Technical Report

UCAM-CL-TR-447
ISSN 1476-2986

Number 447

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Are timestamps worth the effort? A formal treatment

Giampaolo Bella, Lawrence C. Paulson

September 1998

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

ISSN 1476-2986

Abstract

Theorem proving provides formal and detailed support to the claim that timestamps can give better freshness guarantees than nonces do, and can simplify the design of crypto-protocols. However, since they rely on synchronised clocks, their benefits are still debatable. The debate should gain from our formal analysis, which is achieved through the comparison of a nonce-based crypto-protocol, Needham-Schroeder, with its natural modification by timestamps, Kerberos.

1 Introduction

Computer networks are insecure. Information transmitted can be intercepted, altered and forwarded to unintended recipients.

To solve such problems, cryptography was introduced, but throughout the last two decades the problem has proven to be much harder than expected. Crypto-protocols can suffer several weaknesses. One is the illegitimate replay of a message by an eavesdropper, which fools the legitimate recipient into accepting old data as fresh.

The best-known protocol suffering the replay weakness is the shared-key Needham-Schroeder [12], as criticised by Denning and Sacco [5]. After commenting that the problem comes from the inaccurate use of nonces, they suggest the use of timestamps to solve it.

Such a solution is intuitive and widely accepted. With the help of formal methods to analyse crypto-protocols, we can investigate whether this intuition is valid. The analysis of Burrows et al. [4] is acknowledged amongst the main contributions to the topic, but is not conclusive in view of the known limitations of authentication logics. The intrinsic finiteness of state enumeration methods does not allow a broad treatment of timestamping, but a two-message protocol has been analysed with a fixed expiring time of one second [9].

This paper presents the first formal comparison between the use of timestamps or nonces in the same protocol design. Machine proofs based on a detailed crypto-protocol model support the claim that timestamps can simplify the design, and possibly enhance the guarantees assessed.

However, it is well known that, in order to be reliable, timestamps require the use of a secure synchronisation protocol, and it could be argued that the threats to this protocol might offset any gains obtained by timestamps in the crypto-protocol. We do not address this question.

The paper sketches the crypto-protocol model in Section 2, motivates the choice of Needham-Schroeder and Kerberos in Section 3, and presents the model for Kerberos in Section 4. Section 5 discusses the comparison of the two protocols, and Section 6 concludes.

2 Analysing Crypto-Protocols by Induction

Machine proofs can be performed in Isabelle [13] using simple mathematical induction, a concept also adopted by Meadows [10], and more recently by Fabrega et al. [6]. Encouraging results with protocols such as Otway-Rees and Yahalom [15, 16, 17] have been achieved, so that this method could complement other analyses by means of *belief logic* formalisms (e.g. [4, 3]), and *state enumeration* tools (e.g. [7, 8, 18]). The full details are presented elsewhere [14].

A protocol is modelled *inductively* as the set of all possible traces of events. The event Says $A\ B\ msg$ formalises agent A sending message msg to agent B. A spy is formalised with the ability to intercept all network traffic, and to build faked messages out of it, using the shared keys of a set bad of compromised agents, and session keys accidentally leaked by honest agents.

Given a set H of messages (typically the set of messages over a trace), parts H is straightforwardly defined by induction, and denotes all components of messages in H, including bodies of encrypted messages.

3 Choosing the Crypto-Protocols

The shared-key Needham-Schroeder protocol is amongst the best-known ones of the literature, and suffers from a replay weakness found by Denning and Sacco [5]. This protocol, "*modified with the addition of timestamps*" (Miller et al. [11]) becomes the authentication model of Kerberos, as can be seen from Fig. 1 and Fig. 2 showing the versions analysed by Burrows et al. in [4]. The two protocols rest on the same message structure, and this makes them an ideal case-study to compare nonces with timestamps.

$$
\begin{array}{lll}
1. & A \rightarrow \mathsf{S} & : \ A, B, Na \\
2. & \mathsf{S} \rightarrow A & : \ \{Na, B, Kab, \underbrace{\{Kab, A\}_{Kb}}_{ticket}\}_{Ka} \\
3. & A \rightarrow B & : \ \underbrace{\{Kab, A\}_{Kb}}_{ticket} \\
4. & B \rightarrow A & : \ \{Nb\}_{Kab} \\
5. & A \rightarrow B & : \ \{Nb - 1\}_{Kab}
\end{array}
$$

Figure 1: Needham-Schroeder (shared keys)

$$
\begin{array}{lll}
1. & A \rightarrow \mathsf{S} & : \ A, B \\
2. & \mathsf{S} \rightarrow A & : \ \{Ts, B, Kab, \underbrace{\{Ts, A, Kab\}_{Kb}}_{ticket}\}_{Ka} \\
3. & A \rightarrow B & : \ \underbrace{\{Ts, A, Kab\}_{Kb}}_{ticket}, \{A, Ta\}_{Kab} \\
4. & B \rightarrow A & : \ \{Ta + 1\}_{Kab}
\end{array}
$$

Figure 2: Kerberos (BAN version)

Both protocols rely on a trusted third party S that has knowledge of all agents' shared keys. Kx denotes agent X's key shared with S, Kab

the session key for A and B, Nx and Tx respectively the nonce and the timestamp issued by X. Concatenation is expressed by fat braces, outer-level braces being omitted. Lifetimes are omitted in Kerberos, as they are known to all.

4 Formalising the two Crypto-Protocols

The mechanisation of Needham-Schroeder already achieved by the second author was extended a little for the sake of the comparison. It is omitted here, but may be found online[1]. It allows the accidental loss of a session key to the spy, and formalises the fifth message by $\{\!|Nb, Nb|\!\}_{Kab}$.

The full mechanisation of the BAN Kerberos had to be performed [2]. This gained from the experience on few technical results already proven about Kerberos Version IV [1]. The original model is shown in Fig. 3. Recall a protocol is modelled inductively as a set of traces. Rule Base states that the empty trace belongs to the set. Rules Kb1, Kb2, Kb3, 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 ability of the spy to say anything she can fake from the observation of the traffic. Rule Oops models the accidental loss of a session key to the spy.

The function CT formalises the current time over a given trace. Both A and B check the timestamp that accompanies the session key to be fresh. B also checks the freshness of the timestamp found inside the authenticator. The two timestamps have different lifetimes.

5 Comparing the two Crypto-Protocols

The main guarantees proven about Kerberos[2] are slightly more difficult to prove than those about Needham-Schroeder. Technically speaking, this is due to the design of the third message — a double cipher instead of a single one.

Both protocols deliver a confidential session key, and provide strong guarantees to A, the initiator of a run. Chiefly, A can check whether the session key is fresh when she receives it, and can get evidence about the active presence of B on the network. The analysis reported below shows formally how these two guarantees are provided to B by Kerberos in the same form, and by Needham-Schroeder in a weaker form.

[1]See "NS_Shared" at `http://www4.informatik.tu-muenchen.de/~isabelle/library/HOL/Auth/`

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

```
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 Ts, Agent B, Key K, Ticket|}
           ∈ set evs;
         (CT evs) ≤ SesKeyLife + Ts |]
      ⟹ Says A B {|Ticket, Crypt K {|Agent A, Number (CT evs)|}|}
           # evs ∈ kerberos_ban


Kb4   [| evs ∈ kerberos_ban; A ≠ B;
         Says A' B {|Crypt (shrK B) {|Number Ts, Agent B, Key K|},
                     Crypt K {|Agent A, Number Ta|}|} ∈ set evs;
         (CT evs) ≤ SesKeyLife + Ts;  (CT evs) ≤ AutLife + Ta |]
      ⟹ Says B A Crypt K {|Number Ta, Number Ta|}
           # evs ∈ kerberos_ban

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

Figure 3: Formalising BAN Kerberos Inductively

5.1 Freshness of the Session Key

The freshness of the session key rests on the *ticket integrity* theorem stating that, if the ticket $\{\!|Ts, A, Kab|\!\}_{Kb}$ appears on the Kerberos traffic, and agent B is uncompromised, then such a ticket originated with the server:

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

Assuming B to be uncompromised is crucial to protect the ticket encrypted under B's shared key from the spy. Since the server is trustworthy, Ts represents the issue time of the ticket. Therefore, B can determine whether the ticket is fresh by checking Ts: if it turns out to be older than the allowed lifetime, then B can suspect the ticket to be the replay of an old one. Freshness of the ticket also signifies freshness of the session key transmitted inside it.

This result confirms and strengthens the analysis of Kerberos by Burrows et al. [4]. The proof rests on a simple induction on the parts operator and executes in 3 seconds on a Sun SuperSPARC model 61.

To realise the benefit coming from timestamping, let us look look at the *ticket integrity* theorem proven for Needham-Schroeder on the ticket $\{\!|Kab, A|\!\}_{Kb}$:

```
[| Crypt (shrK B) {|Key K, Agent A|} ∈ parts (spies evs);
   B ∉ bad;  evs ∈ kerberos_ban |]
==> ∃ Na. Says Server A Crypt (shrK A) {|NA, Agent B, Key K,
                                         Crypt (shrK B)
                                           {|Key K, Agent A|}
                                       |} ∈ set evs
```

Although the theorem assures B that the ticket originated with the server, this time B can not know whether or not he is accepting old data as fresh, because no knowledge can be inferred about the nonce Na.

One might argue that the presence of a nonce inside the Needham-Schroeder ticket in the place of Ts could achieve the same result achieved for Kerberos, but this is false because B has not yet been involved in the protocol at reception of the ticket. Therefore, he could not check any nonces of his.

This emphasises the different philosophy of timestamps and nonces. Since timestamps are a linear order, any agent could check the freshness of a message containing a timestamp at any point in the protocol run — even without having been involved before — as they only would need to know the adequate lifetime, and lifetimes are not secret. However, the parties must have synchronised clocks. By contrast, in a nonce-based system, an agent wanting to check the freshness of a message should have participated earlier in the protocol and issued a nonce. Then, some other agent should have inserted the same nonce into the message. This generally ends up in a more complicated protocol structure.

5.2 Authentication of A to B

The following theorem states that if the Kerberos authenticator $\{\!|A, Ta|\!\}_{Kab}$ appears under certain conditions, then it originated with A:

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

Note that the authenticator is encrypted under the session key K received by B. Moreover, B has to trust K to have not been leaked by accident.

By stating that A has sent B a message containing the authenticator, the theorem gives evidence to B of the active presence of A at time Ta, i.e. it authenticates A to B. The proof is based on the observation that the authenticator is encrypted under the session key transmitted to B inside the ticket. This ticket must have originated with the server, by application of the ticket integrity theorem discussed in the previous section. The main step is then proving that a session key remains secure provided that no Oops rule applies (this is why the third, strong assumption of the theorem is mandatory). To achieve this, two subsidiary results are necessary. One states that a session key is never distributed twice by the server, namely that a session key uniquely identifies the other components of the message that delivers it. A second and crucial one states that no keys are encrypted by a session key. Finally, induction on parts derives the result. The full script takes 110 seconds to execute on a Sun SuperSPARC model 61.

Needham-Schroeder needs the fifth message to give B a similar guarantee:

```
[| Crypt (shrK B) {|Key K, Agent A|} ∈ parts (spies evs);
   Crypt K {|Nonce NB, Nonce NB|} ∈ parts (spies evs);
   Says B A (Crypt K (Nonce NB)) ∈ set evs;
   ∀ NA NB. Says A Spy {|NA, NB, Key K|} ∉ set evs;
   A ∉ bad;  B ∉ bad;  evs ∈ ns_shared |]
==> Says A B (Crypt K {|Nonce NB, Nonce NB|}) ∈ set evs
```

If the session key received by B has not been leaked by accident, he infers A's activity at some time after he issued Nb. Conversely, Kerberos informed B of the exact instance of time Ta.

Therefore, not only have timestamps avoided the need for a fifth message, but also strengthened the outcome. However, in most real-world applications the guarantees of the two protocols can be regarded as equivalent. On the other hand, if we allow an uncompromised agent A ($A \notin$ bad) to behave badly, she could insert a newer timestamp in the authenticator of Kerberos, and authenticate herself to B as being active at a later time. Needham-Schroeder does not allow this.

The risk of secrets becoming compromised increases over time. On this basis, it is realistic to assume that session keys can be lost only when they expire. The Oops rule is refined by including the temporal check

```
(CT evs) - Ts > SesKeyLife
```

On the refined model, we proved the following theorem:

```
[| Crypt (shrK B) {|Number Ts, Agent A, Key K|} ∈ parts (spies evs);
   Crypt K {|Agent A, Number Ta|} ∈ parts (spies evs);
   (CT evs) - Ts ≤ SesKeyLife;
   A ∉ bad;  B ∉ bad;  evs ∉ kerberos_ban |]
==> Says A B {| Crypt (shrK B) {|Number Ts, Agent A, Key K|},
                                Crypt K {|Agent A, Number Ta|}
                            |} ∈ set evs
```

This is the strongest authentication guarantee amongst those presented above, as it is based on a simple temporal check that B can perform at reception of the session key. If such a key has not expired, then A was present at time Ta. The proof has not become more complex, and rests on the same strategy presented above.

Such a kind of refinement is clearly impossible on Needham-Schroeder.

6 Conclusion

The paper compares the crucial guarantees accomplished by the Needham-Schroeder protocol with those accomplished by Kerberos. The guarantees are proven using a detailed operational formalisation of crypto-protocols, based on induction.

The first Kerberos model shows that this protocol provides stronger freshness guarantees than Needham-Schroeder does, and similar authentication guarantees with one message fewer. The refined model suggests that, in realistic conditions, the authentication of A to B provided by Kerberos is much stronger as it is based on checks that B can perform.

Since Kerberos is the natural modification of Needham-Schroeder "*with the addition of timestamps*" (Miller et al. [11]), the analysis formally supports the claim that, on the same design, timestamps achieve better guarantees than nonces do, and can even simplify the design. However, they rely on synchronised clocks, which signifies a potential risk for other attacks. Whether or not it is worth running these risk is an open question, and could be influenced by the specific application domain.

Our work also suggests that nonce-based crypto-protocols could assess the same guarantees as timestamp-based ones only by means of a more careful design. This is confirmed by other protocols, such as Otway-Rees and Yahalom, analysed by the second author in [14, 17].

References

[1] G. Bella, L. C. Paulson. Using Isabelle to Prove Properties of the Kerberos Authentication System. *Proc. of DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.

[2] G. Bella, L. C. Paulson. Mechanising BAN Kerberos by the Inductive Method. Proc. of *Conference on Computer Aided Verification*, Springer 1998, LNCS 1427.

[3] S. H. Brackin. A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols. In *Proc. of Computer Security Foundations Workshop*, IEEE Press, 1996.

[4] M. Burrows, M. Abadi, R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London,* 426:233-271, 1989.

[5] D. E. Denning, G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8), 533-536, 1981.

[6] F. J. T. Fábrega, J. C. Herzog, J. D. Guttman. Strand Spaces: Why is a Security Protocol Correct?. In *Proc. of Symposium on Security and Privacy*, IEEE Press, 1998.

[7] R. Kemmerer, C. Meadows, J. Millen. Three Systems for Cryptographic Protocol Analysis. *Journal of Cryptology*, 7(2), 79-130, 1994.

[8] 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.), volume 1055 of Lecture Notes in Computer Science, Springer Verlag, 147-166, 1996.

[9] G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. Oxford University, Computing Laboratory, *Technical Report*, 1996.

[10] C. Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 26(2), 113-131, 1996.

[11] S. P. Miller, J. I. Neuman, J. I. Schiller, J. H. Saltzer. Kerberos authentication and authorisation system. *Project Athena Technical Plan,* Sec. E.2.1, 1-36, MIT, 1989.

[12] R. M. Needham, M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 993-999, 1978.

[13] L. C. Paulson. *Isabelle: A Generic Theorem Prover.* Springer, 1994. LNCS 828.

[14] L. C. Paulson. Proving properties of security protocols by induction. In *Proc. of Computer Security Foundations Workshop*, IEEE Press, 1997.

[15] L. C. Paulson. Mechanized proofs of security protocols: Needham-Schroeder with public keys. Cambridge University, Computer Laboratory, *Technical Report No. 413*, 1997.

[16] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proc. of Computer Security Foundations Workshop*, IEEE Press, 1997.

[17] L. C. Paulson. On Two Formal Analyses of the Yahalom Protocol. Cambridge University, Computer Laboratory, *Technical Report No. 432*, 1997.

[18] S. Schneider. Verifying Authentication Protocols Using CSP. In *Proc. of 10th IEEE Computer Security Foundations Workshop*, IEEE Press, 1997.