HERMES: Automatic Verification of Secrecy in Security Protocols*

L. Bozga, Y. Lakhnech, and M. Périn

Verimag, Centre Equation, 2 av. de Vignate, 38610 Gieres, France

Abstract. In this paper we present a tool, called HERMES, for the verification of secrecy properties of cryptographic protocols. Given a protocol and a secret property, HERMES computes conditions on the initial intruder knowledge that guarantee that the secret is never going to be revealed. The result is obtained without any restriction on the size of messages, on the number of sessions nor on the number of participants. Using HERMES, we are able to analyze all the protocols of Clarck-Jacob's survey [5].

1 Introduction

Cryptographic protocols play a major role in any application where data integrity, confidentiality, authenticity and other security related properties are crucial. Such applications include Smart-Cards, e-business and internet-based voting.

A cryptographic protocol can be seen as a set of rules defining the messages that have to be exchanged between a fixed set of participants, called principals, to implement a functionality such as secret message exchange or authentification. The exchanged messages contain encrypted parts that should atmost be communicated to the participants.

Cryptography is not sufficient for implementing secure message exchange. Indeed, logical flows in the protocol design may lead to incorrect behavior with undesired consequences [13] even under the idealized assumption of perfect cryptography. It is logical flows that HERMES helps to identify, respectively, to prove their absence. The verification of security protocols is challenging for at least two reasons:

- the modeling of security protocols involves many subtle points regarding for instance, modeling the intruder, participant capabilities, properties,
- reachability is decidable only for the bounded number of sessions [1, 18, 15] or when nonce creation is not allowed and the size of messages is bounded [9]. These assumptions are in practice rarely justified.

In this paper, we describe the tool HERMES which allows the verification of bounded and unbounded protocols. HERMES is part of a validation toolbox for security protocols developed in the french national project EVA. The toolbox is illustrated in figure 1. The entry point in the toolbox is LAEVA, a high level specification languge for security protocols and their properties. Using an automatic translator [12] LAEVA specifications are flatenned into an intermediate representation.

This intermediate representation is actually the common input of three automatic verification tools: SECURIFY [7], CPV [11] and HERMES.

Most of protocol verification tools are model-checking tools and, in general, they have been applied to discover flaws in cryptographic protocols. Usually these tools bound the number of session, the number of participants [19, 14, 8] as well as the size of message [14]. There exist also tools based on induction and theorem proving providing a general proof strategy [17, 2, 7], but they are not automatic with exception of [7] and the termination is not guaranteed. The closest work to ours are partial algorithms based on abstract interpretation and tree automata that have been presented in [16, 11, 10]. But, HERMES computes an invariant on the intruder knowledge and not the set of messages that can be known by the intruder. Moreover, when a protocol is proved correct, it provides a proof tree that can be exploited for certification.

* This work was supported by the RNTL project EVA (Explication et Verification Automatique pour les Protocoles Cryptographiques).

2 The verification method underlying Hermes

In this section, we give a sketchy idea of the verification method underlying Hermes. A formal and complete presentation of this method can be found in [3, 4].

Cryptographic protocols can be modeled as a set of transitions of the $t \to t'$ where t, t' are terms obtained by applying encryption and pair functions to parameters and free variables. The parameters are the involved principals, the fresh nonces and the used keys. In addition to the principals transitions, the intruder is modeled by a set of transitions that can be seens as a deductive system. This system describes the messages the intruder can forge and analyze from the messages that have been sent during the protocols execution. This is the Dolev-Yao model.

A secrecy goal states that several designated messages (the *secrets*) should not be made public. In our setting, a secret s is defined by a term too.

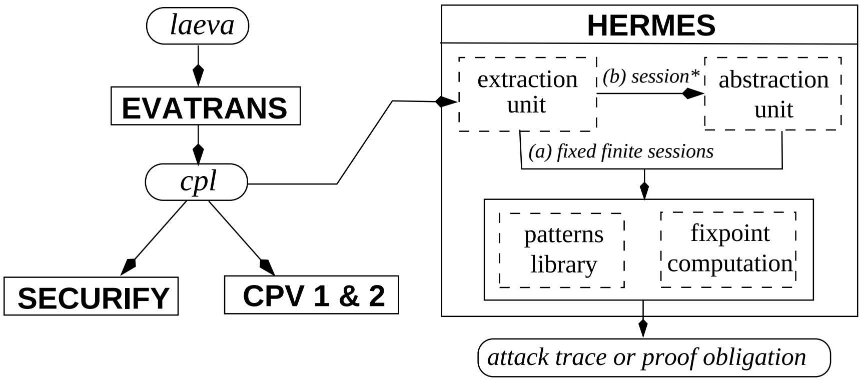

Fig. 1. The architecture of the Eva toolbox and the internal design of Hermes.

2.1 Description of a protocol and its properties

Along the paper we ilustrate each step of the verification on the Needham-Schroeder-Lowe Protocol. The following is the specification of the protocol in LaEva, a high level specification language for the description of security protocols and their properties. This specification is automatically translated into a concrete operational model in *cpl* and a property to check that both constitute the entry point of Hermes (see Figure 1).

```
Needham_Schroeder_Lowe

alg : asym_algo
A, B : principal
Na, Nb : number
keypair^alg pbk, prk (principal)
```

This part consists in declarations of typed variables and two associated key constructors `pbk` and `prk` that take a principal and return an asymetric key. The constructor `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 alg
everybody knows pbk
A knows A, B, prk(A)
B knows B, prk(B)
```

This part specifies the knowledge of the principals. It is necessary to elevate ambiguities when we generate the formal model of the protocol from the description of the messages exchanged during a session of the protocol, see below.

```
{
1. A->B : {A, Na}_(pbk(B))^alg
2. B->A : {Na, Nb,B}_(pbk(A))^alg
3. A->B : {Nb}_(pbk(B))^alg
}
```

The protocol is described in a style close to the standard notation. It defines the order of the messages exchanged during an ideal session. The role of initiator and responder – respectively denoted by `A` and `B` – and the nonces `Na` and `Nb` that they create are the parameters of a session of the protocol.

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

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 (see Figure 1).*

<pre>assume secret(prk(A)@s.A),
       secret(prk(B)@s.B),
       secret(prk(B@s.A)),
       secret(prk(A@s.B))</pre>
	This part provides the secrecy hypothesis that are exploited in HERMES*'s reasoning. It defines keys unknown to the intruder which can be used to safely encrypt messages.* `secret(prk(B@s.A)` *means that the private key – of the entity playing the role of the responder (*`B`*) from* `A`*'s point of view in session* `s` *– is secret.*

<pre>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)</pre>
	The `*A*G` *is required in claims for* 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.*

Using EVATRANS, this specification is compiled into the three rules that define a generic session. Then, a session of the protocol is completely defined by instantiating the parameters (A, B, Na, Nb) in the generic rules: the roles A and B with some principals ; and Na and Nb with two fresh nonces. On the other hand, n_1 and n_2 denote local variables of the session, they remain free.

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

Generic rules of the concret model parameterized by (A,B,Na,Nb)

2.2 The model and the abstraction step

The verification starts with an abstraction step that reduces the concrete model where an unbounded number of protocol sessions are possible to an abstract model where only sessions between a finite number of participants have to be considered.

The idea is the following. First, we fix an arbitrary session where the same *honest principal*, say H, plays the two different roles in the protocol. Then, we identify *the intruder* I with all principals other than H. As a consequence, all sessions in which H is not involved are captured by the Dolev and Yao's model of the intruder's capacities. Concerning the other sessions where H is involved, we identify: (1) all sessions where H plays all roles and which are different from the fixed session, (2) all sessions where H plays the first role while the second role is played by an other principal and (3) all sessions where H plays the second role while the first role is played by another principal. Reducing all principals to one honest principal H and an intruder I has been proved to be a correct abstraction in [6]. Identifying sessions means also identifying to specific constants, the nonces and the keys used in the sessions. The superscripts of a nonce indicate the type of session to which it belongs and nonces without superscript belongs to the distinguished session. Applying this abstraction we obtain the twelve rules below which define the abtract model. They correspond to the instantiation of the generic rules with constants principals and nonces that characterize the four type of sessions we mentioned. This abstraction leaves us with a system where we still have to deal with messages of unbounded size and with arbitrary number of runs of the sessions 1, 2 and 3.

Generic session parameterized by (A,B,Na,Nb)	The abstract models consists in four specific instantiations of the generic session: the fixed session (H,H,N_1,N_2)	sessions of type 1 (H,H,N_1^{HH},N_2^{HH})	sessions of type 2 (H,I,N_1^{HI},N_2^{HI})	sessions of type 3 (I,H,N_1^{IH},N_2^{IH})
$\frac{-}{\{A,Na\}_{pbk(B)}};$	$\frac{-}{\{H,N_1\}_{pbk(H)}};$	$\frac{-}{\{H,N_1^{HH}\}_{pbk(H)}};$	$\frac{-}{\{H,N_1^{HI}\}_{pbk(I)}};$	$\frac{-}{\{I,N_1^{IH}\}_{pbk(H)}};$
$\frac{\{A,n_1\}_{pbk(B)}}{\{n_1,Nb,B\}_{pbk(A)}};$	$\frac{\{H,n_1\}_{pbk(H)}}{\{n_1,N_2,H\}_{pbk(H)}};$	$\frac{\{H,n_1\}_{pbk(H)}}{\{n_1,N_2^{HH},H\}_{pbk(H)}};$	$\frac{\{H,n_1\}_{pbk(I)}}{\{n_1,N_2^{HI},I\}_{pbk(H)}};$	$\frac{\{I,n_1\}_{pbk(H)}}{\{n_1,N_2^{IH},H\}_{pbk(I)}};$
$\frac{\{Na,n_2,B\}_{pbk(A)}}{\{n_2\}_{pbk(B)}};$	$\frac{\{N_1,n_2,H\}_{pbk(H)}}{\{n_2\}_{pbk(H)}};$	$\frac{\{N_1^{HH},n_2,H\}_{pbk(H)}}{\{n_2\}_{pbk(H)}};$	$\frac{\{N_1^{HI},n_2,I\}_{pbk(H)}}{\{n_2\}_{pbk(I)}};$	$\frac{\{N_1^{IH},n_2,H\}_{pbk(I)}}{\{n_2\}_{pbk(H)}};$

2.3 Verification step

Given the abstracted version of the protocol, HERMES computes sufficient conditions ensuring that messages exchanged cannot be used to reveal the secrets. More precisely, HERMES computes the set of so-called *hat messages*, that is, messages able to protect the secrets. Intuitively, such messages are encrypted terms of the form $\{X\}_K$ where the inverse key K^{-1} is not known by the intruder. The computation of hat messages is done iteratively, starting from the (abstract) transitions of the protocol T, the set of secrets S. An initial hypothesis about secret keys is used to initialize the set of hat messages H. Then, at each step, one transition $t \rightarrow t'$ is analyzed and, both the set of secrets S and H are updated.

The infinite set H of hat messages is represented by two sets (G, B) of terms. Roughly speaking, the pair (G, B) represents the set of messages that are instances of (good) terms in G but do not unify with (bad) terms of B.

We provide HERMES with a widening operator that ensure termination in the presence of sequences of rules that can loop and produce an infinite set of growing bad terms. An additional operator *Sup* is used to RESUME ? COVER ? ABSTRACT AWAY ? these terms with a finite term, $Sup(t)$, which is interpreted as set of all terms which contains t, see [4] for details.

At termination, HERMES yields an augmented set of secrets S' ($S \subseteq S'$) and either a representation of a set of hat messages H' ($H' \subseteq H$) that protect S' or an attack in case a secret is revealed.

Example We illustrate this technics on the abstract model of the Needham-Schroeder-Lowe Protocol. HERMES starts with the set of secrets $S = \{prk(H), N_1, N_2\}$, 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 secret and good patterns but an augmented set of bad patterns B' that consists in:

$$\{N_1^{HI}, x_s, I\}_{pbk(H)} \ ; \ \{N_1^{HH}, sup(x_s, I), H\}_{pbk(H)} \ ; \ \{x_s, I\}_{pbk(H)} \ ; \ \{N_1, sup(x_s, I), H\}_{pbk(H)}$$

The first one of these bad protections is obtained from the third rule $\frac{\{N_1^{HI}, x_s, I\}_{pbk(H)}}{\{x_s\}_{pbk(I)}}$ of session $(H, I, N_1^{HI}, N_2^{HI})$ 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 $\{N_1^{HI}, x_s, I\}_{pbk(H)}$. The other bad patterns are obtained in the same way by a backward analysis of the rules of the abstract model. HERMES stops when no more bad patterns or secret are generated and we conclude that the protocol ensures the secrecy proprety in all initial state where the secrets in S' are initially protected by hat messages $G \setminus B'$. For our running example, it is easy to prove that this condition holds for N_1 and N_2 – by freshness, they do not appear in any message before a session run. Moreover, the condition holds on all initial states where the secret key $prk(H)$ was never sent, or never sent without a protection in $G \setminus B'$.

In the original version of the protocol due to Needham-Schroeder, the identity of role `B` does not appear in message 2. On this version HERMES returns a new set of secrets that contains the message $\{H, N_1^{HI}\}_{pbk(H)}$. Obviously, this message cannot be secret: it does not contain any fresh nonce or initial secret, so it could be forged by the intruder. Starting from this message and applying the rules of the abstract model, the intruder can obtain the message $\{(N_a^{HI}, N_2)\}_{pbkH}$, and finally N_2. This attack corresponds to the well known attack of Needham-Schroeder discovered by Lowe [13].

3 Experimental results

HERMES is available online at the url [20]. Several examples are available in the web page and the others may be written directly in the input area using the LAEVA syntax. As output, HERMES provides the result sets S', G and B as well as the sequence of rules leading to each new secret or bad pattern. The following table summarize the results obtained by HERMES on protocols from [5] regarding secrecy properties. In the table, *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 in fact, is not valid on the concrete level.

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 obtainded by running HERMES on a Pentium III 600Mhz PC under Linux.

4 Conclusion and Future Work

In this paper, we presented a tool for the verification of secrecy of unbounded and bounded protocols. Several remarks can be mabe about the underlying method and its implementation. The first is that surprinsingly we have not encountered any false attack on any pratical protocol. Although, one could construct a protocol whose verification leads to a false attack. A second remark is that given a protocol and a secrecy property to verify, HERMES provides a tree that can be interpreted as a proof for the corectness of the protocol. We are currently working on extracting a proof in the automatic prover Coq from the tree provided by HERMES. A third remark, is that the underlying method can be extended to non-atomic keys and xor-encryption. This is a first step towards taking into account algebraic propeties of encryption algorithms.

References

1. Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In *International Conference on Concurrency Theory*, volume 1877 of *LNCS*, pages 380–394, 2000.
2. D. Bolignano. Integrating proof-based and model-checking techniques for the formal verification of cryptographic protocols. In *International Computer Aided Verification Conference*, volume 1427 of *LNCS*, pages 77–87, 1998.
3. L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. Technical report, EVA : http://www-eva.imag.fr/, 2002.
4. L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. In *TACAS'03*, volume To appear of *LNCS*, 2003.
5. J. Clark and J. Joacob. A survey on authentification protocol. Available at the url http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps, 1997.
6. H. Comon-Lundh and V. Cortier. Security properties: Two agents are sufficient. Technical report, LSV, 2002.
7. V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *IEEE Computer Security Foundations Workshop*, pages 97–110, 2001.
8. Grit Denker and Jon Millen. The CAPSL integrated protocol environment. Technical Report SRI-CSL-2000-02, Computer Science Laboratory, SRI International, 2000.
9. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols*, 1999.
10. T. Genet and F. Klay. Rewriting for cryptographic protocol verification. In *International Conference on Automated Deduction*, volume 1831 of *LNCS*, 2000.

11. Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *International Workshop on Formal Methods for Parallel Programming: Theory and Applications*, volume 1800 of *LNCS*, 2000.
12. Jean Goubault-Larrecq. Les syntaxes et la smantique du langage de spcification eva. Technical report, EVA : http://www-eva.imag.fr/, 2002.
13. G. Lowe. An attack on the Needham-Schroeder public-key authentification protocol. *Information Processing Letters*, 56(3):131–133, 1995.
14. G. Lowe. Casper: A compiler for the analysis of security protocols. In *10th IEEE Computer Security Foundations Workshop (CSFW '97)*, pages 18–30, Washington - Brussels - Tokyo, June 1997. IEEE.
15. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.
16. David Monniaux. Abstracting Cryptographic Protocols with Tree Automata. In *Static Analysis Symposium*, volume 1694 of *LNCS*, pages 149–163, 1999.
17. L. Paulson. Proving properties of security protocols by induction. In *IEEE Computer Security Foundations Workshop*, pages 70–83, 1997.
18. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *IEEE Computer Security Foundations Workshop*, 2001.
19. S. Schneider. Verifying authentication protocols with CSP. In *IEEE Computer Security Foundations Workshop*, pages 3–17, 1997.
20. `http://www-verimag.imag.fr/ Liana.Bozga/eva/hermes.php`.