RAPPORT TECHNIQUE EVA

Coq certification for verification with HERMES tool

Date	:	12 Novembre 2003
Auteurs	:	Romain Janvier, Michaël Périn, Yassine Lakhnech, Liana Bozga
Titre	:	Coq certification for verification with HERMES tool
Rapport No. / Version	:	14/ 2

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

1 Description of a protocol and its properties in LAEVA

As an example we consider the Needham-Schroeder-Lowe protocol. Its specification is given below in LAEVA, the high level specification language designed in the EVA project for describing security protocols and their properties [GL02]. It is compiled by EVATRANS into a concrete operational model and a property to check that both constitute the inputs to three automatic verification tools developped in the EVA project: SECURIFY [CMR01], CPV [GL00] and HERMES [BLP03].

```
Needham_Schroeder
A, B : principal
N1, N2 : number
keypair pbk,prk (principal)
```

pbk and prk are key constructors that take a principal and return an asymmetric key: pbk(A) stands for public key of principal A. The private key of A, denoted by prk(A), is the inverse of pbk(A).

```
everybody knows pbk
A knows A, B, prk(A)
B knows B, prk(B)
```

The knowledge of the principals is needed to generate the operational model of the protocol ; it is used to rule out ambiguities.

```
1.A->B: {A, N1}_(pbk(B))
2.B->A: {N1, N2}_(pbk(A))
3.A->B: {N2}_(pbk(B))

s.session* {A,B,N1,N2}
```

*The protocol specification is close to the standard notation. It describes an ideal session between an initiator (role A) and a responder (role B). The roles A, B, and the nonces N1, N2 that they create are the parameters of a session. The * symbol asks the tool to consider an unbounded number of sessions in parallel. For debugging purpose, some tools (e.g., HERMES) can also run with a fixed number of sessions.*

```
assume
  secret(prk(A)@s.A),
  secret(prk(B)@s.B),
  secret(prk(B@s.A)),
  secret(prk(A@s.B))
```

`secret(prk(B@s.A)` *means that the private key – of the entity playing the role B, from A's point of view in session s – is unknown to the intruder. Secrecy hypothesis on keys are needed to reason about encrypted messages.*

```
claim
  *A*G secret(prk(A)@s.A),
  *A*G secret(prk(B)@s.B),
  *A*G secret(N1@s.A),
  *A*G secret(N2@s.B)
```

The three verification tools check that secrecy properties hold `*A`lways *and* `*G`lobally*. The first two claims require that the initial secrets remain secret. The two others asks that the nonces N1 and N2 created by role A (resp. role B) in session s are secret.*

The diagram of Figure 1 illustrates the relation between the specification of the Needham-Schroeder protocol given in LAEVA *(on dashed arrow)* and the process that describe the role of the initiator A and the role of the responder B *(in column)*.

1.1 Hermes front-end

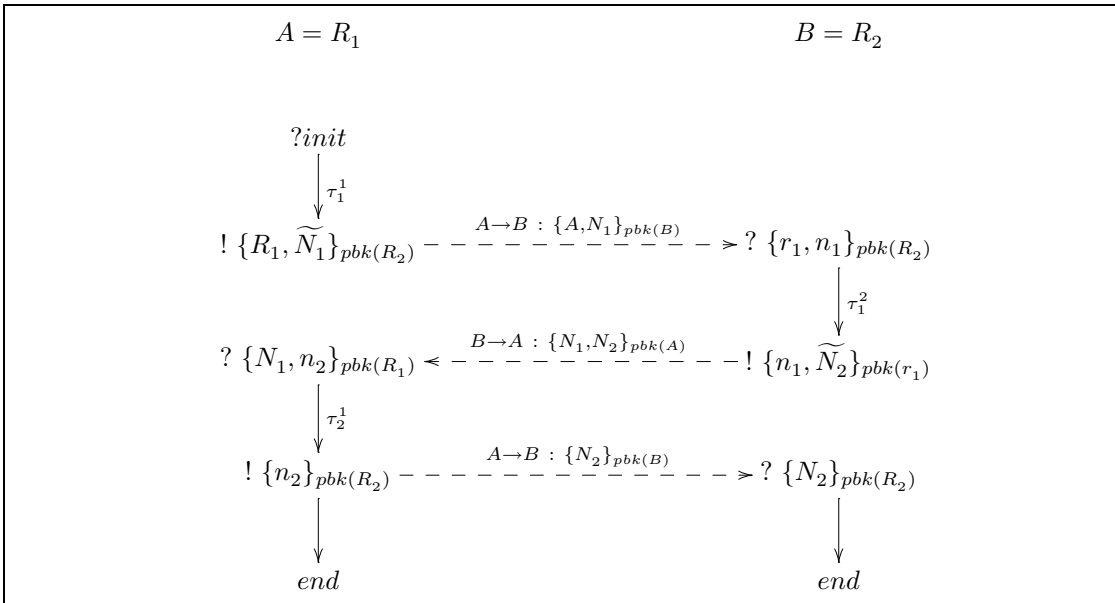
HERMES runs its computation on a protocol defined by the following transitions extracted from the LAEVA specification.

role R_1				*role* R_2			
(1.)	$?init$	$\xrightarrow{\tau_1^1}$	$!\{R_1, N_1\}_{pbk(R_2)}$				
				(2.)	$?\{R_1, n_1\}_{pbk(R_2)}$	$\xrightarrow{\tau_1^2}$	$!\{n_1, N_2\}_{pbk(R_1)}$
(3.)	$?\{N_1, n_2\}_{pbk(R_1)}$	$\xrightarrow{\tau_2^1}$	$!\{n_2\}_{pbk(R_2)}$				

They correspond to the transitions of role R_1 and R_2 in Figure 1. These transitions describe a generic session of the protocol, they are parameterized by (R_1, R_2, N_1, N_2). Then, a session of the protocol is completely defined by instantiating the roles R_1 and R_2 with some principals, and N_1, N_2 with two fresh nonces. Note that n_1 and n_2 denote free variables which are local to the session.

When the user asks for a verification with an unbounded number of sessions (set by the * symbol), HERMES computes an abstract model of the protocol. It considers only one honest principal H and one dishonest principal I (the intruder) and studies the following session instances:

$$(H, H, N_1, N_2),\ (H, H, N_1^{HH}, N_2^{HH}),\ (H, I, N_1^{HI}, N_2^{HI}),\ (I, H, N_1^{IH}, N_2^{IH})$$



- $A = R_1$ and $B = R_2$ means that the principal A plays the role 1 (initiator) and B plays the role 2 (responder).
- τ_i^r denotes the i^{th} transition of role R_r. A transition is a reception of a message that matches a pattern, written ?*pattern*, followed by a emission of a response message, written !*reponse*. Both parts of a transition are extracted from the specification of the protocol in LAEVA.
- $\widetilde{N}$ denotes the creation of a fresh nonce N and its first use.
- Capital letters like N_i, R_i are constants of the given session ; small letters denote variables.
- $pbk(p)$ is the public key of principal p.

Figure 1: Needham-Schroeder protocol specification

Generic session parameterized by (R_1,R_2,N_1,N_2)	The abstract models consists in four specific instantiations of the generic session: the fixed session (H,H,N_1,N_2)	sessions of type *HH* (H,H,N_1^{HH},N_2^{HH})	sessions of type *HI* (H,I,N_1^{HI},N_2^{HI})	sessions of type *IH* (I,H,N_1^{IH},N_2^{IH})
$\frac{init}{\{R_1,N_1\}_{pbk(R_2)}};$	$(\tau_1^1)\frac{init}{\{H,N_1\}_{pbk(H)}};$	$\frac{init}{\{H,N_1^{HH}\}_{pbk(H)}};$	$\frac{init}{\{H,N_1^{HI}\}_{pbk(I)}};$	$\frac{init}{\{I,N_1^{IH}\}_{pbk(H)}};$
$\frac{\{r_1,n_1\}_{pbk(R_2)}}{\{n_1,N_2\}_{pbk(r_1)}};$	$(\tau_1^2)\frac{\{r_1,n_1\}_{pbk(H)}}{\{n_1,N_2\}_{pbk(r_1)}};$	$\frac{\{r_1,n_1\}_{pbk(H)}}{\{n_1,N_2^{HH}\}_{pbk(r_1)}};$	$\frac{\{r_1,n_1\}_{pbk(I)}}{\{n_1,N_2^{HI}\}_{pbk(r_1)}};$	$\frac{\{r_1,n_1\}_{pbk(H)}}{\{n_1,N_2^{IH}\}_{pbk(r_1)}};$
$\frac{\{N_1,n_2\}_{pbk(R_1)}}{\{n_2\}_{pbk(R_2)}};$	$(\tau_2^1)\frac{\{N_1,n_2\}_{pbk(H)}}{\{n_2\}_{pbk(H)}};$ *only once with* (τ_1^1) *before* (τ_2^1)	$\frac{\{N_1^{HH},n_2\}_{pbk(H)}}{\{n_2\}_{pbk(H)}};$	$\frac{\{N_1^{HI},n_2\}_{pbk(H)}}{\{n_2\}_{pbk(I)}};$	$\frac{\{N_1^{IH},n_2\}_{pbk(I)}}{\{n_2\}_{pbk(H)}};$

Figure 2: Inference rules that define the abstract model used in HERMES

They suffice to model unbounded sessions. The first one is the distinguished session whom secrets are under attention. In this session, the honest principal H plays all the roles of the protocol without lost of generality. The other instances model the three possible type of sessions: between two honest principals (session HH), between one honest and one dishonest principal initiated either by the honest one (session HI) or by the dishonest one (session IH). These instances lead to four instantiation of the generic rules (see Figure 2).

```
s0.session(H,H,N1,N2)
s1.session(H,H,N1^HH,N2^HH)
s2.session(H,I,N1^HI,N2^HI)
s3.session(I,H,N1^IH,N2^IH)
```

Then, HERMES proceeds to the verification in the same way as for a finite number of sessions, except that the transitions of sessions `s1, s2, s3` can be played an infinite number of times, in any order, while for a finite number of sessions each transitions is played only once and each role fires its transitions one after the other. In formal terms, the abstract model corresponds to a set of rules on the right hand side of Figure 2: the nine inference rules of session `s1, s2, s3` which can be played any number of times and in any order ; plus the set of the three rules of session $\tau_1^1, \tau_2^1, \tau_1^2$ of the fixed session $(H, H, N1, N2)$ which are played only once and respecting the ordering constraint on transitions of each role. R_1 IMPOSES to fire τ_1^1 before τ_2^1. This is the only ordering constraint since R_2 has only one transition, τ_1^2.

1.2 Verification principle

From the abstract version of the protocol and the secrecy property, both extracted from the LAEVA specification of the protocol. HERMES computes what are the conditions that suffice to guarantee that messages sent by honest principals during a protocol session cannot be used by the intruder to get to know secrets that should be preserved during that session.

Given a protocol specification, a set S of secrets and a collection of hypotheses over those secrets a HERMES run produces:

1. A set G of message patterns which protect the secrets interchanged during a protocol session, so called *Good patterns*, and

2. A set B of message patterns which do not protect those secrets, called *Bad patterns*. The instantiations of these bad patterns are the messages that can be used to mount an attack.

The protecting messages are all ciphered message where K is a key whose inverse is not in the possession of the intruder. These messages can be derived from the protocol specification by using the keys that have been specified as secret keys in the hypothesis. The protecting messages are finitely represented by the set of *good patterns* G defined as all patterns $\{x\}_K$ where x denote a variable and K is a secret key (given in the hypotheses).

The output of HERMES (the algorithm is shown to be terminating) consists of:

1. A set S' of secrets, which contains in particular the declared secrets of S, and

2. A set P of protecting messages, which is defined as the difference of two sets. The protecting messages are the instances of a good pattern which are not an instance of a bad pattern, that is formally:

$$P = \mathit{Instance}(G) \setminus \mathit{Instance}(B)$$

The Secrecy property is verified if, in every message initially known by the intruder, every secret belonging to S' is embedded messages that belong to P.

Correctness of the verification principle We briefly present the verification principle implemented in HERMES for it is useful to understand some technical points in Section 1.5 about interpretation of HERMES output. We select the knowledge that should make a user at ease with HERMES. We here concentrate on the general idea and keep out the technical material and formal definitions. The verification principle implemented in HERMES is fully described in [EVA-RT5].

The verification of a protocol $\mathscr{P}$ described as a set of generic transitions is a fixpoint computation. HERMES starts with a given set of secrets S, the set of good patterns $G = \{ \{x\}_k \mid k^{-1} \in S \}$ and the empty set of bad patterns $B = \emptyset$. It explores all possible sequence of transitions of the protocol and reduces the set of protecting messages (by adding patterns to B) and augments the set of secret S until it reaches a stable triple (S', G, B') such that:

> *whatever the messages sent along this sequence of transitions, the secret of S' are always embedded in a protecting messages – defined as Instance$(G) \setminus$ Instance(B').*

The correctness of HERMES verification relies on the following proposition.

> Consider a protocol $\mathscr{P}$, a set S of secrets, two sets, G and B, of good and bad patterns. Assume that the triple (S, G, B) is a fixpoint for HERMES computation on the protocol $\mathscr{P}$, then we conclude that:
>
> > *if, in the messages initially known by the intruder, the secrets of S are embedded in protecting messages defined by the pair (G, B), then, the secrets are protected in all messages that the intruder can deduce when playing any number sessions of the protocol $\mathscr{P}$.*

This result is formally stated by Proposition [11] of Figure 3 and it is automatically proved in COQ for a given protocol $\mathscr{P}$, secrets S and patterns G, B computed by HERMES.

1.3 Hermes results: Proof obligations

The results of Hermes must be interpreted as constraints that defines the acceptable initial knowledge of the intruder. These constraints can be interpreted as conditions on the manner the protocol must be used. In the context of the EVA project, however, the initial conditions are fixed, and they define what is the information known by the intruder. Therefore the constraints should actually be understood as proof obligations.

The initial conditions of a protocol are specified in the LAEVA language by using declarations of the form `assume *A*G secret(K@s.A)`. This declaration states that the value of the key K (in session s and from A's point of view) is supposed to be unknown to the intruder. Hypotheses like this one are used in order to develop the proof that the constraints imposed by HERMES are satisfied.

1.4 Hermes back-end: counter example or correctness proof

HERMES is provided with a back-end that helps in interpreting the results (S', G, B). The back-end includes an heuristic algorithm to detect if the extended set of secrets S' contains a message that can be forged by the intruder. For instance, a message m in S' cannot be secret if it is only made of public knowledge (the identity of principals, their public keys) and data known by the intruder (e.g., shared in sessions between honest principals and the intruder). In this case, HERMES detects that the proof obligations cannot be fullfiled and provides the sequence of transitions that has led to a requirement of secrecy on m.

Counter example This sequence can be interpreted as the trace of an attack in the abstract model. As it is, it can be difficult to understand without a deep knowledge in HERMES verification principle. So, the back-end provides a counter example feature that computes a corresponding attack on the concrete model and shows it graphically as a Message Sequence Chart (see Section 1.6 and Figure 4 for an example).

All the traces explored by HERMES are summarize graphically as a tree whom leaves conclude either with an attack on a secret, or with a stabilization of computation that allows to cut the exploration of the branch. All the traces leading to an attack leaf can be given to the counter example generator.

Correctness proof in coq On the other hand, when no attack is detected, HERMES conclude to the correctness of the protocol under some assumption on the initial knowledge of the intruder. In order to emit a certificate based on HERMES results, the certifying authority do not need to check and gain confidence neither in the verification principle underlying HERMES, nor in its implementation. To leave no doubt on the results of the verification, HERMES ends with a *proof of correctness of the protocol* that can be replayed in COQ proof-checker.

1.5 Certifying Hermes computation

This section enumerates what the user should admit to recognize the validity of the correctness proof. A COQ theory is a collection of type definitions, predicate and functions definitions, and axioms which are the basis to express and to prove the wanted proposition. To prove a secrecy property of a given cryptographic protocol $\mathscr{P}$ in the presence of an intruder, the theory must contains:

1. the definition in COQ of the execution model of a protocol (predicate definition [6] of Figure 3).
2. the translation in COQ of the specification of protocol $\mathscr{P}$ and of its secrecy property S (definitions [7] of Figure 3).
3. the definition in COQ of the inference rules of Dolev-Yao's model that defines the intruder capabilities (predicate definition [3] of Figure 3). Henceforth, the proof of correctness ensures that the protocol is not vulnerable to an intruder with deduction capabilities and that controls the network. But it tells nothing on an intruder with other capabilities like, for instance, those of guessing a week key or weak password.
4. the proposition that states that protocol $\mathscr{P}$ satisfy the secrecy property S (proposition [11] of Figure 3).

A certifying authority only has to check these four points to recognize the "`proved`" answer of COQ engine as a proof of correctness of the protocol. This task should be very light since the specification in COQ of the protocol, the secrets, and all definitions are obtained by direct translations. Obviously, this approach requires to put confidence in the COQ proof-checker. Figure 3 describes the COQ theory generated by HERMES back-end to prove that a given protocol $\mathscr{P}$ satisfies the secrecy property S.

- The verification principle of HERMES bears on Predicate [2] $E\langle G, B\rangle S$ which is true if in all message of E, the secrets of S are embedded in the protecting message defined by the pair (G, B). This predicate is defined as a check on the structure of messages in E, S, G, B. It does not directly use the definition of Dolev-Yao's intruder deduction.
- Predicate [3] $E \vdash m$ is true if the message m can be deduced from the set of messages E using Dolev-Yao's inference system; This predicate defines the intruder deduction capabilities.
- Predicate [4] $E \nvdash m$ states that a message m cannot be deduced from the set E of messages. It is useful to express the secrecy property.
- Proposition [5] relates the predicate $E\langle G, B\rangle S$, specific to HERMES, to the predicate $E \vdash m$ which defines Dolev-Yao's model. This proposition is proved in COQ, thus ensuring that HERMES predicate gives a sufficient condition to avoid deduction by the intruder.

- Predicate [6] defines the concrete execution model: $(E,\Omega) \xrightarrow{\tau} (E',\Omega')$ is true if the transition τ leads from the state (E,Ω) to the state (E',Ω') where:
  - Ω, Ω' denote sets of protocol sessions in execution. In particular, this data structure records the next fireable transitions (see [EVA-RT5] for details on the concrete model).
  - τ can either be a fireable transition in a session of the protocol that will be fired and deleted from Ω, or the creation of a new session that will be added to Ω.
  - E, E' denote sets of messages that represent the knowledge of the intruder, that is all messages that were sent on the network.
- Proposition [9] is an easy check that is completed automatically using COQ tactic.
- Proposition [11] is proved from [10] by induction on the length of the trace $\langle \tau_1 \dots \tau_n \rangle$, followed by an application of Propositions [5] and [9]. So, assuming that Proposition [10] is established, the proof of [11] is no more dependent of $\mathscr{P}, S, S_h, G_h, B_h$. Then, the proof steps have been recorded to be replayed as they are at each protocol verification.
- Eventually, Proposition [10] is the main proposition to proved. This proposition tells that the predicate $E\langle G_h, B_h \rangle S_h$ is stable with respect to any transitions of any sessions of the protocol $\mathscr{P}$. This is exactly the condition used in HERMES to conclude that it reached a fixpoint (S_h, G_h, B_h). Hence, the last computation step of HERMES that concludes to the stability of the predicate produces the arguments needed for the automatic proof of Proposition [10]. This proof is the only one that is specific to the protocol $\mathscr{P}$ and secrets S. It has been automated using tactics extracted from HERMES computation strategy. The tactic mimics the case that are considered by HERMES and recursion in HERMES computation corresponds to calls to the induction tactic in the proof.

  The certification feature of HERMES outputs a file for the COQ engine. It contains the theory of Figure 3 ; the proof of all propositions which are independent of $\mathscr{P}$ and S ; and some tactics dedicated to the specific proof of Proposition [10].

  When this file is loaded into the COQ engine, the proof-checker prints all the proved lemmas related to Proposition [10]. The proof is split in several lemmas: each one considers one transition of the protocol $\mathscr{P}$. Then, gathering all this lemmas leads to proposition [10].

  This proof cannot be done using a general tactic as for other propositions. Indeed, it depends on HERMES computation in the following way: COQ asks for a witness to prove some existential property that appear in the proof of Proposition [10]. The witness we need are recorded during HERMES run and a specific tactic is generated that provides the witness computed by HERMES at the right place in the proof.

Note that HERMES do not explicitly play a part in the proof. It is only used to compute a fixpoint (S_h, G_h, B_h) for a given protocol $\mathscr{P}$ and secrets S. Then, the property of this fixpoint and the conclusion it implies are checked by COQ. Hence, making the proof independent of HERMES theory and implementation.

1.6 Attack as Message Sequence Charts

HERMES computations take place on the abstract model. So, it deals with abstract traces which collapsed all honest principals on the principal `H` and all intruders on `I` ; it also merges the nonces that are indistinguishable due to the abstraction. The abstract traces are produced by a backward computation ; they gives the essence of the attack but they forget about the messages that are needed to get an effective attack.

For a user that is not interested in HERMES principle, it is easier to get the attack on the concrete model illustrated as a message sequence chart (MSC). We developed a prototype that take a raw abstract attack given by HERMES and provides a concrete one: it computes the number of sessions, nonces and principals needed to get a concrete attack then it fills the abstract trace with the missing part of the protocol.

General theory for proving property of cryptographic protocol in Dolev-Yao's model

`type:`	*message type*	[1], `definition`
`def:`	$E\langle G, B\rangle S$	[2], `predicate definition using` [1]
`def:`	$E \vdash m$	[3], `predicate definition using` [1]
`def:`	$E \nvdash S \stackrel{def}{=} \forall m.\ E \vdash m \Rightarrow m \notin S$	[4], `predicate definition using` [3]
`prop:`	$E\langle G, B\rangle S \Rightarrow E \nvdash S$	[5], `proved` $\forall_{E,G,B,S}$

Theory specific to the protocol $\mathscr{P}$ and its secrets S.

`def:`	$(E,\Omega) \xrightarrow{\tau} (E',\Omega')$ *(for protocol transitions and session creation)*	[6], `predicate definition`
`def:`	$\mathscr{P}, S$ *extracted from the* LAEVA *specification*	[7], `definition using` [1]
`def:`	S_h, G_h, B_h *computed by* HERMES *for* $\mathscr{P}$ *and* S	[8], `definition using` [1]
`prop:`	$S \subseteq S_h$	[9], `proved`
`prop:`	$E\langle G_h, B_h\rangle S_h \Rightarrow \forall \tau \in \mathscr{P}.\ (E,\Omega) \xrightarrow{\tau} (E',\Omega') \Rightarrow E'\langle G_h, B_h\rangle S_h$	[10], `proved` $\forall_{E,E',\Omega,\Omega'}$
`prop:`	$E_0\langle G_h, B_h\rangle S_h \Rightarrow \forall \langle \tau_1 \dots \tau_n\rangle \in \mathscr{P}^*.\ (E_0,\Omega) \xrightarrow{\tau_1} \cdots \xrightarrow{\tau_n} (E',\Omega') \Rightarrow E' \nvdash S$	[11], `proved` $\forall_{E_0,E',\Omega,\Omega'}$

Figure 3: Proof in COQ certifying the correctness of HERMES results on protocol $\mathscr{P}$ and secrets S

Example We consider the running example of Needham-Schroeder protocol specified in LAEVA in Section 1. From this specification, HERMES computes the abstract model described in Section 1.1, then the verification process leads to the following trace of an abstract attack:

$$!\{H, N_1^{HI}\}_{pbk(H)} \rightarrow ?\{H, n_1\}_{pbk(H)} \stackrel{\tau_1^2}{\rightarrow} !\{n_1, \widetilde{N_2^\star}\}_{pbk(H)} \rightarrow ?\{N_1^{HI}, n_2\}_{pbk(H)} \stackrel{\tau_2^1}{\rightarrow} !\{n_2\}_{pbk(I)}$$

where

- All nonces and roles are annotated by the session to which they belong. For instance, N^π is a nonce created for the session π ; $N^\star$ is a nonce of the distinguished session, denoted by $\star$ that involves two honest principals.
- H denotes the honest principal, and I denotes the intruder.
- N_1^{HI} is a constant that represents the nonce N_1 in a session between H and I.
- $N_2^\star$ is the second nonce of *the distinguished session* (denoted by $\star$) between two honest principals both represented by H. The secrecy property of Needham-Schroeder protocol states that $N_2^\star$ must remain secret.

The counter example generator concretizes and fills this trace to come to the Message Sequence Chart of Figure 4 where:

- horizontal arrows correspond to message passing between two roles and vertical arrows correspond to transition of a role.
- bold arrows come from the abstract trace given by HERMES; all the other ones (transitions, dashed and plain arrows) are added automatically by the process that reconstruct the concrete attack.
- dashed arrows do not correspond to actual communication; they denote how a message is interpreted (following the protocol) by the principal which receive the message
- (τ)-transitions are played in accordance with the protocol of Figure 1, ($\vdash$)-transitions denote computation of the intruder.

- π and $\star$ are session identifiers:
  - π is a session between one honest principal R_1^π and one dishonest principal R_2^π.
  - $\star$ is a session between two honest principals $R_1^\star$ and $R_2^\star$.
- All constants and variables are annotated (in exponent) with a session identifier. This indicates the session to which they belong.
- Every principal knows the public key $pbk(p)$ of any principal p.
- The secrecy property (S_{NS}) of Needham-Schroeder protocol specifies that $N_2^\star$, the nonce created by $R_2^\star$ in session $\star$, should only be known by $R_1^\star$ and $R_2^\star$.

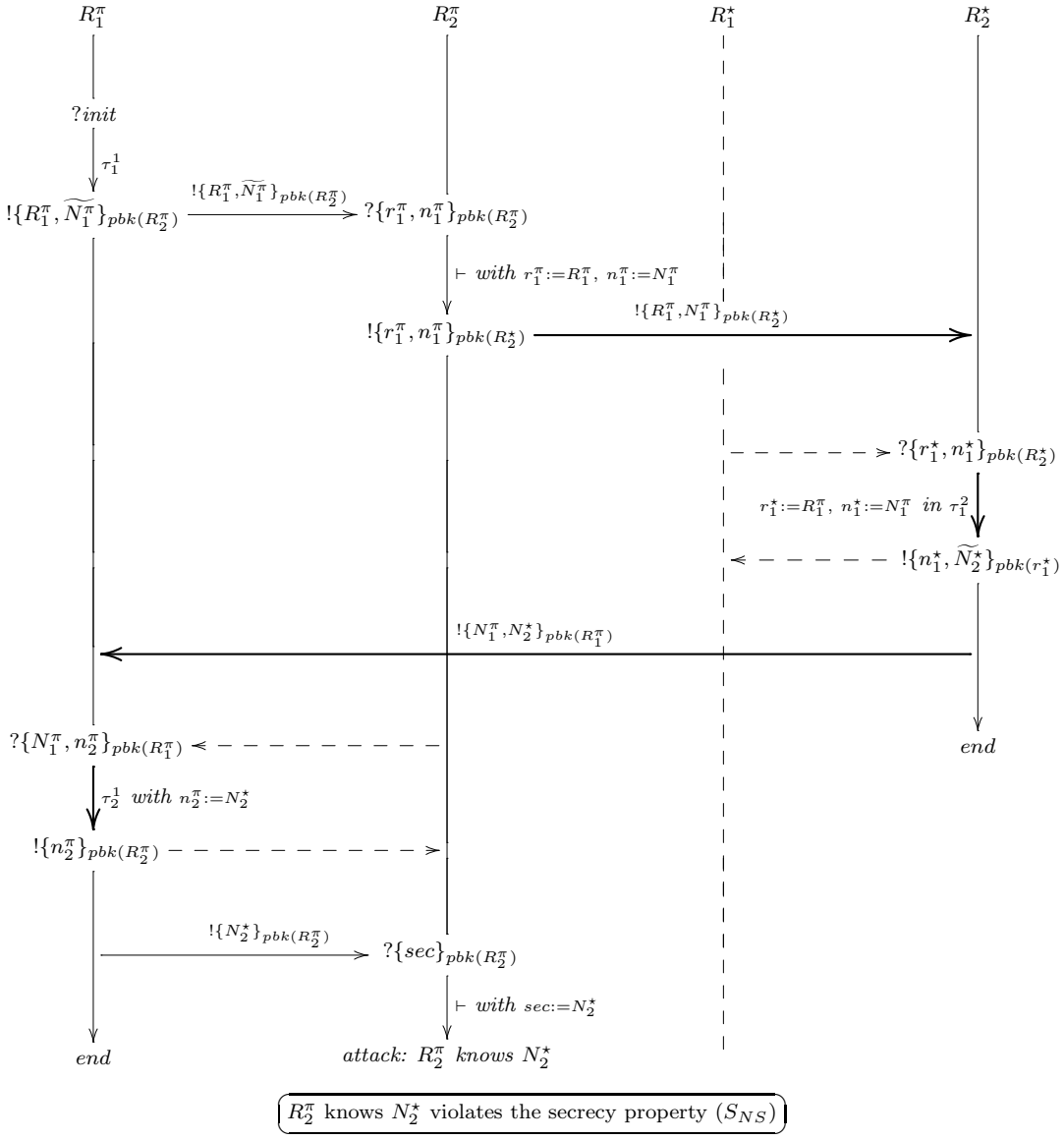


Figure 4: MSC describing the "man in the middle attack" of Needham-Schroeder protocol

It can happen that HERMES notify an attack which is not effective on the concrete model. This unprecise result are unavoidable: *false attack* are the so-called *inconclusive answer* in the abstract interpretation theory. They come necessarily with abstraction. If an attack detected by HERMES is not an actual attack in the concrete model, then:

- either the counter example generator will fail to compute the number of sessions needed, the user won't get a concrete attack. Then, the user has to examine HERMES trace to check if the abstract attack corresponds to an actual attack in the concrete model.
- or, it will produce an attack that actually cannot happen in the concrete model – this cannot be detected automatically since it is a problem known to be as hard as the verification problem itself.

The following table summarize the results obtained by HERMES on protocols from [EVA-RT4] regarding secrecy properties. In the Table 1, "OK" means that the protocol has been successfully verified for the secrecy property ; "ATTACK" means that an attack have been found. In general, a third result, "INCONCLUSIVE" can be obtained because of abstraction step: it is possible to find an attack at abstract level which is not valid on the concrete level. Surprisingly we have not encountered any false attack on any practical protocol. Although, one could construct a protocol whose verification leads to a false attack.

Protocol Name	Result	Time (sec)
Yahalom	OK	12.67
Needham-Schroeder Public Key	ATTACK	0.01
Needham-Schroeder Public Key (with a key server)	ATTACK	0.90
Needham-Schroeder-Lowe	OK	0.02
Otway-Rees	OK*	0.02
Denny Sacco Key Distribution with Public Key	ATTACK	0.02
Wide Mouthed Frog (modified)	OK	0.01
Kao-Chow	OK	0.07
Neumann-Stubblebine	OK*	0.04
Needham-Schroeder Symmetric Key	ATTACK	0.04
ISO Symmetric Key One-Pass Unilateral Authentication	ATTACK	0.01
ISO Symmetric Key Two-Pass Unilateral Authentication	OK	0.01
Andrew Secure RPC	ATTACK	0.04
Woo and Lam	OK	0.06
Skeme	OK	0.06

* There is a known attack of the untyped version of the protocol. Discovering this type attack automatically requires to deal with non-atomic keys. This is not yet implemented in HERMES.

Table 1: This figure has been obtained by running HERMES on a Pentium III 600MHz PC under Linux.

References

[BLP03] L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. In H. Garavel and J. Hatcliff, editors, *Proceedings of TACAS'03 - 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Warsaw, Poland*, volume 2619 of *LNCS*, pages 299–314. Springer, April 2003.

[CMR01] V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *Proceedings of CSFW'01 - 14th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada*, pages 97–110. IEEE Computer Society Press, June 2001.

[GL00] J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In Jose Rolim, editor, *Proceedings of 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, Cancun, Mexico*, volume 1800 of *LNCS*, pages 977–984. Springer, 2000.

[GL02] J. Goubault-Larrecq. La syntaxe et la sémantique du langage de spécification eva. Technical report, Projet EVA, `http://www-eva.imag.fr/`, November 2002.