Mechanising BAN Kerberos by the Inductive Method

Giampaolo Bella* Lawrence C Paulson**

Computer Laboratory – University of Cambridge
New Museums Site, Pembroke Street
Cambridge CB2 3QG (UK)

Abstract. The version of Kerberos presented by Burrows et al. [5] is fully mechanised using the *Inductive Method*. Two models are presented, allowing respectively the leak of any session keys, and of expired session keys. Thanks to timestamping, the protocol provides the involved parties with strong guarantees in a realistically hostile environment. These guarantees are supported by the generic theorem prover Isabelle.

1 Introduction

Although pioneered two decades ago [10], he use of formal methods in the field of security protocols has become common practice only during the 1990s. Two paradigms are dominant.

The seminal paper of Burrows et al. [5] suggested the use of a *belief logic* to reason about properties such as freshness. The limitations of this approach have been widely discussed: for instance, reasoning about secrecy. Many extensions have been developed to enhance expressiveness, but they tend to sacrifice the intuitions. Another approach consists in the exhaustive *enumeration of the states* reachable during the computation of a protocol (e.g. [6]). This method requires keeping the state space at a manageable size, which is achieved by simplifying assumptions. However, belief logics can easily reason about authentication, and state enumeration methods can pinpoint simple flaws quickly.

Deep structural properties that had only informal treatment in the past can now be formally expressed by the *inductive method* [12]. The method relies on an algebraic theory of messages with inductively-defined operators applicable to a set of messages. An attacker with the ability of intercepting all traffic over the network is modelled. The attacker can also exploit the accidental loss of secrets by honest agents. Compared with other work, our model is quite realistic. Meadows's approach [8] is important and has an element of induction.

Promising results have been achieved with nonce-based protocols such as Needham-Schroeder, Otway-Rees, Yahalom [12–14], and a flaw has been discovered on a variant of Otway-Rees. This paper presents results about Kerberos, which is based on timestamps. In spite of this difference, many technical results

* Giampaolo.Bella@cl.cam.ac.uk
** Larry.Paulson@cl.cam.ac.uk

from the previously analysed, nonce-based protocols could be easily reused. We present here two models: a basic one leaking any session keys, and a refined one leaking only session keys that have expired. The latter idealisation seems realistic, as the risk of leaking session keys increases over time. The proof script takes a little longer in the latter model, but provides stronger guarantees.

We pay particular attention to what the agents need to check to infer the results stated by the theorems. If theorems rely on assumptions that require knowledge coming from the network, then their importance might be merely theoretic. One might prove that if the spy never gets hold of a session key, then messages encrypted under such a key are reliable. The importance of this theorem is limited, as no honest agent can check its assumption.

The main concepts of the inductive method are given in Section 2. The formalisation of Kerberos is given in Section 3, and its refinement in Section 4. Related work appears in Section 5. Section 6 concludes the paper.

2 The Inductive Method

Only some guidelines are given here. A complete description is published elsewhere [12].

2.1 Overview

A concrete notion of *event* is borrowed from the state enumeration approach. The traffic over the network is created by agents sending messages to each other. So, the basic event Says $A\,B\,msg$ formalises an agent A sending a message msg to an agent B. A *trace* is a set of events.

Intuitively, a security protocol is a non-deterministic program that should guarantee certain properties during its operation. Security protocols are inductively defined as the set of all possible traces. Given as a base case that the empty trace belongs to the set, the formalisation describes how to extend a trace of the set with a new event, according to the protocol operation.

The model does not force agents to reply to any message. Agents can reply late, or reply more than once, or not reply at all. Interleaved runs are possible because agents can even reply to old messages.

Proving a property of a protocol is done by induction. The property should hold on the base trace and, if it holds on a certain trace, then it should hold on all traces extending it. This is simple induction, but it often involves a number of steps that are difficult to manage without tools. The theorem prover Isabelle [11] provides automation.

2.2 Algebra of Messages

The model formalises the knowledge of the attacker (called *spy* in the sequel) by the operator **spies** as follows:

1. $\mathsf{spies}\,[] = \{\mathsf{shrK}\,A \mid A \in bad\}$
2. $\mathsf{spies}\,(\mathsf{Says}\,A\,B\,X\,\#\,evs) = \{X\} \cup \mathsf{spies}\,evs$

The first rule says that the spy's knowledge over the empty trace — i.e. the initial knowledge — consists in the long-term keys of compromised agents. The second, inductive rule expresses the spy's ability to intercept any message on the network.

Given a set H of messages, which is typically expressed in terms of the **spies** operator, we define inductively the following operators.

- $\mathsf{parts}\,H$ is intuitively the set of all components of messages in H. The only items that **parts** can not catch are the encryption keys.
- $\mathsf{analz}\,H$ is intuitively the subset of $\mathsf{parts}\,H$ that does not break ciphers. Thus, to add the body of an encrypted message, its encryption key must be analysable.
- $\mathsf{synth}\,H$ is what the spy can synthesise from H by concatenation and encryption. In particular

$$\mathsf{Agent}\,A \in \mathsf{synth}\,H \qquad\qquad \mathsf{Number}\,T \in \mathsf{synth}\,H$$

  These rules allow the spy to synthesise agent names and timestamps, because they can be guessed with no previous analysis. Note that there is no such a rule for nonces, for they are built as non-clashing random numbers by honest agents. This makes a timestamp-based protocol harder to mechanise.

Everything the spy can synthesise from the observation of the traffic over a trace *evs* is formalised by the set

$$\mathsf{synth}(\mathsf{analz}(\mathsf{spies}\,evs))$$

Recall that the long-term keys of compromised agents belong to the set $\mathsf{spies}\,evs$. Session keys lost by accident belong to the set $\mathsf{analz}(\mathsf{spies}\,evs)$ — this will be explicitly formalised below by the "oops" event. Therefore, such a spy has a potentially infinite behaviour [12].

3 A Model for BAN Kerberos

Kerberos is a cryptographic protocol designed during the mid 1980s at MIT. The BAN version, coming from the paper of Burrows et al. [5], is shown in fig. 1 with lifetimes omitted, as suggested by Bellovin and Meritt [2]. The trusted third party S (called *server* in the sequel) sends A the session key to be shared with B and a ticket that contains the copy of the session key for B. A forwards the

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{Tk, B, Kab, \underbrace{\{Tk, A, Kab\}_{Kb}}_{ticket}\}_{Ka}$
3. $A \rightarrow B : \underbrace{\{Tk, A, Kab\}_{Kb}}_{ticket}, \underbrace{\{A, Ta\}_{Kab}}_{authenticator}$
4. $B \rightarrow A : \{Ta + 1\}_{Kab}$

Fig. 1. BAN Kerberos

ticket to B together with an authenticator to assure B that the sender is the same party to whom the ticket had been issued.

Fig. 2 shows the formalisation of Kerberos by the inductive method, with a few mathematical symbols in place of their ASCII equivalents. Rules `Kb1` to `Kb4` describe how to extend a given trace of the set according to the protocol operation. For instance, rule `Kb2` states that if the first message of Kerberos appears on a trace of the set, then the concatenation of the given trace with the second message of the protocol also is a trace of the set. Rule `Fake` models the introduction on the traffic of all fake messages that the spy can build up. Rule `Oops` models the accidental loss of any session key to the spy. The function

$$\mathsf{Ct} : \mathit{event\ list} \longrightarrow \mathit{bool}$$

formalises the current time over a given trace. The observation that a trace is extended by any protocol step suggested the definition of the current time as the length of the trace. This simple definition has shown sufficient expressiveness thanks to the monotonicity of the length of traces.

The lifetime of a session key, i.e. the time interval within which the key is accepted as fresh by any party, is formalised by the natural number SesKeyLife. Similarly, AutLife formalises the time interval within which an authenticator is considered recent. Therefore, the predicate Expired *Tk evs*, expressing that the timestamp *Tk* has expired over the trace *evs*, is defined by

$$(\mathsf{Ct}\, evs) - Tk > \mathsf{SesKeyLife}$$

The predicate ExpiredAuth *Ta evs*, expressing that the timestamp *Ta* has expired over *evs*, is defined by

$$(\mathsf{Ct}\, evs) - Tk > \mathsf{AutLife}$$

Note that in rule Kb3, A will only forward a ticket that has come with a non-expired session key, and that B requires the same condition to hold in Kb4 together with the condition of having received a non-expired authenticator.

Rule Kb4 does not increment *Ta*. It could do so, but we believe this is irrelevant.

```
kerberos_ban :: event list set
inductive kerberos_ban

Base  [] ∈ kerberos_ban

Fake  [| evs ∈ kerberos_ban; B ≠ Spy; X ∈ synth(analz(spies evs)) |]
      ⟹ Says Spy B X # evs ∈ kerberos_ban

Kb1   [| evs ∈ kerberos_ban; A ≠ Server |]
      ⟹ Says A Server {|Agent A, Agent B|} # evs ∈ kerberos_ban


Kb2   [| evs ∈ kerberos_ban; A ≠ B; A ≠ Server; Key Kab ∉ used evs;
         Says A' Server {|Agent A, Agent B|} ∈ set evs |]
      ⟹ Says Server A Crypt (shrK A)
                        {|Number (Ct evs), Agent B, Key Kab,
                          Crypt (shrK B)
                            {|Number (Ct evs), Agent A, Key Kab|}
                        |} # evs ∈ kerberos_ban


Kb3   [| evs ∈ kerberos_ban; A ≠ B;
         Says A Server {|Agent A, Agent B|} ∈ set evs;
         Says S A Crypt (shrK A) {|Number Tk, Agent B, Key K, X|}
           ∈ set evs;
         ¬ Expired Tk evs |]
      ⟹ Says A B {|X, Crypt K {|Agent A, Number (Ct evs)|}|}
           # evs ∈ kerberos_ban


Kb4   [| evs ∈ kerberos_ban; A ≠ B;
         Says A' B {|Crypt (shrK B) {|Number Tk, Agent B, Key K|},
                     Crypt K {|Agent A, Number Ta|}|} ∈ set evs;
         ¬ Expired Tk evs; ¬ ExpiredAuth Ta evs |]
      ⟹ Says B A Crypt K (Number Ta)
           # evs ∈ kerberos_ban

Oops  [| evs ∈ kerberos_ban; A ≠ Spy;
         Says Server A Crypt (shrK A)
                         {|Number Tk, Agent B, Key K, Ticket|}
           ∈ set evs |]
      ⟹ Says A Spy {|Number Tk, Key K|} # evs ∈ kerberos_ban
```

Fig.2. Formalising BAN Kerberos

3.1 Guarantees about BAN Kerberos

This section presents the main theorems proven about BAN Kerberos. Some proofs were easily adapted from those for the protocols already analysed, others

had to be performed from scratch. Confidentiality guarantees are now expressed from the viewpoint of each party involved in the protocol. The authenticity theorems are new.

1. There must be a trace containing the last message of the protocol that involves two given agents different from the trusted third party.

```
[| A ≠ B; A ≠ Server; B ≠ Server |]
⟹ ∃ Timestamp K. ∃ evs ∈ kerberos_ban.
       Says B A (Crypt K (Number Timestamp)) ∈ set evs
```

   This is the main *possibility* property. The proof is straightforward: resolve by all protocol rules, then simplify.

2. Spy never sees another agent's shared key, unless the agent is compromised.

```
evs ∈ kerberos_ban
⟹ (Key (shrK A) ∈ parts (spies evs)) = (A ∈ bad)
```

   The proof exploits the definition of **spies** (see pag. 3) that allows the spy to see the shared keys of agents belonging to **bad**. Then, induction verifies that the protocol messages protect the other shared keys.

3. The server only sends well-formed messages.

```
[| Says Server A (Crypt K' {|Number Tk, Agent B, Key K, X|})
     ∈ set evs; evs ∈ kerberos_ban |]
⟹ K' = shrK A & K ∉ range shrK &
   X = (Crypt (shrK B) {|Number Tk, Agent A, Key K|})
```

   Induction and simplification form the proof. Despite its simplicity, this technical lemma is useful to prove more complicated guarantees, because it expresses the form of the ticket.

4. If a message of the form of the second of the protocol appears on the traffic, then it originated with the server.

```
[| Crypt (shrK A) {|Number Tk, Agent B, Key K, X|}
     ∈ parts (spies evs); A ∉ bad; evs ∈ kerberos_ban |]
⟹ Says Server A (Crypt (shrK A)
                    {|Number Tk, Agent B, Key K, X|}) ∈ set evs
```

   A simple induction proves that the message originated with the server. The spy could not fake it because A is uncompromised. When A gets hold of such a message, she infers that the session key K really was created by the server at time Tk. By checking Tk against the current time, she is able to decide the *freshness* of K .

5. If the ticket appears on the traffic, then it originated with the server.

```
[| Crypt (shrK B) {|Number Tk, Agent A, Key K|}
     ∈ parts (spies evs);
   B ∉ bad;  evs ∈ kerberos_ban |]
⟹ Says Server A (Crypt (shrK A) {|Number Tk, Agent B, Key K,
                    Crypt (shrK B) {|Number Tk, Agent A, Key K|}|})
      ∈ set evs
```

   The proof follows the same strategy presented for the previous theorem, and B gets the same guarantees as A does: the session key K originated with the server at time Tk.

6. The session key uniquely identifies the message sent by the server.

```
[| Says Server A (Crypt (shrK A) {|Number Tk, Agent B, Key K, X|})
     ∈ set evs;
   Says Server A' (Crypt (shrK A')
                      {|Number Tk', Agent B', Key K, X'|})
     ∈ set evs; evs ∈ kerberos_ban |]
⟹ A=A' & Tk=Tk' & B=B' & X = X'
```

   This is the main *unicity* result, stating that a session key only was generated at one point Tk for one specific pair of agents. The proof rests on induction to find out that session keys are only generated by the server, and that the same key is never generated more than once. This result can be applied to show that the agent who forwards K to B in the third message is the same agent to whom the second message (containing K) was addressed. It simplifies several proofs.

7. If a key can be analysed from the traffic and another session key, then either the two keys are the same, or the first key can be analysed from the traffic alone.

```
[| evs ∈ kerberos_ban;  Kab ∉ range shrK |]
⟹ Key K ∈ analz (insert (Key Kab) (spies evs)) =
    (K = Kab | Key K ∈ analz (spies evs))
```

   The theorem means that session keys are never used to encrypt other keys, so the compromise of one key would not compromise others. It is a crucial rewrite rule for other theorems based on the analz operator. Although the proof requires a number of lemmas about analz [12], it executes in only 20 seconds.

8. Spy can not see the session key sent by the server in the second message if such a key has not been accidentally lost (by an "oops" event), and the two recipients are uncompromised.

```
[| Says Server A (Crypt K' {|Number Tk, Agent B, Key K, X|})
     ∈ set evs;
   (ALL T. Says A Spy {|T, Key K|} ∉ set evs);
   A ∉ bad;  B ∉ bad;  evs ∈ kerberos_ban |]
⟹ Key K ∉ analz (spies evs)
```

This theorem states the *confidentiality* of the session key from the server's viewpoint, because it relies on a Says event that only the server can check. The key is obviously required not to have been accidentally leaked, although this can not be checked (this strong assumption is relaxed in the refined model — see next section). The recipients of the session key must be uncompromised, otherwise they would trivially reveal it to the spy. Because A is uncompromised, the external encryption of the second message can be proven safe. The ticket forwarded in the third message keeps the session key secure because also B is uncompromised.

9. If a message of the form of the second message of the protocol appears on the traffic, and contains a session key for two uncompromised agents that has not been leaked by accident, then the session key can not be seen by the spy.

```
[| Crypt (shrK A) {|Number Tk, Agent B, Key K, X|}
     ∈ parts (spies evs);
   (ALL T'. Says A Spy {|T', Key K|} ∉ set evs);
   A ∉ bad;  B ∉ bad;  evs ∈ kerberos_ban |]
⟹ Key K ∉ analz (spies evs)
```

The theorem expresses the confidentiality of the session key from A's viewpoint, as it rests on conditions that A can check when she receives the second message of the protocol, provided that her interlocutor is uncompromised. The proof applies theorem 4 to theorem 8.

10. If the ticket appears on the traffic, and contains a session key for two uncompromised agents that has not been leaked by accident, then the session key can not be seen by the spy.

```
[| Crypt (shrK B) {|Number Tk, Agent A, Key K|}
     ∈ parts (spies evs);
   (ALL T'. Says A Spy {|T', Key K|} ∉ set evs);
   A ∉ bad;  B ∉ bad;  evs ∈ kerberos_ban |]
⟹ Key K ∉ analz (spies evs)
```

This theorem is analogous to the previous one, but expresses the confidentiality of the session key from B's viewpoint. The proof applies theorem 5 to theorem 8.

11. If the fourth message appears, and is encrypted under a safe session key, then it originated with B.

```
[| Crypt K (Number Ta) ∈ parts (spies evs);
   Crypt (shrK A) {|Number Tk, Agent B, Key K, X|}
     ∈ parts (spies evs);
   ALL T. Says A Spy {|T, Key K|} ∉ set evs;
   A ∉ bad;  B ∉ bad;  evs ∈ kerberos_ban |]
⟹ Says B A (Crypt K (Number Ta)) ∈ set evs
```

This theorem expresses the *authentication* of B to A. If B were compromised, the spy could easily impersonate him. The non-trivial case is when B is honest. To assure that the session key is kept secret, theorem 8 about confidentiality is applied. Induction then shows that the fourth message only could originate with B. If A can successfully decrypt by K the message containing the timestamp Ta, then she gets evidence that B shares K with her as a session key. A also infers that B was present after she has issued Ta (this timestamp is actually used as a nonce).

12. If the authenticator appears, and is encrypted under a safe session key, then it originated with A.

```
[| Crypt K {|Agent A, Number Ta|} ∈ parts (spies evs);
   Crypt (shrK B) {|Number Tk, Agent A, Key K|}
     ∈ parts (spies evs);
   ALL T. Says A Spy {|T, Key K|} ∉ set evs;
   A ∉ bad;  B ∉ bad;  evs ∈ kerberos_ban |]
⟹ Says A B {|Crypt (shrK B) {|Number Tk, Agent A, Key K|},
               Crypt K {|Agent A, Number Ta|}|} ∈ set evs
```

This theorem expresses the authentication of A to B and can be discussed as the previous one. The proof follows the same pattern: apply theorem 8 about confidentiality, and then use induction. If B can decrypt the authenticator successfully, he understands that A agrees on the session key K. Then, B can check the timestamp Ta against the current time and infer when A was present. Therefore, the authenticator fulfils the aims for which it was envisaged.

4 Refining the Model

The theorems presented in the previous section support the claim that BAN Kerberos assesses strong goals of confidentiality and of authentication. (Another protocol providing similar authentication goals is Yahalom [14]). However, these guarantees rely on session keys that have not been leaked by accident, a condition that can not be checked by any honest agents.

Since the probability of secrets to become compromised increases over time (the longer they are on the traffic, the higher the risk), it seems realistic to

assume that session keys are only leaked when they have expired. The Oops rule is refined accordingly by adding to its assumptions the temporal check

Expired *Tk evs*

The main guarantees can be refined as follows.

8′. Confidentiality of the session key for the sever.

```
[| Says Server A (Crypt K {|Number Tk, Agent B, Key K, X|})
     ∈ set evs;
   ¬ Expired Tk evs; A ∉ bad; B ∉ bad; evs ∈ kerberos_ban |]
⟹ Key K ∉ analz (spies evs)
```

9′. Confidentiality of the session key for A.

```
[| Crypt (shrK A) {|Number Tk, Agent B, Key K, X|}
     ∈ parts (spies evs);
   ¬ Expired Tk evs; A ∉ bad; B ∉ bad; evs ∈ kerberos_ban |]
⟹ Key K ∉ analz (spies evs)
```

10′. Confidentiality of the session key for B.

```
[| Crypt (shrK B) {|Number Tk, Agent A, Key K|}
     ∈ parts (spies evs);
   ¬ Expired Tk evs; A ∉ bad; B ∉ bad; evs ∈ kerberos_ban |]
⟹ Key K ∉ analz (spies evs)
```

11′. Authentication of B to A.

```
[| Crypt K (Number Ta) ∈ parts (spies evs);
   Crypt (shrK A) {|Number Tk, Agent B, Key K, X|}
     ∈ parts (spies evs);
   ¬ Expired Tk evs; A ∉ bad; B ∉ bad; evs ∈ kerberos_ban |]
⟹ Says B A (Crypt K (Number Ta)) ∈ set evs
```

12′. Authentication of A to B.

```
[| Crypt K {|Agent A, Number Ta|} ∈ parts (spies evs);
   Crypt (shrK B) {|Number Tk, Agent A, Key K|}
     ∈ parts (spies evs);
   ¬ Expired Tk evs; A ∉ bad; B ∉ bad; evs ∈ kerberos_ban |]
⟹ Says A B {|Crypt (shrK B) {|Number Tk, Agent A, Key K|},
               Crypt K {|Agent A, Number Ta|}|} ∈ set evs
```

The proofs for the basic model could be adapted by including some arithmetic reasoning to deal with the temporal checks. However, the new theorems provide stronger guarantees, for the temporal assumptions can be easily checked by any agents.

5 Related Work

Bolignano analyses crypto-protocols by modelling the states of agents, and gives a procedure to decide mechanically whether the spy can see certain items [3]. Although the spy's knowledge is unbounded, the method needs a substantial formal overhead. It is only applied to a trivial protocol that uses neither nonces nor timestamps, and that establishes no secrets. It is not clear to us how the method could handle general protocols.

The mentioned work of Burrows et al. [5] contains the first application of formal methods to BAN Kerberos. Although they provide no confidentiality analysis, they state formally that, at the end of the protocol run, the two parties know they are agreeing on the same session key. The proof is very short, but the whole reasoning has been criticised as too abstract, and the same approach has failed to discover known weaknesses of other protocols.

Brackin [4] extends and mechanises this work using the HOL theorem prover. It is a good attempt of supporting the BAN logic by machine, but it does not enhance the expressiveness of the logic itself.

Lowe has analysed timestamps by state enumeration on a simple two-message protocol [7].

State enumeration has been tested on Kerberos Version IV by Mitchell et al. [9]. They first tackle a system of size three, and find no attacks. Then, they discover and fix a known weakness on a system of size four. However, their analysis omits timestamps, and does not allow multiple runs. Relaxing the last two limitations is promised by the authors as future work.

6 Conclusion

The paper has presented the mechanisation of the BAN Kerberos protocol by the Inductive Method using the theorem prover Isabelle. The work is based on the formalisation of timestamps, and has benefited from the technical results sketched by the authors about Kerberos Version IV [1].

Two models are investigated: the first allows the leak of any session keys, the second only considers the leak of session keys that have expired. The second model only requires some minor modifications to the first. Strong guarantees of freshness, confidentiality, and authentication could be proven in both cases. Confidentiality is now stated from the viewpoint of each party involved in the protocol. Authentication is expressed in a form that is useful to the parties.

Although the second model makes the — fairly realistic — assumption that session keys can only be leaked when they have expired, it provides strong guarantees based on simple temporal checks. The choice of the most realistic model is left to the reader.

The proofs of the theorems require a deep knowledge of Isabelle, and are omitted for space limitations[1]. The entire work (both models) required three

[1] Full proof scripts available at `http://www.cl.cam.ac.uk/~gb221/BanKerberos/`

weeks human time. The proof script of the basic model amounts to 80 commands, and runs in 140 seconds CPU time on a Sun SuperSPARC Model 61. Commands become 90 and execution time rises to 160 seconds for the script of the second model.

References

1. G. Bella, L. C. Paulson. Using Isabelle to Prove Properties of the Kerberos Authentication System. In *Proc. of Workshop on Design and Formal Verification of Security Protocols,* Orman and Meadows (eds.), DIMACS, 1997.
2. S. M. Bellovin, M. Merritt. Limitations of the Kerberos authentication system. *Computer Comm. Review*, 20(5), 119-132, 1990.
3. D. Bolignano. Towards a Mechanization of Cryptographic Protocol Verification. In *Proc. of Conference on Computer Aided Verification*, Springer Verlag, 1997.
4. S. H. Brackin. A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols. In *Proc. of Computer Security Foundations Workshop*, IEEE Press, 1996.
5. M. Burrows, M. Abadi, R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London,* 426:233-271, 1989.
6. G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems,* Margaria and Steffen (eds.), LNCS1055, Springer Verlag, 147-166, 1996.
7. G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. Oxford University, Computing Laboratory, *Technical Report*, 1996.
8. C. Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 26(2), 113-131, 1996.
9. J. C. Mitchell, M. Mitchell, U. Stern: Automated Analysis of Cryptographic Protocols Using Murphi. In *Proc. of Symposium on Security and Privacy*, IEEE Press, 1997.
10. R. M. Needham, M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 993-999, 1978.
11. L. C. Paulson. *Isabelle: A Generic Theorem Prover.* Springer, 1994. LNCS 828.
12. L. C. Paulson. Proving properties of security protocols by induction. In *Proc. of Computer Security Foundations Workshop*, IEEE Press, 1997.
13. L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proc. of Computer Security Foundations Workshop*, IEEE Press, 1997.
14. L. C. Paulson. On Two Formal Analyses of the Yahalom Protocol. Cambridge University, Computer Laboratory, *Technical Report No. 432*, 1997.