B: & \{N_b\}_{PK(B)} \end{array}$$

Analyses. ref. [Low95].

Attack 10 NeedhamSchroederPublicKey/NSPK-Lowe.eva

```
NSPK_Lowe
 A, B, S :       principal
 PK(principal) : key
 SK(principal) : key
 Na, Nb :        number
 A knows A, B, S, PK(A), PK(S)
 B knows B, S, PK(S)
 S knows S, A, B, PK
 {
  1. A -> S : A, B
  2. S -> A : { PK(B), B }_SK(S)
  3. A -> B : { Na, A }_PK(B)
  4. B -> S : B, A
  5. S -> B : { PK(A), A }_SK(S)
  6. B -> A : { Na, Nb }_PK(A)
  7. A -> B : { Nb }_PK(B)
 }
 session A = a, B = b, S = s
 session A = a, B = I, S = s
 claim Agreement(A, B, Na, Na)
       Agreement(A, B, Nb, Nb)
```

19 Hwang and Chen's Modified SPLICE/AS

Cryptosystem. Public keys and certificates (with Certificate Authority).

Description. Mutual authentication between client C and server S, using the certificate authority AS to distribute session keys.

Protocol 19 SPLICE_AS/SPLICE_AS.eva

```
SPLICE_AS
 A, AS, C, S :   principal
 N1, N2, N3 :    number
 L :             lifetime
 T :             timestamp
 PK(principal) : key
 SK(principal) : key
 inc(number) :   number
 AS knows AS, PK(AS), SK(AS), PK(S)
 C knows C, PK(C), SK(C), S
 S knows S, PK(S)
 {
  1. C -> AS : C, S, N1
  2. AS -> C : AS, { AS, C, N1, S, Ks }_SK(AS)
  3. C -> S  : C, S, { C, T, L, { N2 }_PK(S) }_SK(C)
  4. S -> AS : S, C, N3
  5. AS -> S : AS, { AS, S, N3, Kc }_SK(AS)
  6. S -> C  : S, C, { S, inc(N2) }_PK(C)
 }
 claim Aliveness(C, S)
```

Attack 1. Replay attack, ref. [HC95]

$$
\begin{array}{llcll}
a.3 & C & \rightarrow & I(S): & C, S, \{C, T, L, \{N2\}_{PK(S)}\}_{SK(C)} \\
b.3 & I & \rightarrow & S: & I, S, \{I, T, L, \{N2\}_{PK(S)}\}_{SK(I)} \\
a.6 & S & \rightarrow & I: & S, I, \{S, inc(N2)\}_{PK(I)} \\
b.6 & I(S) & \rightarrow & C: & S, C, \{S, inc(N2)\}_{PK(C)}
\end{array}
$$

Attack 11 SPLICE_AS/SPLICE_AS-HwangChen.eva

```
SPLICE_AS
 A, AS, C, S :   principal
 N1, N2, N3 :    number
 L :             lifetime
 T :             timestamp
 PK(principal) : key
 SK(principal) : key
 inc(number) :   number
 AS knows AS, PK(AS), SK(AS), PK(S)
 C knows C, PK(C), SK(C), S
 S knows S, PK(S)
 {
  1. C -> AS : C, S, N1
  2. AS -> C : AS, { AS, C, N1, S, Ks }_SK(AS)
  3. C -> S  : C, S, { C, T, L, { N2 }_PK(S) }_SK(C)
  4. S -> AS : S, C, N3
  5. AS -> S : AS, { AS, S, N3, Kc }_SK(AS)
  6. S -> C  : S, C, { S, inc(N2) }_PK(C)
 }
 session C = c, S = s, AS = as
 session C = c, S = I, AS = as
 claim aliveness(C, S)
```

20 Diffie Helman

Cryptosystem. Diffie Helman key. The well known Diffie Helman key exchange algorithm. See the presentation of this specification in the EVA report [JLM01].

Reference: [DH76]

Protocol 20 `DiffieHelman/DH.eva`

```
Diffie_Hellman
 A, B :         principal
 P, G, Xa, Xb : number
 un() :         number
 kap (number, number, number) : number hash
 kas (number, number, number) : number hash
 A knows B, kap, kas, un
 B knows kap, kas
 axiom Kas(P,Kap(P,G,Y),X) = Kas(P,Kap(P,G,X),Y)
 {
  1. A -> B : P, G
  2. A -> B : kap(P, G, Xa)
  3. B -> A : kap(P, G, Xb)
  4. A -> B : {un()}_kas(P, kap (P, G, Xb), Xa)%kas(P, kap (P, G, Xa), Xb)
 }
 claim Agreement(A,B,kas(P, kap (P, G, Xb), Xa))
```

The functions `Kap` and `Kas` must satisfy `Kas(P,Kap(P,G,Y),X)` = `Kas(P,Kap(P,G,X),Y)`. They are implemented by $\texttt{Kap}(p,g,x) = g^x \ mod\ p$ for some prime p and $g < p$ and $\texttt{Kas}(p,x,y) = x^y \ mod\ p$. The key exchanged is `Kas(P,Kap(P,G,Xb),Xa)` = `Kas(P,Kap(P,G,Xa),Xb)`. The protocol guaranties the secrecy of this key but not authenticity of the participants.

Analyse. [Bla01].

21 SKEME

Cryptosystem. Public keys and pseudo-random functions.

Description. Protocol for exchange of a shared key and authentication.

Reference: [Kra96]

Note: This protocol makes use of a pseudo-random function `F` to compute a MAC `F(K,X)`, i.e. a hash code of `X` with the help of the key `K`. The function `F` is such that two MACs computed with different keys must be incomparable and `F` is not invertible even for someone who knows the key. EVA has no special syntactic construction for pseudo-random functions. In the specification below, we simply declare `F` as a one-way function, with the keyword `Hash`. The semantics are the same. The shared key is `H(Ka,Kb)`. The authentication is ensured by parts `EXCH` and `AUTH`.

Protocol 21 `SKEME/SKEME.eva`

```
SKEME
 A, B :       principal
 Ka, Kb :     key
 X, Y : number
 PK(principal) : key
 SK(principal) : key
 F(key, number) : number hash
 H(key,key) : key
 A knows A, B, PK(A), SK(A), F, H
 B knows B, PK(B), SK(B), F, H
 intruder knows F, H
 {
  SHARE1. A -> B : { Ka }_PK(B)
  SHARE2. B -> A : { Kb }_PK(A)
  EXCH1.  A -> B : X
  EXCH2.  B -> A : Y
  AUTH1.  A -> B : F(H(Ka,Kb), Y, X, A, B)
  AUTH2.  B -> A : F(H(Ka,Kb), X, Y, B, A)
 }
 claim Secret(H(Ka,Kb))
       Agreement(A,B,X,X)
       Agreement(A,B,Y,Y)
```

22 SKEME_PFS

Cryptosystem. Public keys and Diffie Helman keys and pseudo-random functions.

Description. A variant of the protocol in Section 21 to satisfy the property of *perfect forward secrecy*. It uses a Diffie-Helman scheme. The shared key is the Diffie Helman key computed from

Protocol 22 SKEME_PFS/SKEME_PFS.eva

```
SKEME_PFS
 A, B :           principal
 Ka, Kb :         key
 X, Y :           number
 PK(principal) :  key
 SK(principal) :  key
 F(key, number) : number hash
 H(key,key) :     key
 Kap(number) :    number hash
 A knows A, B, PK(A), SK(A), F, H, Kap
 B knows B, PK(B), SK(B), F, H, Kap
 {
  SHARE1. A -> B : { Ka }_PK(B)
  SHARE2. B -> A : { Kb }_PK(A)
  EXCH1.  A -> B : Kap(X)
  EXCH2.  B -> A : Kap(Y)
  AUTH1.  A -> B : F(H(Ka,Kb), Kap(Y), Kap(X), A, B)
  AUTH2.  B -> A : F(H(Ka,Kb), Kap(X), Kap(Y), B, A)
 }
 claim Secret(H(Ka,Kb))
       Agreement(A,B,Kap(X),Kap(X))
       Agreement(A,B,Kap(Y),Kap(Y))
```

`Kap(X)` $(= g^{\mathtt{X}}\, mod\, p)$ and `Kap(Y)` $(= g^{\mathtt{X}}\, mod\, p)$; i.e. this key is$(= g^{\mathtt{XY}}\, mod\, p)$.

Analyse. [Bla01].

23 Open Platform Mutual Authentication Protocol

Cryptosystem. Symmetric keys (3-DES).

description. This mutual authentication protocol from Visa is part of the initialization of a secure channel between a smart card and a terminal in the Open Platform environment

Reference: Secure Channel Protocol 01 in OP specification version 2.1, [VOP01].

This protocol allows a smartcard and a terminal to agree on a session key for encryption and another session key to compute MACs, and ensures that they share the same keys. After performing this protocol, a connexion can be established with any of the three following security measure (the corresponding *security level* is established during the mutual authentication protocol, by the variable `L`)

0 none

1 a MAC (computed with the MAC session key) is sent along each message for ensuring the integrity and origin of data

3 MACs are sent like in the previous case and moreover every message is transmitted encrypted with the encryption session key, ensuring data confidentiality.

The verification of data integrity is crucial e.g. when loading an applet into a smart card. Data confidentiality is required when transmitting some keys to the card.

Description of variables:

- `H` is the terminal (off-card entity)
- `C` is the card (on-card system)
- `L` is a security level for communications unsing the secure channel (among 0, 1, 3)
- `{_}_` is 3DES in ECB mode
- `f(Nh,Nc)` = `rightHalf(Nc),leftHalf(Nh),leftHalf(Nc),rightHalf(Nh)`; Note that this function is reversible (not one-way).
- `leftHalf(Nc)` = the 4 bytes left half of the 8 byte nonce `Nc`
- `rightHalf(Nc)` = the 4 bytes right half of the 8 byte nonce `Nc`
- `SENC` = static Secure Channel Encryption Key, a secret symmetric key (double length DES) shared by `H` and `C`
- `SMAC` = static Secure Channel Message Authentication Code Key, a secret symmetric key (double length DES) shared by `H` and `C`.

For the sake of conciseness, we have omitted some auxiliary informations in messages, and paddings in cryptograms here. Note that the `MAC` prevents the intruder from replaying the last message with a security level changed to 0.

Protocol 23 `OpenPlatform/OP.eva`

```
OpenPlatform
 H, C :             principal
 Nh, Nc :           number
 L:                 number
 f(number, number) : number
 Senc, Smac :       key
 H knows StMK, StEK
 C knows StMK, StEK
 alias MAC =  { L, { Nc, Nh }_{ f(Nc, Nh) }_SENC }_{ f(Nc, Nh) }_Smac
 {
  INITIALIZE_UPDATE.
   H -> C :  Nh
  INITIALIZE_UPDATE_response.
   C -> H :  Nc, { Nh, Nc }_{ f(Nc, Nh) }_Senc
  EXTERNAL_AUTHENTIFICATE.
   H -> C : L, { Nc, Nh }_{ f(Nc, Nh) }_SENC, MAC
 }
 claim Aliveness(H, C)
```

24 Horn and Preneel authentication and payment protocol

Cryptosystem. Diffie Helman scheme, one-way functions and (initialisation of) Lamport's password scheme (one time password).

Description. Protocol for the mutual authentication and exchange of a symmetric key `K` in mobile systems, between a user `U` and an UMTS *value added service provider* `V`. This protocol embeds also the initialization of a stack of micro-payment tokens.

The user `U` possesses a pair of public/private keys for signatures, together with a certificate `certU` from an authority `CAU`. The provider `U` has long term secret and public key agreement (Diffie Helman scheme), respectively `Y` and `e(G, Y)`, where `G` is the generator of a finite group in which the discrete logarithm problem is NP-hard. He has also a certificate `certV` for this agreement, from an authority `CAV`.

The choice of cryptographic functions is let free in the paper, as well as the form of the certificates `certU` and `certV`, though in our specification we had to choose a particular form for these certificates (otherwise, the translation process into abstract syntax [GL01a] would fail).

The variables `CD` (charging data) and `TV` (time stamp), as well as `IV` (initialization vector), `f` (function), `T` and `AO` are not relevant in the protocol specified, but are concerned with the payment scheme taking place after a successful execution of this protocol.We write them in the protocol specification only for the sake of conformity.

Reference:. [HP00]

Protocol 24 `HornPreneel/HornPreneel.eva`

```
HornPreneel
  U, V, CAU, CAV : principal
  S, R, Y, G : number
  CD : data
  TV : timestamp
  T, A0, IV : number
  kap(number,number) : number         hash
  h1(number) : number                 hash
  h2(number) : number                 hash
  h3(number) : number                 hash
  f(number, number, number) : number  hash
  keypair PK, SK (principal)
  certU, certV, K, At : number
  alias certU = { CAU, U, PK(U) }_SK(CAU)
  alias certV = { CAV, V, kap(G, Y) }_SK(CAV)
  alias K = h1((kap(kap(G, S), V), R))
  alias At = f(T, IV, A0)
  U knows V, PK(U), SK(U), certU, CAU, CAV, PK(CAV), h1,h2,h3, G, kap, T, f
  V knows Y, certV, CAV, CAU, PK(CAU), h1,h2,h3, G, kap, T, f
  axiom kap(kap(G,S),V) =  kap(kap(G,V),S)
  {
   1. U -> V : kap(G, S), CAV
   2. V -> U : R, h2((K, R, V)), CD, TV, certV
   3. U -> V : { { h3((kap(G,S),kap(G,Y), R, V, CD, TV, At, IV)) }_SK(U),
                 certU, At, IV }_K
  }
  claim Aliveness(U,V)
        Agreement(U,V,K,K)
        Agreement(U,V,At,At)
        Agreement(U,V,IV,IV)
```

25 MSR

Cryptosystem. Modulo square root public keys (fast), certificates, symmetric keys.

Description. This hybrid protocol due to Beller, Chang and Yacobi uses a public key cryptosystem to establish a shared symmetric key between a mobile `M` and a base station `B`. It relies on an efficient public key encryption from Rabin (Modulo Square Root), denoted `msr` in our specification (though this information is not relevant for the verification of the protocol).

References. [BCY93], see also [BM98].

Protocol 25 `MSR/MSR.eva`

```
MSR
 msr :      algo
 B, M, CA : principal
 K :        key
 certM :    number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 {
  1. B -> M : B, PK(B)
  2. M -> B : { K }_PK(B)^msr, { M, certM }_K
 }
 claim Secret(K)
       Agreement(M,B,K,K)
```

Attack 1. [Car94]. There is no certificate for `B`'s public key `PK(B)` in the first message, hence it is possible for an intruder to impersonate `B`.

$$\begin{array}{lrcll} 1 & I(B) & \rightarrow & M: & B, PK(I) \\ 2 & M & \rightarrow & I(B): & \{K\}_{PK(I)}, \{M, CertM\}_K \end{array}$$

Attack 2. [Car94]. Replay (freshness) attack. The intruder may also replay an old second message from a previous session of the protocol, impersonating B. In this attack, K is not a fresh key but the key exchanged in the previous session.

$$\begin{array}{lrcll} 1 & B & \rightarrow & M: & B, PK(B) \\ 2 & I(M) & \rightarrow & B: & \{K\}_{PK(B)}, \{M, CertM\}_K \end{array}$$

If an old key K, has been compromised (in another session), then the intruder is also able (masquerading M) to forge and distribute a new shared key K' (because he knows $M, CertM$).

$$\begin{array}{lrcll} 1 & B & \rightarrow & M: & B, PK(B) \\ 2 & I(M) & \rightarrow & B: & \{K'\}_{PK(B)}, \{M, CertM\}_{K'} \end{array}$$

26 IMSR

Cryptosystem. Modulo square root public keys (fast), certificates, symmetric keys.

Description. An improved version of the above protocol (§ 25), to fix the first flaw presented in § 25 by the addition of a certificate for the base station B in the first message.

References. [BCY93], see also [BM98].

Protocol 26 IMSR/IMSR.eva

```
IMSR
 msr :          algo
 B, M, CA :     principal
 K :            key
 certM, certB : number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. B -> M : B, PK(B), certB
  2. M -> B : { K }_PK(B)^msr, { M, certM }_K
 }
 claim Secret(K)
       Agreement(M,B,K,K)
```

Attack 2. [Car94]. This patch of the certificate does not fix the above replay attack (see § 25).

27 IIMSR

Cryptosystem. Modulo square root public keys (fast), certificates, symmetric keys.

Description. An improved version from Carlsen [Car94] of the improved version of the MSR protocol (§25, § 26), to fix both flaws presented in § 25.

References. [Car94], see also [BM98].

Protocol 27 `IIMSR/IIMSR.eva`

```
IIMSR
 msr :          algo
 B, M, CA :     principal
 K :            key
 Nb :           number
 certM, certB : number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. B -> M : B, Nb, PK(B), certB
  2. M -> B : { K }_PK(B)^msr, { Nb, M, certM }_K
 }
 claim Secret(K)
       Agreement(M,B,K,K)
       Agreement(M,B,Nb,Nb)
```

Attack 2. [Car94]. Actually, this improvement from [Car94] does not overcome the second kind of replay attack, presented in § 25 (when the intruder has compromised a key K and hence knows the certificate $certM$).

28 I3MSR

Cryptosystem. Modulo square root public keys (fast), certificates, symmetric keys.

Description. A last improvement on MSR (§ 25) to overcome the replay attack when a session key has been compromised.

References. [BM98].

Protocol 28 `I3MSR/I3MSR.eva`

```
I3MSR
 msr :          algo
 B, M, CA :     principal
 K :            key
 Nb :           number
 certM, certB : number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. B -> M : B, Nb, PK(B), certB
  2. M -> B : { K,Nb,certM }_PK(B)^msr, { M }_K
 }
 claim Secret(K)
       Agreement(M,B,K,K)
       Agreement(M,B,Nb,Nb)
```

29 Beller Yacobi

Cryptosystem. Public keys (Al Gamal), certificates, symmetric keys.

Description. It is another variation on the IMSR protocol, where the mobile `B` has an assymetric key pair for signature.

References. [BY93] see also [BM98].

Protocol 29 `BellerYacobi/BellerYacobi.eva`

```
BellerJacobi
 algamal :      algo
 B, M, CA :     principal
 K :            key
 Nb :           number
 certM, certB : number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. B -> M : B, PK(B), certB
  2. M -> B : { K }_PK(B)^algamal
  3. B -> M : { Nb }_K
  4. M -> B : { M,PK(M),certM, { Nb }_SK(M) }_K
 }
 claim Secret(K)
       Agreement(M,B,K,K)
       Agreement(M,B,Nb,Nb)
```

Attack. There is a parallel session attack described in [BM98]. The intruder I must be a legitimate user known by the base station B.

$$\begin{array}{lrcll}
a.1. & B & \rightarrow & I(M): & B, PK(B), certB \\
a.2. & I(M) & \rightarrow & B: & \{K\}_{PK(B)} \\
a.3. & B & \rightarrow & I(M): & \{N_b\}_K \\
b.1. & I & \rightarrow & M: & I, PK(I), certI \\
b.2. & M & \rightarrow & I: & \{K'\}_{PK(I)} \\
b.3. & I & \rightarrow & M: \{N_b\}_{K'} & \\
b.4. & M & \rightarrow & I: & \{M, PK(M), CertM, \{N_b\}_{SK(M)}\}_{K'} \\
a.4. & I(M) & \rightarrow & B: & \{M, PK(M), CertM, \{N_b\}_{SK(M)}\}_K
\end{array}$$

Attack 12 BellerYacobi/BellerYacobi-BoydMathuria.eva

```
BellerJacobi
 algamal :      algo
 B, M, CA :     principal
 K :            key
 Nb :           number
 certM, certB : number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 intruder knows B, M, PK(I), SK(I), certI
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 alias certI = { CA, I, PK(I) }_SK(CA)
 {
  1. B -> M : B, PK(B), certB
  2. M -> B : { K }_PK(B)^algamal
  3. B -> M : { Nb }_K
  4. M -> B : { M,PK(M),certM, { Nb }_SK(M) }_K
 }
 session B=b, M=m
 session B=I, M=m
 claim Agreement(M,B,K,K)
       Agreement(M,B,Nb,Nb)
```

30 Beller Yacobi amended

Cryptosystem. Public keys (Al Gamal), certificates, symmetric keys.

Description. This revision of the Beller Yacobi protocol (§ 29) prevents the above parallel session attack, by forcing `M` to sign the session key `K` when it sends it to `B` in the second message.

References. [BM98].

Protocol 30 `BellerYacobi_c/BellerYacobi_c.eva`

```
BellerJacobi_c
 algamal :      algo
 B, M, CA :     principal
 K :            key
 Nb :           number
 certM, certB : number
 keypair PK, SK (key)
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. B -> M : B, PK(B), certB, Nb
  2. M -> B : { K }_PK(B)^algamal, { M, PK(M), certM }_K,
             { h(B,M,Nb,K) }_SK(M)
  3. B -> M : { Nb }_K
 }
 claim Secret(K)
       Agreement(M,B,K,K)
       Agreement(M,B,Nb,Nb)
```

31 Aziz Diffie

Cryptosystem. Public keys, certificates, one-way functions.

Description. In this public key protocol a mobile M and a base station B exchange the components Xb and Xm which will allow them to build a shared symmetric key (session key) Xm $\oplus$ Xb.

The participants are supposed to have certificates for their respective public keys. In our specification, we assume that both certificates of B and M are signed by the same certificate authority CA.

In the first message of the protocol, the mobile M proposes a list of symmetric-key algorithms, among which the base B chooses one algorithm for future sessions with the newly computed shared key. In our specification, we assume that this list is restricted to one protocol A and hence that B chooses A (hence, the verifications of the protocol based on this specification will ignore the choice of protocol).

Note the heavy use of public key cryptography made by this protocol.

Reference. [AD94], see also [BM98].

Protocol 31 AzizDiffie/AzizDiffie.eva

```
AzizDiffie
 A :      algo
 B, M, CA :     principal
 Xb, Xm :       key
 Nm :           number
 certM, certB : number
 keypair PK, SK (key)
 h(number) : number
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. M -> B : certM, Nm, A
  2. B -> M : certB, { Xb }_PK(M), A,
              { h(({ Xb }_PK(M), A, Nm, A)) }_SK(B)
  3. M -> B : { Xm }_PK(B), { h(( {Xm}_PK(B), {Xb}_PK(M) )) }_SK(M)
 }
 claim Secret(Xm)
       Secret(Xb)
       Agreement(M,B,Xm,Xm)
       Agreement(M,B,Xb,Xb)
```

Attack 1. C. Meadows [Mea95].

$$
\begin{array}{lrcll}
a.1 & M & \rightarrow & I(B): & cert_M, N_m, A \\
b.1 & I & \rightarrow & B: & cert_I, N_m, A \\
b.2 & B & \rightarrow & I(M): & cert_B, \{X_b\}_{PK(I)}, A \\
 & & & & \{h(\{X_b\}_{PK(I)}, A, N_m, A)\}_{SK(B)} \\
b.2 & I(B) & \rightarrow & M: & cert_B, \{X_b\}_{PK(I)}, A \\
 & & & & \{h(\{X_b\}_{PK(I)}, A, N_m, A)\}_{SK(B)}
\end{array}
$$

After this run, M computes $\{\{X_b\}_{PK(I)}\}_{SK(M)}$ to obtain what he thinks to be X_b. Hence, M and B wont share the same symmetric key though they authentified themself. This may result in a denial of service during the opening of a secure session.

Attack 13 `AzizDiffie/AzizDiffie-Meadows.eva`

```
AzizDiffie
 A :       algo
 B, M, CA :      principal
 Xb, Xm :        key
 Nm :            number
 certM, certB : number
 keypair PK, SK (key)
 h(number) : number
 B knows M, PK(B), SK(B), certB, CA, PK(CA)
 M knows B, PK(M), SK(M), certM, CA, PK(CA)
 intruder knows B, M
 alias certM = { CA, M, PK(M) }_SK(CA)
 alias certB = { CA, B, PK(B) }_SK(CA)
 {
  1. M -> B : certM, Nm, A
  2. B -> M : certB, { Xb }_PK(M), A,
              { h(({ Xb }_PK(M), A, Nm, A)) }_SK(B)
  3. M -> B : { Xm }_PK(B), { h(( {Xm}_PK(B), {Xb}_PK(M) )) }_SK(M)
 }
 session M=m, B=b
 session M=I, B=b
 claim Agreement(M,B,Xb,Xb)
```

Attack 2. [BM98].

$$
\begin{array}{lrcll}
a.1. & I(M) & \rightarrow & B: & cert_M, N_i, A \\
a.2. & B & \rightarrow & I: & cert_B, \{X_b\}_{PK(M)}, A, \\
 & & & & \{h(\{X_b\}_{PK(M)}, A, N_i, A)\}_{SK(B)} \\
b.1. & M & \rightarrow & I: & cert_M, N_m, A \\
b.2. & I & \rightarrow & M: & cert_I, \{X_b\}_{PK(M)}, A, \\
 & & & & \{h(\{X_b\}_{PK(M)}, A, N_m, A)\}_{SK(I)} \\
b.3. & M & \rightarrow & I: & \{X_m\}_{PK(I)}, \{h(\{X_m\}_{PK(I)}, \{X_b\}_{PK(M)})\}_{SK(M)} \\
a.3 & I(M) & \rightarrow & B: & \{X_m\}_{PK(I)}, \{h(\{X_m\}_{PK(I)}, \{X_b\}_{PK(M)})\}_{SK(M)}
\end{array}
$$

Reference

References

[AD94] A. Aziz and W. Diffie. Privacy and authentication in wireless local area networks. *IEEE Personal Communications*, 1:25–31, 1994.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[BAN89] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. Technical Report 39, Digital Systems Research Center, february 1989.

[BCY93] M. Beller, L. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. *IEEE J. Selected Areas in Communications*, 11(6):821–829, 1993.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In IEEE, editor, *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, june 2001.

[BM98] C. Boyd and A. Mathuria. Key establishment protocols for secure mobile communications: A selective survey. In *Information Security and Privacy*, volume 1438 of *LNCS*, pages 344–355. Springer-Verlag, 1998.

[BY93] M. Beller and Y. Yacobi. Fully-fledged two-way public key authentication and key agreement for low cost terminals. *Electronic Letters*, 30:999–1001, 1993.

[Car94] U. Carlsen. Optimal privacy and authentication on a portable communications system. *Operating Systems Review*, 28(3):16–23, July 1994.

[CJ] John Clark and Jeremy Jacob. Freshness is not enough : Note on trusted nonce generation and malicious principals. attack on a mutual authentification protocol by Woo and Lam.

[CJ97] John Clark and Jeremy Jacob. A survey of authentication protocol literature : Version 1.0., November 1997.

[DH76] W. Diffie and M. Helman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6):644–654, november 1976.

[DS81] D. Denning and G. Sacco. Timestamps in key distributed protocols. *Communication of the ACM*, 24(8):533–535, 1981.

[GL01a] Jean Goubault-Larrecq. Langage de spécification de protocoles cryptographiques de eva: syntaxe abstraite et sémantique. Technical report, EVA, 2001.

[GL01b] Jean Goubault-Larrecq. Les syntaxes et la sémantique du langage de spécification eva. Technical Report 3, EVA, 2001.

[Gon89] Li Gong. Using one-way functions for authentication. *Computer Communication Review*, 19(5):8–11, october 1989.

[HC95] Tzonelih Hwang and Yung-Hsiang Chen. On the security of splice/as : The authentication system in wide internet. *Information Processing Letters*, 53:97–101, 1995.

[HLL+95] Tzonelih Hwang, Narn-Yoh Lee, Chuang-Ming Li, Ming-Yung Ko, and Yung-Hsiang Chen. Two attacks on neumann-stubblebine authentication protocols. *Information Processing Letters*, 53:103 – 107, 1995.

[HP00] Günther Horn and Bart Preneel. Authentification and payment in future mobile systems. *Journal of Computer Security*, 8:183–207, 2000.

[JLM01] Florent Jacquemard and Daniel Le Métayer. Langage de spécification de protocoles cryptographiques de eva: syntaxe concrète. Technical report, EVA, 2001.

[KC95] I Lung Kao and Randy Chow. An efficient and secure authentication protocol using uncertified keys. *Operating Systems Review*, 29(3):14–21, 1995.

[Kra96] Hugo Krawczyk. SKEME: A versatile secure key exchange mechanism for the Internet. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 114–127, feb 1996.

[lM90] Colin l'Anson and Chris Mitchell. Security defects in the ccitt recomendation x.509 - the directory authentication framework. *Computer Communication Review*, 20(2):30–34, april 1990.

[Low95] Gavin Lowe. An attack on the needham-schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, november 1995.

[Low96] Gavin Lowe. Some new attacks upon security protocols. In IEEE Computer Society Press, editor, *In Proceedings of the Computer Security Foundations Workshop VIII*, 1996.

[Low98] Gavin Lowe. Towards a completeness result for model checking of security protocols. Technical Report 1998/6, Dept. of Mathematics and Computer Science, University of Leicester, 1998.

[LR97] G. Lowe and A. W. Roscoe. Using CSP to detect errors in the TMN protocol. *Software Engineering*, 23(10):659–669, 1997.

[Mea95] C. Meadows. Formal verification of cryptographic protocols: A survey. In *Advances in Cryptology - Asiacrypt 94*, volume 917 of *LNCS*, pages 133–150. Springer-Verlag, 1995.

[NS78a] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), December 1978.

[NS78b] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), december 1978.

[NS93] B. Clifford Neumann and Stuart G. Stubblebine. A note on the use of timestamps as nonces. *Operating Systems Review*, 27(2):10–14, april 1993.

[NT94] B. Clifford Neuman and Theodore Ts'o. Kerberos : An authentication service for computer networks. Technical Report ISI/RS-94-399, USC/ISI, 1994.

[Pau01] Lawrence C. Paulson. Relations between secrets: Two formal analyses of the yahalom protocol. *J. Computer Security*, 2001.

[SMB90] Michael Merritt Steven M. Bellovin. Limitations of the kerberos authentication system. *Computer Communication Review*, 20(5):119–132, october 1990.

[Syv94] Paul Syverson. A taxonomy of replay attacks. In *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pages 131–136. IEEE Computer Society Press, 1994.

[TMN89] M. Tatebayashi, N. Matsuzaki, and D.B. Newman. Key distribution protocol for digital mobile communication systems. In *Advance in Cryptology — CRYPTO '89*, volume 435 of *LNCS*, pages 324–333. Springer-Verlag, 1989.

[VOP01] Open platform card specification, june 2001. Version 2.1.

[Wei99] Christoph Weidenbach. Towards an automatic analysis of security protocols. In Harald Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, volume 1632 of *LNAI*, pages 378–382. Springer, 1999.

[WL94] T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, 1994.