Automatic Verification of Cryptographic Protocols with SETHEO

Johann Schumann

Institut für Informatik
Technische Universität München
D-80290 München, Germany
schumann@informatik.tu-muenchen.de

Abstract. In this paper, we describe, how the automated theorem prover SETHEO is used for automatic verification of safety properties of cryptographic protocols. The protocols and their properties are specified using the so-called BAN logic, a multi-sorted modal logic capable of expressing beliefs about secure communication. The resulting formulas and inference rules are transformed into first order predicate logic and processed by the prover SETHEO. Proofs found by SETHEO are then automatically converted into a human-readable form. Experiments with several well-known protocols (Kerberos, Secure RPC handshake, and CCITT509) revealed very good results: the required properties of the protocols (as described in the literature) could be shown automatically within a few seconds of run-time.

1 Introduction

Cryptographic protocols are used when secure services are to be provided between two or more partners. Typical examples are banking applications or exchange of user information e.g., during a login procedure.

Due to the dramatic increase in network applications (e.g., WWW, Java) during the last few years, cryptographic protocols have gained more and more importance. Safety and security of communication has thus become very important issues (cf. [17]). Providing secure services is the key feature of cryptographic protocols. If, however, a protocol is designed not flawless, the protocol may fail to do so. Intruders then can supply false messages to the protocol, causing severe security problems.

Such security flaws are in general very hard to detect. Therefore, many methods and tools to *analyze* the behaviour of a cryptographic protocol have been studied and developed (for an overview see e.g. [6, 15]). Methods for verification (based on logic) have come up since about 1989. The most well-known approach in that area has been presented in [3, 4]. The authors have proposed a logic of believe (hence called BAN logic after the initials of the authors), which is capable of expressing the behaviour of a protocol and with which safety and security properties can be specificed. With the help of specific inference rules, the validity of these requirements can be shown.

Since then, many approaches, based on that work, have been made. Several problems with the BAN logic (e.g., with respect to modelling intruders), and a missing denotational semantics have led to a number of extensions, e.g. GNY [8], SVO [19], AT [1], or AUTLOG [10]. In these approaches, a slightly extended logic with additional (or modified) inference rules is defined.

Reasoning with the BAN logic (or its extensions) has been done manually in most cases. Only quite recently, tools have been developed to perform such proofs in a computer-supported way either interactively (e.g.,[5]), or automatic (e.g. [2] or [10]). With most examples mentioned in the literature, the run-times for finding the proofs are considerable.

We apply the automated theorem prover SETHEO to *automatically* verify theorems formulated in the BAN logic. Although one of the more advanced logics (e.g. GNY) could have been used, the work reported in this paper has been made to demonstrate SETHEO's ability to efficiently cope with proof tasks arising in the area of analyzing cryptographic protocols. This also means that with our application we can only prove properties which can be shown within the original BAN logic. No attempts have been made to correct any deficiencies of that logic.

Three approaches to support reasoning in the BAN logic or extensions thereof have been reported in the literature. [10] has extended the logic for better ways of dealing with intruders. The authors have implemented a PROLOG program which can (in a forward-chaining reasoning process) automatically show many properties. With this approach, however, the size of the resulting search space is huge[1]. [2], on the other hand, transforms a specification of a cryptographic protocol given in the specification language ICL, into a formula of the GNY logic and uses the higher-order theorem prover HOL [9] to perform automatic reasoning. The approach described in [5] is based on the BAN logic. The authors transform BAN formulas into first-order formulas (in a similar way as in our approach) and use the EVES system to interactively perform reasoning. In order to increase the level of automatic processing, they use a forward-chaining way of reasoning. Our approach, on the other hand performs automatic reasoning in the backward-chaining, goal-oriented way of the prover SETHEO.

The paper proceeds as follows: first, we give a short introduction into the BAN logic, its notation and inference rules. The main section of this paper focuses the transformation necessary to convert an (idealized) protocol, the safety requirements and the inference rules into First Order Predicate Logic, the input language of SETHEO. For this paper, we assume the reader to be familiar with the basic notions of the Model Elimination Calculus [13] and SETHEO. For details on SETHEO see e.g., [12, 11, 7, 16].

Then, we present the results of experiments with some well-known protocols and show an automatically generated proof. We conclude by summarizing our approach, discussing problems of this approach, and presenting ongoing and future work.

[1] "...and it takes too long if there are too many rules." [10],pg. 8.

2 The BAN logic

The goal of an authentication protocol is that, after a protocol run has taken place, *"the principals should be entitled to believe that they are communicating with each other, and not with intruders."*[2] The principals can be persons, other computers, or even services.

The BAN logic ("Burrows, Abadi, Needham" [3]) has been developed to formalize such beliefs and to allow to draw inferences within that logic. The BAN logic is a multi-sorted modal logic. The basic sorts of objects are: *principals*, *encryption keys*, and *formulas*.

In the following, we give a short informal introduction into the basics of that logic. However, we only describe the syntax and inference rules of BAN and do not focus on the (informal or formal) semantics. Furthermore, we adhere to the notations, found in [3]. Principals are denoted by A, B, P, S and pA, pB, pS, formulas as X, Y, encryption keys (shared between two principals) are written as K_{ab}, K_{as}, K_{ba}. Public keys are K_a, K_b, K_c, whereas $K_a^{-1}, \ldots$ denote the corresponding secret keys. Nounces (e.g., time-stamps) are T, T_a, T_b.

Furthermore, the BAN logic contains a set of operators which may be connected via "$\wedge$" to construct new formulas. A complete list of operators (and its representation for SETHEO – see Section 3) is shown in Table 1. A detailed description of them can be found in [3] or [4]. As an example, two operators are explained: P **believes** X (or $P|\equiv X$) means that principal P believes a statement X. This also means that P acts as if X was true. $P \overset{K}{\leftrightarrow} Q$ says that principals P and Q communicate via a shared key K.

Notation in [BAN90]	Shorthand	SETHEO	[BAN90]	SETHEO
P **believes** X	$P\|\equiv X$	`bel(P,X)`	A, B, S, P	`a,b,s,p`
P **sees** X	$P \triangleleft X$	`sees(P,X)`	K	`k`
P **said** X	$P\|\sim X$	`said(P,X)`	K_{ab}	`kab`
P **controls** X	$P\|\Rightarrow X$	`controls(P,X)`	K_{ba}	`kba`
fresh(X)	$\#X$	`fresh(X)`	K_{as}	`kas`
K^{-1}	-	`inv(K)`	K_{bs}	`kbs`
$P \overset{K}{\leftrightarrow} Q$	-	`sk(P,Q,K)`	$\{M_1, \ldots, M_n\}$	`cons(m1,cons(...`
$\{X\}_K$	-	`encr(X,K)`		`cons(mn,nil)...)`
$\overset{K}{\mapsto} Q$	-	`pub(Q,K)`	$\{\}$	`nil`
$P \overset{K}{=} Q$	-	`ss(P,Q,K)`		
$\langle X \rangle_Y$	-	`comb(X,Y)`		

Table 1. Notation of BAN operators as used in [3], a common short-hand, and the operators in SETHEO's input language and SETHEO

[2] [3], pg. 1.

In [3], a list of logical postulates (or inference rules) is given which define the BAN logic. The most important inference rules are described in the following:

Message-meaning The message-meaning rule concerns the interpretation of a message. As soon as a principal P believes that it communicates with some other principal Q using encrypted messages (with shared keys, public keys, or shared secret keys), and such a message arrived, we may infer that P believes that Q has actually sent the message X. In a formal notation we get (for shared keys):[3]

$$\frac{P \models P \overset{K}{\leftrightarrow} Q \qquad P \triangleleft \{X\}_K}{P \models Q \mid\sim X}$$

Nonce-verification Nonces are specific messages generated by a principal. Their main purpose is to guarantee the freshness of a message. Therefore, often time-stamps are used as nonces. The nonce-verification rule states that a message is "recent", i.e., that the sender still believes in it when the corresponding nonce is fresh:

$$\frac{P \models \#X \qquad P \models Q \mid\sim X}{P \models Q \models X}$$

Jurisdiction This rule states that if P believes that Q has jurisdiction (control) over X, then P trusts Q on the truth of X[4]:

$$\frac{P \models Q \mapsto X \qquad P \models Q \models X}{P \models X}$$

Components Several inference rules of the BAN logic concern the "packing" and "unpacking" of messages which consist of more than one atomic message. Furthermore, rules exist that a principal can decipher messages, for which the appropriate keys are known. For a list of these inference rules see [3].

Symmetry of communication Two inference rules (which are not defined in [3], but used there) define the symmetry of communication via shared keys, and via shared secrets.

$$\frac{P \models Q \overset{K}{\leftrightarrow} R}{P \models R \overset{K}{\leftrightarrow} Q} \qquad \frac{P \models Q \overset{K}{\rightleftharpoons} R}{P \models R \overset{K}{\rightleftharpoons} Q}$$

In general, authentication protocols are specified by a finite sequence of messages which are being transmitted between the principals. Often, such a message (principal P sends a message to Q) is written as:

$$P \rightarrow Q : \text{message}$$

[3] For public keys and shared secrets, we obtain similar rules.

[4] [3], pg. 3.

Within the BAN logic, each message of the protocol is given in an *idealized* form. In such an idealized protocol, uncrypted messages are not included. This means that all messages are of the form $\{X\}_K$ or sequences thereof. As an example, consider the message $A \rightarrow B : \{A, K_{ab}\}_{K_{bs}}$. This message means that A sends its own identification plus the shared key K_{ab}, encrypted with the key K_{bs} to principal B. It is idealized as

$$A \rightarrow B : \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$$

For a detailed description of the process of idealizing a protocol see [3]. If a message $P \rightarrow Q : X$ has been received, the formula $Q \triangleleft X$ (Q sees X) holds. A list of such formulas (one for each message) comprises the input specification of a protocol in BAN.

The theorems we want to prove in general depend on the protocol we are analyzing. However, there are default requirements for the services an authentication protocol should provide. These are of the form that the principals (A and B) believe that they properly communicate via a shared key K: $A \models A \overset{K}{\leftrightarrow} B$ and $B \models A \overset{K}{\leftrightarrow} B$, and the principals believe the believes of their respective partners: $B \models A \models A \overset{K}{\leftrightarrow} B$, $A \models B \models A \overset{K}{\leftrightarrow} B$.

Example: The Kerberos Protocol. Before we focus on transforming formulas of the BAN logic into First Order Predicate logic, we present the formalisation and analysis of a well-known protocol, the Kerberos protocol. This protocol has also been analysed with our SETHEO-based system; the results are shown in Section 4 and a machine-generated example-proof is given in the Appendix.

The Kerberos protocol establishes a shared key between two principals pA, pB with the help of an authentication server pS. This protocol has been developed at MIT within the Athena project. The Kerberos protocol consists of 4 messages, the last three of which are being idealized for the analysis. These messages are shown in Table 2. With the first message, pA contacts the server pS, sending its own name and that of the proposed communication partner pB. pS provides pA in the second message with the session key K_{ab}, and a certificate conveying the session key and pA's identity (encrypted by the key K_{bs}). The entire message is encrypted by pA's key K_{as}. Then, the latter part of the message is sent from pA to pB, along with a time-stamp (nounce T_a), issued by pA. Finally, pB acknowledges the establishment of secure communication by incrementing the time-stamp T_a and sending it back to pA (encrypted now by the mutual session key K_{ab}). After that step, a secure communication with the session-key K_{ab} is established.

For the analysis of this protocol, we must give a set of additional *assumptions*: statements about shared keys, jurisdiction, and the freshness of the nonces. These assumptions (from [3]) are shown in Table 3[5].

[5] The experiments with SETHEO revealed, however, that one assumption essential for the proofs, $pA \models \#T_a$ (marked by a †) is not described in the literature. This

#	Kerberos protocol	idealized Kerberos protocol
1	$pA \rightarrow pS : pA, pB$	—
2	$pS \rightarrow pA : \{T_s, L, K_{ab}, pB, \{T_s, L, K_{ab}, pA\}_{K_{bs}}\}_{K_{as}}$	$pS \rightarrow pA : \{T_s, pA \overset{K_{ab}}{\leftrightarrow} pB, \{T_s, pA \overset{K_{ab}}{\leftrightarrow} pB\}_{K_{bs}}\}_{K_{as}}$
3	$pA \rightarrow pB : \{T_s, L, K_{ab}, pA\}_{K_{bs}}, \{pA, T_a\}_{K_{ab}}$	$pA \rightarrow pB : \{T_s, pA \overset{K_{ab}}{\leftrightarrow} pB\}_{K_{bs}}, \{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}_{K_{ab}}$
4	$pB \rightarrow pA : \{T_a + 1\}_{K_{ab}}$	$pB \rightarrow pA : \{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}_{K_{ab}}$

Table 2. The Kerberos Protocol: messages in the original and idealized format

$pA \models pA \overset{K_{as}}{\leftrightarrow} pS$	$pA \models (pS \models\!\!\rightarrow pA \overset{K}{\leftrightarrow} pB)$
$pB \models pB \overset{K_{bs}}{\leftrightarrow} pS$	$pB \models (pS \models\!\!\rightarrow pA \overset{K}{\leftrightarrow} pB)$
$pS \models pA \overset{K_{as}}{\leftrightarrow} pS$	$pA \models \#T_s$
$pS \models pB \overset{K_{bs}}{\leftrightarrow} pS$	$pB \models \#T_s$
$pS \models pA \overset{K_{ab}}{\leftrightarrow} pB$	$pB \models \#T_a$
$pA \models \#T_a$ (†)	

Table 3. The Kerberos Protocol: assumptions for the analysis

The security properties which have to be shown, and which will comprise our proof tasks 1–4 for the SETHEO experiments (see Section 4) are:

$$\begin{aligned}
&\text{Kerb1:} && pA \models pA \overset{K_{ab}}{\leftrightarrow} pB \\
&\text{Kerb2:} && pB \models pA \overset{K_{ab}}{\leftrightarrow} pB \\
&\text{Kerb3:} && pA \models pB \models pA \overset{K_{ab}}{\leftrightarrow} pB \\
&\text{Kerb4:} && pB \models pA \models pA \overset{K_{ab}}{\leftrightarrow} pB
\end{aligned}$$

As an example for a proof in the BAN logic, we take proof task "Kerb3"[6]: The theorem, we want to show is that $pA \models pB \models pA \overset{K_{ab}}{\leftrightarrow} pB$ holds, after messages 1–3 have arrived. Before message 3 has arrived, we already know from proof task "Kerb2" that

$$pB \models pA \overset{K_{ab}}{\leftrightarrow} pB. \tag{1}$$

fact has also been detected (independently) in [5]. With this additional assumption, proof task "Kerb4" can be shown.

[6] For comparison, we present the machine-generated proof of this proof task in the Appendix. For further remarks see Section 4.

By the inference rule "message-meaning" and with message 3 (second part) of the idealized protocol, we obtain

$$pB \models pA \mid\sim \{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}_{K_{ab}}. \tag{2}$$

Since, by assumption $pB \models \#T_a$, we have (if a part of a message is believed to be fresh, then the entire message is)

$$pB \models \#(\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}_{K_{ab}}). \tag{3}$$

Finally, by (2) and (3) and "nonce-verification", we can prove our theorem. q.e.d.

3 Transformation for SETHEO

The BAN logic is defined by giving an alphabet of atoms (principals and keys), functions and operators, and inference rules. One way to transform such a logic into first-order predicate logic is "simulation": On a meta-level, all formulas of BAN are represented as first-order *terms*, and the inference rules are specified as first-order formulas. Such an approach has been successfully used e.g., for preparing single-axiom propositional calculi for automated theorem proving [14].

Applying this approach to our BAN logic we obtain the following transformation:

A unary FOL predicate $holds(X)$ is defined which has the meaning: $holds(X)$[7] is true, if and only if the BAN-formula X is *derivable* from other BAN-formulas using the BAN inference rules.

For example, a BAN-formula (assumption) $A \models \#T_s$ (A believes the freshness of the nonce T_s) is transformed into:

$$holds(A \models \#T_s)$$

Additionally, each BAN inference rule is transformed into a FOL formula. Given an inference rule of the form

$$\frac{\mathcal{F}_1 \ldots \mathcal{F}_n}{\mathcal{G}}$$

we obtain the following formula:

$$\forall P, Q, K, \ldots : \; holds(\mathcal{F}_1) \wedge \ldots \wedge holds(\mathcal{F}_n) \rightarrow holds(\mathcal{G})$$

Variables (e.g., P, Q, K) which occur in the inference rules become all-quantified in the transformation. E.g., the nonce-verification rule from Section 2 is written as:

$$\forall P, Q, X : holds(P \models \#X) \wedge holds(P \models Q \mid\sim X) \rightarrow holds(P \models Q \models X)$$

[7] In shorthand, we write $\vdash X$.

Then, in our approach, the entire input formula for SETHEO is of the form

$$\mathcal{A}_1 \wedge \ldots \wedge \mathcal{A}_n \wedge \mathcal{M}_1 \wedge \ldots \wedge \mathcal{M}_m \wedge \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_k \rightarrow \mathcal{T}$$

where the $\mathcal{A}_i$ are the transformed assumptions, the $\mathcal{R}_i$ the transformed inference rules, $\mathcal{M}_i$ are the transformed messages of the idealized protocol, and $\mathcal{T}$ is the theorem to be shown. A formula of this structure can be easily transformed into Clausal Normal form (by Skolemization) and then fed into SETHEO.

However, several additional items are still to be done: performing a *syntactic transformation*, since SETHEO does not accept special symbols or infix operators in its input, handling of finite *sequences* of atomic messages which comprise a compound message, and defining *auxiliary predicates* to implement the above additions.

Syntactic Transformation. The conversion of the symbols and operators is straight-forward and shown in Table 1. Since SETHEO's input language (LOP) is similar to PROLOG, all constant and function symbols start with a lower-case letter, all variables start with an upper-case character. An (abbreviated) example of such a LOP formula (from the Kerberos protocol in Table 2 and Table 3) is shown in the following Table 4.

```
/*************************************************************/
/*    NAME:            kerb-1.lop                            */
/*    SCCS:            1.1.7   3/12/1996                     */
/*************************************************************/
#clausename assumption1
holds(bel(a,sk(a,s,kas)))<-.
#clausename assumption2
holds(bel(b,sk(b,s,kbs)))<-.
...
#clausename message2
holds(sees(a,encr(cons(ts,cons(sk(a,b,kab),
                 cons(encr(cons(ts,cons(sk(a,b,kab),nil)),
                 kbs),nil))),kas)))<-.
...
#clausename message_meaning
holds(bel(P,said(Q,X))) <-
    holds(bel(P,sk(Q,P,K))), holds(sees(P,encr(X,K))).
...
#clausename theorem
<-holds(bel(a,sk(a,b,kab))).
```

Table 4. BAN-formula transformed into SETHEO-notation (An example from the Kerberos protocol of Section 2)

Sequences. Within the BAN logic, messages can consist of a single atomic

message, or a (finite) sequence of messages enclosed in braces {}. A notation of such sequences of variable length as well as predicates for accessing, breaking-up and constructing such sequences have to be defined in first-order logic. For this prototypical transformation of BAN into FOL, we have defined an additional function symbol *cons* with arity two, and a symbol *nil*, representing the empty sequence. Then, a message $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ is represented as the term $cons(m_1, cons(m_2, \ldots cons(m_n, nil)\ldots)$.

Auxiliary predicates to select *one* message out of such a combined message $\mathcal{M}$ ("$\iota(\mathcal{M})$") and to select a subsequence $\mathcal{S} = \{m_i | 1 \leq i \leq n\}$ ("$\mathcal{S} \sqsubseteq \mathcal{M}$") have been defined as follows:

$$
\begin{array}{l}
nil \sqsubseteq nil \wedge \\
\forall F, T, R : T \sqsubseteq R \rightarrow cons(F, T) \sqsubseteq cons(F, R) \wedge \\
\forall F, T, R : T \sqsubseteq R \rightarrow T \sqsubseteq cons(F, R)
\end{array}
$$

$$
\begin{array}{l}
\forall X, Y : X = \iota(cons(X, Y)) \wedge \\
\forall X, Y, Z : X = \iota(Z) \rightarrow X = \iota(cons(Y, Z))
\end{array}
$$

4 Experiments and Results

Using the transformation described above, we made experiments with several well-known protocols, namely the Kerberos protocol, the Secure RPC-handshake, and the CCITT-509 protocol. For each of the experiments, the idealized protocols, assumptions and the theorems to be proven have been directly taken from the literature.

Each proof, found by SETHEO can be converted fully automatic into a human-readable form, using the tool ILF-SETHEO [21]. Due to space restriction, we only present one proof, namely that of proof task "Kerb3" of the Kerberos protocol[8]. The manual proof of this theorem has be given above in Section 2. It can be seen clearly that the structure of SETHEO's proof closely resembles that, found in the literature. Due to the auxiliary predicates which have been introduced during the transformation, however, the proof gets much longer and many technical details (e.g., about submessages) are shown. Future versions of our system will attempt to hide these technical details (see also the Conclusions).

All four proof tasks concerning the Kerberos protocol could be shown automatically, using SETHEO. The results are shown in Table 5. All run-times have been obtained on a SUN sparc 10 workstation and are given in seconds (the times are measures with a granularity of $1/60s$). In all cases, SETHEO was started with its default parameters. "msgs" indicates which messages M_i of the idealized protocol (of the form $P \triangleleft M_i$) have been considered. We also give the number of clauses of the input formula, the number of Model Elimination inference i steps for the proof, and the depth in which a proof could be found.

[8] This piece of LaTeX-code has been included exactly as it was produced by ILF-SETHEO [21]. No manual changes (except one line break) had to be applied.

The run-time consists of the time needed to compile the formula T_c, and the time for the search T_{SAM}. $T_{tot} = T_c + T_{\mathrm{SAM}}$ comprises the total run-time needed to solve the proof task.

Proof task Kerb3-1 and Kerb4-1 have been generated by *not* adding the theorems of proof task 1 and 2 as lemmas to the formula. This resulted in much longer and deeper proofs. A fifth proof task was constructed according to a remark in [3]: Using the first three messages of the protocol only, it is not able to show $pA \models pB \models pA \overset{K_{ab}}{\leftrightarrow} pB$. As expected, SETHEO cannot find a proof for that task. Unfortunatedly, SETHEO does not stop saying "no proof", but attempts to search for the proof arbitrarily long. This general problem in our approach of using SETHEO will be discussed in the Conclusions.

Task	Theorem	msgs	cl	i	d	T_c[s]	T_{SAM}[s]	T_{tot}[s]
Kerb1	$pA \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2	34	16	7	0.18	0.34	0.52
Kerb2	$pB \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2,3	35	17	7	0.21	0.36	0.57
Kerb3	$pA \models pB \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2,3,4	37	15	7	0.18	0.59	0.77
Kerb3-1	$pA \models pB \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2,3,4	35	31	11	0.18	81.94	82.14
Kerb4	$pB \models pA \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2,3,4	40	13	6	0.20	0.29	0.49
Kerb4-1	$pB \models pA \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2,3,4	37	28	12	0.20	592.55	592.75
Kerb5	$pA \models pB \models pA \overset{K_{ab}}{\leftrightarrow} pB$	2,3	39	–	–	0.22	–	–
RPC1	$pB \models pA \overset{K'_{ab}}{\leftrightarrow} pB$	1–4	34	1	2	0.45	0.13	0.58
RPC2	$pA \models pB \mathrel{\sim\!\!\!\vdash} (pA \overset{K'_{ab}}{\leftrightarrow} pB, N'_b)$	1–4	34	4	3	0.48	0.16	0.64
RPC2A	$pB \triangleleft N_a$	1–4	34	3	2	0.42	0.15	0.57
RPC3	$pB \models pA \models N_b$	1–4	34	5	3	0.45	0.15	0.60
RPC4	$pA \models pB \models (N_a, N_b)$	1–4	34	8	4	0.47	0.17	0.64
CCITT1	$pA \models pB \models X_b$	1–3	34	17	8	0.31	1.88	2.19
CCITT2	$pB \models pA \models X_a$	1–3	34	14	7	0.33	0.67	1.00

Table 5. Experimental results for the Kerberos protocol, the RPC handshake, and the CCITT X.509 protocol.

Similar experiments have also been carried out with the two other protocols, the Andrew Secure RPC-handshake, and the CCITT X.509 protocol. The Andrew Secure RPC-handshake consists of 4 messages and the CCITT X.509 protocol uses only three messages to establish a secure communication. For details on these protocols see e.g., [3, 4]. Here again, SETHEO could show all required proof tasks within a few seconds of run-time. The results are also shown in Table 5.

5 Conclusions

In this paper, we have described, how SETHEO can be applied to verify security properties of cryptographic protocols, using the BAN logic. We presented experimental results on several well-known cryptographic protocols. Looking at the manual proofs in the literature [3, 4, 5], we could verify our results.

The proof tasks could be solved fully automatic within very short run-times (up to a few seconds), making our approach feasible for real-world applications. As a further advantage, the proofs found by SETHEO can be converted into easy-to-understand natural-language proofs.

The experiments, described in this paper, however, comprise only a first case-study in this area. Several problems are yet to be solved and many wishes are still open.

Probably the most severe drawback of our approach is the handling of non-theorems: if one wants to prove a conjecture which is not true (see e.g., proof-task 5 of the Kerberos protocol), SETHEO searches forever instead of terminating with the message "Theorem not valid". Here, additional techniques (e.g., Model Checking or disproving techniques) would be helpful to check, if a theorem is not valid, and to give hints why a proof cannot be found. Experiments with such approaches are planned for the future.

The model of inferencing in the BAN logic has no clear semantics. Our approach just models the operational semantics of the BAN logic. Extensions of the BAN logic (e.g., [8, 19, 10]) have been developed to overcome this and other problems. Future work will attempt to use one of these extended logics for our approach.

Furthermore, the practical usability of our system can and must be increased drastically. This e.g. means that SETHEO should be integrated into a user-friendly system for development and verification of cryptographic protocols. Such a tool which currently is under development [20] contains a specification language for protocols. From this specification and the user-defined assumptions, all proof tasks can be generated automatically and then fed into SETHEO. Future versions will incorporate a graphics-based simulator for protocol runs, and tool-support for modeling many different kinds of intruders.

On the side of the automated prover SETHEO, using a combination of (default) top-down, backward chaining search with bottom-up processing would be of interest. Besides the possibility to reduce the search-space for complicated proof tasks, the bottom-up preprocessor DELTA [18] would allow to draw conclusions on the kinds of beliefs which can be deduced. This information can also be helpful in detecting possible "holes" for intruders.

A second extension to our approach which will be considered in the future is the transformation of the BAN logic (or extensions thereof) into *sorted* first-order predicate logic. The sort structure in the BAN logic is rather simple and thus allows a very efficient handling within SETHEO. One of the major benefits will be the automatic checking of the correct sorts in the input formulas and during reasoning, thus eliminating many possible flaws during a verification process.

References

1. M. Abadi and M. R. Tuttle. A Semantics for a Logic of Authentication. In *Proc. of the Tenth Annual ACM Symp. on Principles of Distributed Computing*, pages 201–216. ACM press, 1991.
2. S. H. Brackin. A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols. In *Proc. IEEE Computer Security Foundations Workshop IX*. IEEE, 1996.
3. M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. In *ACM Operating Systems Review 23(5) / Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*, Dec 1989.
4. M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb. 1990.
5. D. Craigen and M. Saaltink. Using EVES to Analyze Authentication Protocols. Technical Report TR-96-5508-05, ORA Canada, March 1996.
6. J. Geiger. Formale Methoden zur Verifikation kryptographischer Protokolle. Fortgeschrittenenpraktikum, Institut für Informatik, Technische Universität München, 1995. in German.
7. Chr. Goller, R. Letz, K. Mayr, and J. Schumann. SETHEO V3.2: Recent Developments (System Abstract) . In *Proc. CADE 12*, pages 778–782, June 1994.
8. L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proc. of IEEE Symposium on Security and Privacy, Oakland, Ca.,USA*, pages 234–248. IEEE, 1990.
9. M. J. C. Gordon. HOL: A proof generating system for higher-order logic. In G. Birtwistle and P. A. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis*, pages 73–128. Kluwer, 1988.
10. V. Kessler and G. Wedel. AUTLOG — An Advanced Logic of Authentication. In *Proc. IEEE Computer Security Foundations Workshop IV*, pages 90–99. IEEE, 1994.
11. R. Letz, K. Mayr, and C. Goller. Controlled Integration of the Cut Rule into Connection Tableau Calculi. *Journal Automated Reasoning*, (13):297–337, 1994.
12. R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.
13. D. W. Loveland. *Automated Theorem Proving: a Logical Basis*. North–Holland, 1978.
14. W. W. McCune and L. Wos. Experiments in Automated Deduction with Condensed Detachment. Technical report, Argonne National Laboratory, 1991.
15. C. A. Meadows. Formal verification of Cryptographic Protocols: A Survey. In *Proc. AsiaCrypt*, 1994.
16. Max Moser, Ortrun Ibens, Reinhold Letz, Joachim Steinbach, Christoph Goller, Johann Schumann, and Klaus Mayr. The Model Elimination Provers SETHEO and E-SETHEO. *special issue of the Journal of Automated Reasoning*, 1997. (to appear).
17. P. G. Neumann. *Computer Related Risks*. ACM Press, 1995.
18. J. Schumann. DELTA — A Bottom-up Preprocessor for Top-Down Theorem Provers, System Abstract. In *CADE 12*, 1994.
19. P. F. Syverson and P. van Oorschot. On Unifying Some Cryptographic Protocol Logics. In *Proc. of the IEEE Comp. Soc. Sympos. on Research in Security and Privacy*, pages 14–28, 1994.

20. Klaus Wagner. SIL: Ein SETHEO-basiertes Werkzeug zur Analyse kryptographischer Protokolle. Fortgeschrittenenpraktikum, Technische Universität München, 1997. (in preparation).
21. Andreas Wolf and Johann Schumann. ILF-SETHEO: Processing Model Elimination Proofs for Natural Language Output (System Description). In *CADE 14*, 1997. (submitted).

Kerberos Protocol: Proof Task 3

Theorem 1 *query.*

Proof[9]. We show directly that[10]

$$query. \tag{4}$$

Because of *message_3*

$$\vdash\ pB \triangleleft\ (\{\{\{T_s, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{bs}}, \{\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{ab}}\}). \tag{5}$$

Because of *theorem*

$$query :- \vdash\ pB\models\ pA\models\ pA \overset{K_{ab}}{\leftrightarrow} pB. \tag{6}$$

Because of *make_singleton*

$$\vdash\ A\models\ B\models\ C :- \vdash\ A\models\ B\models\ (C). \tag{7}$$

Because of *break_up_believed_messages*

$$\vdash\ A\models\ B\models\ C :- \vdash\ A\models\ B\models\ D \wedge C \sqsubseteq D. \tag{8}$$

Because of *submessage*

$$A \sqsubseteq\ (\{B, C\}) :- A \sqsubseteq\ C. \tag{9}$$

Because of *submessage* $(\{A, B\}) \sqsubseteq\ (\{A, C\}) :- B \sqsubseteq\ C$. Because of *submessage* $\langle\rangle \sqsubseteq\ \langle\rangle$. Therefore $(pA \overset{K_{ab}}{\leftrightarrow} pB) \sqsubseteq\ (pA \overset{K_{ab}}{\leftrightarrow} pB)$. Hence by (9)

$$(pA \overset{K_{ab}}{\leftrightarrow} pB) \sqsubseteq\ (\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}). \tag{10}$$

Because of *nonce_verification*

$$\vdash\ A\models\ B\models\ C :- \vdash\ A\models\ B\mid\sim\ C \wedge \vdash\ A\models\ \#C. \tag{11}$$

Because of *freshness*

$$\vdash\ A\models\ \#B :- \vdash\ A\models\ \#C\ \wedge\ C = \iota(B). \tag{12}$$

Because of $oneof\ T_a = \iota(\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\})$. Because of *assumption_1* $\vdash\ pB\models\ \#T_a$. Therefore by (12)

$$\vdash\ pB\models\ \#(\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}). \tag{13}$$

[9] by Setheo

[10] Due to technical reasons, our prototype always starts the proof with the artificial symbol *query*. The real theorem then is shown in line (6).

Because of *message_meaning*

$$\vdash A \models B \mid\sim C :- \vdash A \triangleleft \{C\}_D \wedge \vdash A \models B \overset{D}{\leftrightarrow} A. \tag{14}$$

Because of *lemma_from_task_2*

$$\vdash pB \models pA \overset{K_{ab}}{\leftrightarrow} pB. \tag{15}$$

Because of *sees_components*

$$\vdash A \triangleleft B :- \vdash A \triangleleft C \wedge B = \iota(C). \tag{16}$$

Because of $oneof\ A = \iota(\{B,C\}) : -A = \iota(C)$. Because of $oneof\ \{\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{ab}} = \iota(\{\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{ab}})$. Therefore $\{\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{ab}} = \iota(\{\{\{T_s, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{bs}}, \{\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{ab}}\})$. Hence by (16) and by (5) $\vdash pB \triangleleft \{\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\}\}_{K_{ab}}$. Hence by (14) and by (15) $\vdash pB \models pA \mid\sim (\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\})$. Hence by (11) and by (13) $\vdash pB \mid\equiv pA \models (\{T_a, pA \overset{K_{ab}}{\leftrightarrow} pB\})$. Hence by (8) and by (10) $\vdash pB \models pA \models (pA \overset{K_{ab}}{\leftrightarrow} pB)$. Hence by (7) $\vdash pB \models pA \models pA \overset{K_{ab}}{\leftrightarrow} pB$. Hence by (6) *query*. Thus we have completed the proof of (4).

q.e.d.

This article was processed using the LaTeX macro package with LLNCS style