PIL/SETHEO: A Tool for the Automatic Analysis of Authentication Protocols*

Johann Schumann

Institut für Informatik, TU München
schumann@in.tum.de

1 Introduction

Authentication protocols are used in distributed environments ensure the identity of the communication partners and to establish a secure communication between them. With the widespread use of distributed computing (e.g., Internet, electronic commerce), authentication protocols have gained much importance. Because high values can be at stake, such protocols must have extremely high quality and must be resistant with respect to intruders. Therefore, usually formal methods are used for their design and verification. In the literature, a variety of different methods and techniques for protocol analysis have been developed (cf. [Mea94] for an overview). Typically, the methods exhibit their strength in different stages of the development of an authentication protocol: in early design stages, conformance to a development standard [AG98] and absence of major deficiencies of a protocol can be ensured by type checking. As a next step, modal logics of belief are used to model a protocol and its properties. Such logics (e.g., BAN [BAN89], SVO, GNY, or AUTLOG [KW94]) are convenient for the verification of important properties, but are relatively weak with respect to modeling intricate intruder scenarios. Here, model-checking approaches (e.g., [KW96]) can be used. They can efficiently and automatically analyze a protocol. However, they usually cannot provide a positive proof and are limited by the size of the state-space they can explore. Methods for verification which are based on CSP, like [Pau97], avoid this problem by simultaneously modeling a potentially infinite number of interleaving protocol runs, but their degree of automatic processing (e.g., with Isabelle) is still rather small.

The tool PIL/SETHEO addresses the second stage: PIL/SETHEO is capable of automatically proving safety properties of authentication protocols, formalized in the modal belief logics BAN [BAN89] and AUTLOG [KW94]. PIL/SETHEO is based SETHEO, an automated theorem prover for first order predicate logic.

2 Requirements and System Architecture

PIL/SETHEO was designed with the goal of practical usability. Therefore, the following important requirements are the basis for PIL/SETHEO's system design:

* This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) within Habilitation-grant Schu908/5-1, Schu908/5-2, and SFB 342/A5.

N. Halbwachs and D. Peled (Eds.): CAV'99, LNCS 1633, pp. 500–503, 1999.

- *automatic processing*: after specifying the protocol and the desired properties the tool should run automatically. Response-times are to be kept below one minute.
- *representation level*: the protocol and its properties are specified in the modal BAN or AUTLOG logic. The transformation into first-order logic must be kept transparent to the user. Thus, no knowledge about first-order theorem proving or SETHEO should be required to use PIL/SETHEO.
- *human readable proofs*: a major benefit of protocol analysis with modal belief logics is that the resulting proofs are relatively short and provide valuable insights to the protocol designer. This is in sharp contrast to model checking techniques (where no proof is provided) and CSP-based techniques which produce rather lengthy and complex proofs. Hence, all proofs are to be presented on the level of the source logic (BAN or AUTLOG) and must be human-readable.
- *feedback on failed conjectures*: during development of a protocol, it is likely that some of the conjectures cannot be proven due to errors in design or formalization. Then, a simple answer "no" (or an endless loop) is rather insufficient. Thus PIL/SETHEO has to offer several ways to provide feedback on what might be wrong in case a proof attempt fails.

These requirements are reflected in PIL/SETHEO's system architecture. Its input is a specification of the protocol's messages, additional assumptions, and the theorems to be proven. The specification language developed for PIL/SETHEO [Wag97] is close to the underlying modal logic (BAN or AUTLOG). An example for a simple protocol (a variant of the RPC-handshake) is shown in Figure 1A.

This input specification in translated into one or more proof tasks in first-order logic (in clausal normal form). PIL/SETHEO uses the approach of meta-interpretation which transforms each BAN (or AUTLOG) formula into a term. A newly introduced predicate symbol `holds` (abbreviated as $\vdash$) is true, if and only if its argument (a translated modal formula) can be derived using the inference rules of the resp. modal logic. Thus, all inference rules of the BAN (or AUTLOG) logic are transformed into first-order implications. For details see [Sch97].

These proof tasks form the input of SETHEO, a high performance theorem prover for first-order logic in clausal normal form [Let92]. SETHEO features a wide variety of techniques for pruning the search space which is traversed in a depth-first manner with iterative deepening. When SETHEO finds a proof, a tree-like model elimination tableau is returned. A proof is this form, however, is not readable. Therefore, it is automatically translated into a human-readable form using the tool ILF-SETHEO [WS97]. After a transformation into a sequent-style calculus (block calculus), the proof is syntactically converted into a proof of the original BAN (or AUTLOG) logic and type-set using LaTeX. A short example of the output is shown in Figure 1B. This representation of the proof directly corresponds to the representation level of the input of PIL/SETHEO (left side of Figure). For details on the notation cf. [BAN89,Sch97].

In case, a conjecture cannot be proven, SETHEO usually reaches a run-time limit. In order to increase usability of the tool, PIL/SETHEO features

A

```
Objects:
  principal A,B;
  sharedkey K_a_b, Kp_a_b;
  statement N_a, N_b;
Assumptions:
  A believes sharedkey K_a_b;
  B believes sharedkey K_a_b;
  A believes B controls
       sharedkey K_a_b;
  B believes sharedkey Kp_a_b;
  A believes fresh N_a;
  B believes fresh N_b;
Idealized Protocol:
  message 1: A -> B {N_a}(K_a_b);
  message 2: A <- B
       {f(N_a),N_b}(K_a_b);
  message 3: A -> B {N_b}(K_a_b);
  message 4: A <- B
       {sharedkey Kp_a_b}(K_a_b);
Conjectures: after message 4:
  B believes A believes N_b;
```

B

Theorem 1. *conjecture.*

Proof. We show directly that

$$conjecture. \tag{1}$$

Because of *Message-Meaning*, $Assumption_2$, and by $Message_3$

$$\vdash B \models A \mid\sim N_B. \tag{2}$$

Because of *Theorem*

$$conjecture \Leftarrow \vdash B \models A \models N_B. \tag{3}$$

Because of *Nonce-Verification*: $\forall P, Q, \forall R : P \mid\equiv Q \mid\equiv R \Leftarrow P \mid\equiv Q \mid\sim R \wedge P \models \#R$. Hence by (2) and by $Assumption_6$ $\neg$ *conjecture*. Hence by (3) *conjecture*. Thus we have completed the proof of (1). q.e.d.

Fig. 1. Example input (A) and output of PIL/SETHEO (B).

two ways of producing feed-back: *belief-generation* and *abduction*. In the first case, PIL/SETHEO generates all beliefs which are derivable from the given specification and which conform to given syntactic criteria. Let us assume that we had "forgotten" the last assumption (`B believes fresh N_b`,$Assumption_6$) in Figure 1A. Then, our theorem cannot be proven. In that case, the user can ask PIL/SETHEO which kinds of BAN-formulas B believes. PIL/SETHEO, which uses a variant of the DELTA-preprocessor [Sch94] to generate the formulas in a bottom-up way, returns a list of BAN-formulas (in our example 124). PIL/SETHEO's user interface allows to further restrict the focus of the formulas by specifying a syntactic filter. For example, we might ask what B believes to be fresh (freshness is an important issue in protocol analysis with BAN-logic). Now, PIL/SETHEO returns a much shorter list of formulas (8 in our case). From them, it is quite obvious that there are no terms which contain any reference to freshness of time-stamp N_B. This is a clear indication that something is wrong with that time-stamp: B does not belief the validity of its own time-stamps. This immediately leads to the missing assumption $B \models \#N_B$ (`B believes fresh N_b`) which then yields the desired proof.

In the abductive mode, additional assumptions (or patterns, like B believes the freshness of each time-stamp) can be given by the user. PIL/SETHEO then tries to prove the theorem and returns a list of (most specific) instantiations of the additional assumptions which were required to find a proof with given resources. From there, the user can find out those assumptions which might be important for the analysis.

The user interface for PIL/SETHEO is straight forward and easy to use. PIL/SETHEO uses the tool "make" to make sure that for a complete analysis all conjectures have been proven. Upon completion, PIL/SETHEO returns a LaTeX-document containing a full report and all proofs.

3 Conclusions

We have used PIL/SETHEO to analyze a number of well-known protocols (Kerberos, Andrew Secure RPC Handshake, Needham Schroeder, Needham Schroeder with pubic keys, Otway Rees, wide-mouthed frog, Yahalom, CCITT-X.509, ISO10181 and others). All proof tasks arising from the verification of these protocols (with BAN or AUTLOG) could be shown fully automatically within less than one minute per protocol (actual proof times have been below 20 seconds). As far as possible with the formalism of belief logics, we were able to "re-detect" errors in early versions of the protocols. With its fully automatic operation and its capability to generate human-readable proofs in the BAN or AUTLOG logic PIL/SETHEO is a powerful, yet easy to use tool, especially suited for early protocol design phases.

References

[AG98] M. Abadi and D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 1998.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. In *ACM Operating Systems Review 23(5)*, 1989.

[KW94] V. Kessler and G. Wedel. AUTLOG — An Advanced Logic of Authentication. In *Proc. IEEE Computer Security Foundations Workshop IV*, pages 90–99. IEEE, 1994.

[KW96] D. Kindred and J. Wing. Fast, automatic checking of security protocols. In *2nd USENIX Workshop on Electronic Commerce*, pages 41–52, 1996.

[Let92] R. Letz, et al. SETHEO: A High-Performance Theorem Prover. *JAR*, 8(2):183–212, 1992.

[Mea94] C. A. Meadows. Formal verification of Cryptographic Protocols: A Survey. In *Proc. AsiaCrypt*, 1994.

[Pau97] L. Paulson. Proving properties of security protocols by induction. In *PCSFW: Proc. 10th Computer Security Foundations Workshop*. IEEE, 1997.

[Sch94] J. Schumann. DELTA — A Bottom-up Preprocessor for Top-Down Theorem Provers, System Abstract. In *Proc. CADE 12*. Springer, 1994.

[Sch97] J. Schumann. Automatic verification of cryptographic protocols with SETHEO. In *Proc. CADE 14*. Springer, pp. 87–100, 1997.

[Wag97] K. Wagner. PIL: Ein SETHEO-basiertes Werkzeug zur Analyse kryptographischer Protokolle. Fortgeschrittenenpraktikum, TUM, 1997.

[WS97] A. Wolf and J. Schumann. ILF-SETHEO: Processing Model Elimination Proofs for Natural Language Output. In *Proc. CADE 14*. Springer, 1997.