HERMES: An Automatic Tool for Verification of Secrecy in Security Protocols*

Liana Bozga, Yassine Lakhnech, and Michaël Périn

Verimag, Centre Équation, 2 av. de Vignate, 38610 Gières, France
{lbozga,lakhnech,perin}@imag.fr

1 Introduction

Cryptography is not sufficient for implementing secure exchange of secrets or authentification. Logical flaws in the protocol design may lead to incorrect behavior even under the idealized assumption of perfect cryptography. Most of protocol verification tools are model-checking tools for bounded number of sessions, bounded number of participants and in many case also a bounded size of messages [11,8,5,10]. In general, they are applied to discover flaws in cryptographic protocols. On the contrary, tools based on induction and theorem proving provide a general proof strategy [9,4], but they are either not automatic with exception of [4] or the termination is not guaranteed.

In this paper, we present Hermes, a tool dedicated to the verification of secrecy properties of cryptographic protocols. Hermes places no restriction on the size of messages, neither on the number of participants, nor on the number of sessions. Given a protocol and a secret, Hermes provides either an attack or an invariant on the intruder knowledge that guarantees that the secret will not be revealed by executing the protocol. Moreover, when a protocol is proved correct, it returns a proof tree that can be exploited for certification. Hermes is available online from the authors' webpage.

2 The Model and the Verification Method

We give a sketchy idea of the verification method underlying Hermes. A formal and complete presentation of this method can be found in [1]. Cryptographic protocols can be modeled as a set of transitions of the form $t \rightarrow t'$ where t, t' are *terms* contructed by applying pairing and the encryption operator $\{\_\}_K$, to some free variables and the *parameters of a session*, which are the principals, the fresh nonces, and the fresh keys of the session. The intruder is modeled by additional transitions due to Dolev-Yao [6]. They can be seen as a deductive system that describes the messages that the intruder can deduce and forge from the messages sent during the protocols execution. A secrecy goal states that several designated messages (the *secrets*) should not be made public: a secret s is defined by a term too.

* This work and the developpement the LaEva language and accompanying tools EvaTrans, Securify, Cpv and Hermes was supported by the RNTL project EVA (Explication et Vérification Automatique de Protocoles Cryptographiques).

W.A. Hunt, Jr. and F. Somenzi (Eds.): CAV 2003, LNCS 2725, pp. 219–222, 2003.

2.1 Description of a Protocol and Its Properties

Along the paper we illustrate the main step of the verification on the Needham-Schroeder-Lowe Protocol. Its specification is given below in LaEva, a high level specification language designed for describing security protocols and their properties. 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 [4], Cpv [7] and Hermes [1].

```
Needham_Schroeder_Lowe
A, B : principal
Na, Nb : 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, Na}_(pbk(B))
2.B->A: {Na, Nb,B}_(pbk(A))
3.A->B: {Nb}_(pbk(B))

s.session* {A,B,Na,Nb}
```

*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 Na, Nb that they create are the parameters of a session. With the * symbol,* Hermes *considers an unbounded number of sessions in parallel. For debugging purpose, it 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(Na@s.A),
  *A*G secret(Nb@s.B)
```

Hermes *checks that secrecy properties hold* *Always *and* *Globally*. The first two claims require that the initial secrets remain secret. The two others asks that the nonces Na and Nb created by role A (resp. role B) in session s are secret.*

This specification leads to three rules parameterized by (A, B, N_a, N_b) (see below). Then, a session of the protocol is completely defined by instantiating the roles A and B with some principals, and N_a, N_b with two fresh nonces. On the other hand, n_1 and n_2 denote free variables which are local to the session.

$$(1.)\ \frac{-}{\{A, N_a\}_{pbk(B)}}; \qquad (2.)\ \frac{\{A, n_1\}_{pbk(B)}}{\{n_1, N_b, B\}_{pbk(A)}}; \qquad (3.)\ \frac{\{N_a, n_2, B\}_{pbk(A)}}{\{n_2\}_{pbk(B)}};$$

2.2 Abstraction and Verification Steps

We reduce the model with unbounded number of sessions in parallel to an finite set of rules which can be applied infinitely and in any order. The rule are

obtained by applying the safe abstraction of [3] that distinguishes only one honest principal H and the intruder I. Identifying principals means also identifying their nonces and keys. Hence, the session to consider are reduced to: the session we observe, *i.e.* (H, H, N_a, N_b), the sessions with a dishonest participant $(I, H, N_a^{IH}, N_b^{IH})$ and $(H, I, N_a^{HI}, N_b^{HI})$, and the others sessions between honest participants $(H, H, N_a^{HH}, N_b^{HH})$, see [1] for details.

Given this abstract model of the protocol, HERMES computes the set of messages that protect the secrets in all sent messages. Intuitively, such protections are terms of the form $\{x\}_K$ where the inverse key K^{-1} is not known by the intruder. The hypotheses on secret keys are used to define an initial superset P of protecting messages, which is then refined by a fixpoint computation. At each step, one transition $t \rightarrow t'$ of the abstract protocol is analyzed and bad protections are removed from P or new secrets are added to S, see [1] for details.

The infinite set P of protections is represented as two sets (G, B) of terms. Roughly speaking, the pair (G, B) is meant for the set of ground messages that are instances of (good) terms in G but do not unify with (bad) terms of B.

We provided HERMES with a widening operator that forces termination in the presence of rules that can loop and produce an infinite set of growing bad terms. At termination, HERMES yields an augmented set of secrets S' $(S \subseteq S')$ and either a representation of a set of protecting messages P' $(P' \subseteq P)$ that protect S' or an attack in case a secret is revealed.

2.3 Application to Our Running Example

We illustrate our approach on the abstract model of the Needham-Schroeder-Lowe Protocol. HERMES starts with the set of secrets $S = \{N_a, N_b, prk(H)\}$, the set of good patterns $G = \{\{x_s\}_{pbk(H)}\}$ and an empty set of bad patterns. The fixpoint computation leads to the same set of secrets, the same good patterns and an augmented set of bad protections B' that consists in four patterns:
$\{N_a^{HI}, x_s, I\}_{pbk(H)}$; $\{N_a^{HH}, sup(x_s, I), H\}_{pbk(H)}$; $\{x_s, I\}_{pbk(H)}$; $\{N_a, sup(x_s, I), H\}_{pbk(H)}$

The first one of these bad patterns is obtained from $\frac{\{Na^{HI}, x_s, I\}_{pbk(H)}}{\{x_s\}_{pbk(I)}}$, which is the third rule of session $(H, I, N_a^{HI}, N_b^{HI})$ in the abstract model. Since the conclusion of this rule reveals the secret x_s to the intruder, we have to remove from the good protections the particular case of messages of the form $\{N_a^{HI}, x_s, I\}_{pbk(H)}$. HERMES stops when no new bad pattern or secret are generated. We then conclude that the protocol satisfies the secrecy property in all initial configurations where the secrets in S' appears only in messages in compliance with (G, B'). In our example, the nonces N_a and N_b of S' do not put constraints on the initial configurations since they do not appear at all before the observed session. Finally, the set of secrets S' restricts the initial configurations to those where the key $prk(H)$ was never sent, or sent with a protection in compliance with (G, B').

In the original version of the protocol due to Needham-Schroeder, the identity of role `B` did not appear in message 2. Applied to this version, HERMES requires $\{H, N_a^{HI}\}_{pbk(H)}$ to be secret. Obviously, this message can be forged by the intruder. The exploration tree computed by HERMES shows an attack that reveals the secret N_b. This corresponds to the well known attack discovered by Lowe.

3 Experimental Results and Future Work

The following table summarizes the results obtained by HERMES for secrecy properties of protocols from Clark-Jacob's survey [2]. Surprisingly we have not encountered any false attack on any of these protocols, although one could construct a protocol that leads to a false attack. We are currently working on extracting a proof for the Coq prover from the exploration tree provided by HERMES.

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

* 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.

References

1. L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. In *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2619 of *LNCS*, p. 299–314, 2003.
2. J. Clark and J. Jacob. A survey on authentification protocol literature. Available at the url `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.
3. H. Comon-Lundh and V. Cortier. Security properties: Two agents are sufficient. In *European Symposium On Programming*, vol. 2618 of *LNCS*, p. 99–113, 2003.
4. V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *Computer Security Foundations Workshop*, p. 97–110, 2001.
5. G. Denker and J. Millen. The CAPSL integrated protocol environment. In *IEEE DARPA Information Survivability Conference and Exposition*, p. 207–222, 2000.
6. D. Dolev and A. C. Yao. On the security of public key protocols. *Transactions on Information Theory*, 29(2):198–208, 1983.
7. J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *International Workshop on Formal Methods for Parallel Programming: Theory and Applications*, vol. 1800 of *LNCS*, p. 977–984, 2000.
8. G. Lowe. Casper: A compiler for the analysis of security protocols. In *Computer Security Foundations Workshop*, p. 18–30, 1997.
9. L. Paulson. Proving properties of security protocols by induction. In *Computer Security Foundations Workshop*, p. 70–83, 1997.
10. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Computer Security Foundations Workshop*, p. 174–190, 2001.
11. S. Schneider. Verifying authentication protocols with CSP. In *Computer Security Foundations Workshop*, p. 3–17, 1997.