Mechanized Proofs of Security Protocols: Needham-Schroeder with Public Keys

Lawrence C. Paulson
Computer Laboratory
University of Cambridge

lcp@cl.cam.ac.uk

January 1997

Abstract

The inductive approach to verifying security protocols, previously applied to shared-key encryption [8], is here applied to the public key version of the Needham-Schroeder protocol. As before, mechanized proofs are performed using Isabelle/HOL. Both the original, flawed version and Lowe's improved version are studied; the properties proved highlight the distinctions between the two versions. The results are compared with previous analyses of the same protocol. The analysis reported below required only 30 hours of the author's time. The proof scripts execute in under three minutes.

Contents

1 Introduction

Many researchers are investigating formal methods of modelling and verifying security protocols. The Needham-Schroeder public-key protocol has become a standard test case because it is simple and well-known. It is one of the many protocols analyzed by Burrows et al. [2]. Lowe has demonstrated the potential of his model-checking approach by discovering a subtle flaw [4]. Schneider [10] has published hand proofs concerning two versions of the protocol, while Meadows [6] has done extensive studies using the NRL Protocol Analyzer.

I have recently reported [8] a method of modelling protocols as traces and of establishing correctness properties using the proof assistant Isabelle [7]. (Please read that paper before attempting this one.) Hitherto, I have assumed shared-key encryption. In applying the method to public-key protocols, Needham-Schroeder is the obvious choice.

This paper describes the protocol (§2) and a formal model of it (§3). It discusses the mechanical proofs, first from A's point of view (§4) and then from B's (§5). It presents some Isabelle notation, mainly for purposes of illustration (§6). Lowe's strengthened version requires little additional discussion (§7) because most of the proofs are unaffected. Finally there is a discussion of related work and conclusions (§§8–9).

2 The Protocol and Lowe's Attack

If public keys are universally known then the Needham-Schroeder protocol consists of three steps:[1]

$$\begin{array}{ll} 1. & A \rightarrow B : \{\!|Na, A|\!\}_{Kb} \\ 2. & B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka} \\ 3. & A \rightarrow B : \{\!|Nb|\!\}_{Kb} \end{array}$$

Message 2 assures A of B's presence, since only B could have decrypted $\{\!|Na, A|\!\}_{Kb}$ to extract the freshly-invented nonce Na. Similarly, message 3 assures B of A's presence. Burrows et al. [2] claimed a further property, namely that Na and Nb become known only to A and B. (Such shared secrets might be used as passwords or to compute a session key.) Lowe refuted this claim, noting that if A ran the protocol with an enemy C, then C could start a new run with any agent B, masquerading as A [4].

One might argue that this is no attack at all. An agent who is careless enough to talk to the enemy cannot expect any guarantees. The mechanized analysis presented below reveals that the protocol's guarantees for A are

[1]Dispensing with that assumption requires four additional steps to distribute public keys. Burrows et al. [2] identified a flaw in this part of the protocol: there was no guarantee that the public keys were fresh.

adequate. However, those for B are not: they rely upon A's being careful, which is a stronger assumption than mere honesty. Moreover, the attack can also occur if A talks to an honest agent whose private key has been compromised. Lowe suggests a simple fix that provides good guarantees for both A and B.

3 Modelling the Protocol

The shared-key model reported earlier [8] assumes that every agent shares a long-term key with a server. Some of these keys have been *compromised*: lost to the spy. In public-key model, an agent A has a public key $\mathsf{pubK}\,A$, known to all agents, and a private key $\mathsf{priK}\,A$, possibly compromised. No private key coincides with any public key.

Let us start with the original (uncorrected) Needham-Schroeder protocol. It is modelled by an inductive definition, as usual. There are five rules: one for the empty trace, one for enemy action (which is identical to the rule adopted for shared-key protocols) and three for the protocol steps. More precisely, the protocol steps are as follows:

1. If *evs* is a trace, Na is a fresh nonce and B is an agent distinct from A, then we may add the event

$$\mathsf{Says}\,A\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|Na, A|\!\}).$$

2. If *evs* is a trace containing an event of the form

$$\mathsf{Says}\,A'\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|Na, A|\!\}),$$

   and Nb is a fresh nonce and $B \neq A$, then we may add the event

$$\mathsf{Says}\,B\,A\,(\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb|\!\}).$$

   Writing the sender as A' means that B does not know who sent the message.

3. If *evs* is a trace containing the two events

$$\begin{array}{l}\mathsf{Says}\,A\,B\;\;(\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|Na, A|\!\})\\ \mathsf{Says}\,B'\,A\,(\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb|\!\})\end{array}$$

   and $B \neq A$, then we may add the event

$$\mathsf{Says}\,A\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|Nb|\!\}).$$

   A decrypts the message and checks that Na agrees with the nonce she previously sent to B. She replies to B's challenge by sending back Nb.

The event $\mathsf{Says}\,A\,B\,X$ represents an attempt by A to send X to B. The act of receiving a message is not modelled explicitly, but an agent can only perform step $i+1$ if it has received something suitable as step i. Steps 2 and 3 are specified above according to this convention.

Two further rules are required in order to allow for the empty trace and enemy action. As in my previous work, the enemy may say anything he feasibly could say, and agents may engage in any number of possibly interleaved runs. There is no 'oops' message [8] because the protocol does not distribute session keys. However, one can ask—as has Meadows [6]—what might happen if one of the nonces is compromised.

Figure 1 presents the Isabelle theory file. The gist of the protocol should be evident even to readers unfamiliar with Isabelle. However, the paper takes some liberty with Isabelle's syntax for the sake of readability. Some mathematical symbols appear instead of their ASCII equivalents, and variables are occasionally renamed. Some comments are omitted.

4 Proving Guarantees for A

The guarantees for A are that her nonce remains secret—from the spy—and that B is present. The latter follows from the former, for if the spy does not know Na then he could not have sent message 2. The proofs require, as lemmas, unicity properties for Na. Informally, they say that Na is only used once.

- No value is ever used both as Na and as Nb, even in separate runs.

- The value of nonce Na in any message of the form $\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|Na, A|\!\}$ uniquely determines the agents A and B, over all traffic.

Both lemmas assume Na to be secret and form part of an inductive proof that Na really is secret. They hold because honest agents are specified to choose fresh nonces, with a negligible probability of collision.[2] The model assumes that nonces are virtually impossible for an enemy to predict.

The guarantee for A after step 2 is that the message indeed originated with B, provided it contains the expected nonce. The guarantee is consistent with Lowe's attack because, as always, it considers runs between two uncompromised principals. If A runs the protocol with the spy then her guarantee is void. Lowe himself found no problem with the protocol from A's viewpoint [4, §3.2]; his attack concerns the guarantee for B.

[2]For example, the agent might hash its private key alongside a counter.

```
NS_Public_Bad = Public +

consts  lost       :: agent set
        ns_public  :: event list set
inductive ns_public
  intrs
    Nil  "[] ∈ ns_public"

         (*The spy MAY say anything he CAN say.*)
    Fake "[| evs ∈ ns_public;  B≠Spy;
             X ∈ synth (analz (sees lost Spy evs)) |]
          ⟹ Says Spy B X  # evs ∈ ns_public"

         (*Alice initiates a protocol run.*)
    NS1  "[| evs ∈ ns_public;  A≠B;  Nonce NA ∉ used evs |]
          ⟹ Says A B (Crypt (pubK B) {|Nonce NA, Agent A|})
                # evs  ∈  ns_public"

         (*Bob responds to Alice's message.*)
    NS2  "[| evs ∈ ns_public;  A≠B;  Nonce NB ∉ used evs;
             Says A' B (Crypt (pubK B) {|Nonce NA, Agent A|})
               ∈ set_of_list evs |]
          ⟹ Says B A (Crypt (pubK A) {|Nonce NA, Nonce NB|})
                # evs  ∈  ns_public"

         (*Alice proves her existence by returning NB to Bob.*)
    NS3  "[| evs ∈ ns_public;  A≠B;
             Says A  B (Crypt (pubK B) {|Nonce NA, Agent A|})
               ∈ set_of_list evs;
             Says B' A (Crypt (pubK A) {|Nonce NA, Nonce NB|})
               ∈ set_of_list evs |]
          ⟹ Says A B (Crypt (pubK B) (Nonce NB))
                # evs  ∈  ns_public"
```

Figure 1: Specifying a Protocol

5 Proving Guarantees for B

The situation as seen by B is almost symmetrical to that seen by A. Proving by induction that Nb remains secret would authenticate A. Most of the Isabelle proof scripts for A's theorems also work for B with trivial alterations. It is easy to prove that, if Nb is secret, then its value in any message of the form $\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb|\!\}$ uniquely determines A and Na.

Unfortunately, Nb does not remain secret. The attempt to prove its secrecy fails, leaving a subgoal that contains (as a past event) A's sending message 1 to a compromised agent. The subgoal describes a consistent set of circumstances: Lowe's attack. Details appear in §6 below.

Weaker properties can be proved. If A never sends Nb to anybody in step 3 of the protocol, then Nb remains secret. In consequence, if B receives Nb in step 3 then A has sent it, and is therefore present. However, A may have sent it to anybody.

Here is a sketch of the proof. Follow the usual argument (based on A's proofs), but assume that A says no messages of the form $\mathsf{Crypt}(\mathsf{pubK}\,C)Nb$ for any C. With this additional assumption, Nb does remain secret; it then follows that if B sends $\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb|\!\}$ as step 2 and receives $\mathsf{Crypt}(\mathsf{pubK}\,B)Nb$, then this reply came from A. Since this conclusion contradicts the assumption, B cannot receive $\mathsf{Crypt}(\mathsf{pubK}\,B)Nb$. Taking the contrapositive[3] of this result negates the assumption, leading to the conclusion that A has indeed sent the message $\mathsf{Crypt}(\mathsf{pubK}\,C)Nb$ for some C.

Thus, there is a general strategy for proving that decrypting a message of the form $\mathsf{Crypt}(\mathsf{pubK}\,A)X$ indicates A's presence: prove that if A never performs the step in which that message is decrypted, then some item in X remains secret. Later, infer that A has performed that step by taking a contrapositive.

This roundabout procedure is necessary because the mere act of decryption gives weaker guarantees than exhibiting a signed message. Consider the following protocol:

$$\begin{array}{ll}
1. & A \to B : Na, A \\
2. & B \to A : \{\!|Na|\!\}_{Kb^{-1}}, Nb \\
3. & A \to B : \{\!|Nb|\!\}_{Ka^{-1}}
\end{array}$$

The nonces are broadcast to the world, but the signatures obviously assure A and B of the other's presence.

6 A Glimpse at the Machine Proofs

To give an impression of the Isabelle formalization, Figure 2 presents the theorems providing guarantees for A. They are numbered as follows.

[3] The contrapositive of $\neg P \longrightarrow \neg Q$ is $Q \longrightarrow P$.

```
1  [| Nonce NA ∉ analz (sees lost Spy evs);
      Crypt (pubK B) {|Nonce NA, Agent A|}
        ∈ parts (sees lost Spy evs);
      evs ∈ ns_public |]
   ⟹ Crypt (pubK C) {|NA', Nonce NA|}
          ∉ parts (sees lost Spy evs)

2  [| Nonce NA ∉ analz (sees lost Spy evs);  evs ∈ ns_public |]
   ⟹ ∃A' B'. ∀A B.
            Crypt (pubK B) {|Nonce NA, Agent A|}
              ∈ parts (sees lost Spy evs)
            ⟶ A=A' & B=B'

3  [| Crypt(pubK B)  {|Nonce NA, Agent A|}
        ∈ parts(sees lost Spy evs);
      Crypt(pubK B') {|Nonce NA, Agent A'|}
        ∈  parts(sees lost Spy evs);
      Nonce NA ∉ analz (sees lost Spy evs);
      evs ∈ ns_public |]
  ⟹ A=A' & B=B'

4  [| Says A B (Crypt (pubK B) {|Nonce NA, Agent A|})
        ∈ set_of_list evs;
      A ∉ lost;  B ∉ lost;  evs ∈ ns_public |]
   ⟹ Nonce NA ∉ analz (sees lost Spy evs)

5  [| Says A  B (Crypt (pubK B) {|Nonce NA, Agent A|})
        ∈ set_of_list evs;
      Says B' A (Crypt (pubK A) {|Nonce NA, Nonce NB|})
        ∈ set_of_list evs;
      A ∉ lost;  B ∉ lost;  evs ∈ ns_public |]
  ⟹ Says B A (Crypt(pubK A) {|Nonce NA, Nonce NB|})
        ∈ set_of_list evs
```

Figure 2: The Guarantees for A in Isabelle/HOL Notation

1. This unicity lemma states that Na (if secret) is not also used as Nb. It is proved by induction.

2. This unicity lemma states that, if Na is secret, then its appearance in any instance of message 1 determines the other components. It too follows by induction, with a standard proof script.

3. This corollary of the previous lemma has a trivial proof.

   These unicity lemmas refer to the presence of encrypted messages anywhere in past traffic. The remaining theorems refer to events of the form $\mathsf{Says}\, A\, B\, X$ involving such encrypted messages.

4. This crucial theorem guarantees the secrecy of Na. The conditions $A \notin \mathit{lost}$ and $B \notin \mathit{lost}$ express that both A and B are uncompromised. The proof is by induction; it relies on the previous three lemmas, which assume the secrecy of Na as an induction hypothesis.

5. This theorem is A's final guarantee. If A has used Na to start a run with B and receives the message $\mathsf{Crypt}(\mathsf{pubK}\, A)\{\!|Na, Nb|\!\}$, then B has sent that message. It is subject to both agents' being uncompromised. The proof is by induction and relies on the secrecy and unicity of Na.

The proof script for all five theorems comprises 35 commands (tactic invocations) and executes in 52 seconds. That is seven commands and 10.4 seconds per theorem.[4]

What about the guarantees for B? Attempting to prove the secrecy of Nb leads to a subgoal that appears to have no proof.

```
[| A∉lost;  B∉lost;  C∈lost;  A≠C;
   evs∈ns_public;
   Says A  C (Crypt (pubK C) {|Nonce NA, Agent A|})
     ∈set_of_list evs;
   Says B' A (Crypt (pubK A) {|Nonce NA, Nonce NB|})
     ∈set_of_list evs;
   Says B  A (Crypt (pubK A) {|Nonce NA, Nonce NB|})
     ∈set_of_list evs;
   Nonce NB∉analz (sees lost Spy evs) |]
==> False
```

This situation might arise when the last event is an instance of step 3. The assumptions state that A and B are uncompromised and that evs is a trace. Agent A has used Na to start a run with a compromised agent, C; somebody has sent the message $\mathsf{Crypt}(\mathsf{pubK}\, A)\{\!|Na, Nb|\!\}$. We must show that these circumstances are contradictory, since the conclusion is just False. The conclusion is the simplified form of the claim that Nb remains secret even after A has sent the step 3 message $\mathsf{Crypt}(\mathsf{pubK}\, C)\{\!|Nb|\!\}$, but this message reveals Nb to the spy.

[4]Runtimes were measured on a Sun SuperSPARC model 61.

Such proof states can be hard to interpret. Does the induction formula require strengthening? Must additional lemmas be proved? But, in this case, we easily recognize Lowe's attack. The assumptions describe events that could actually occur: Nb need not remain secret.

7 Analyzing the Strengthened Protocol

Lowe [4] suggests improving the Needham-Schroeder protocol by adding explicitness. In step 2, agent B includes his identity:

$$\begin{array}{ll} 1. & A \to B : \{\!|Na, A|\!\}_{Kb} \\ 2. & B \to A : \{\!|Na, Nb, B|\!\}_{Ka} \\ 3. & A \to B : \{\!|Nb|\!\}_{Kb} \end{array}$$

The previous proof scripts, by and large, still work for this version. Thanks to Isabelle's high level of automation, minor changes such as that above seldom interfere with existing proofs. The guarantees for A are proved precisely as before, and with almost no human effort.

In proving guarantees for B, we naturally seek to strengthen them. The unicity property for Nb states that, if Nb is secret, then its presence in step 2 uniquely determines all other message components (recall §5). Step 2 now has the form

$$\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb, B|\!\}.$$

It determines not only A and Na, but also B. This additional fact lets us prove the secrecy of Nb. Recall the subgoal presented in §6. With the new version of the protocol, somebody has sent the message

$$\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb, C|\!\}.$$

Also, agent B has sent the message

$$\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|Na, Nb, B|\!\}.$$

The unicity theorem for Nb implies $B = C$, a contradiction because C is compromised and B is not.

8 Comparison with Related Work

There have been at least four previous analyses of the Needham-Schroeder public-key protocol.

Burrows et al. [2] analyze the protocol using the BAN logic. They consider the full version with seven message steps, and identify a flaw in the distribution of public keys. As always with the BAN logic, the proofs are short and easy to follow. But, it is not entirely clear what is proved: secrecy lies outside the logic's scope, and yet the proofs contain (incorrect!) assertions that the two nonces remain secret.

Lowe's work is impressive. Not only did he find a quite subtle flaw, but he demonstrated how the model checker FDR could find it [4]. I have found his paper useful in developing the Isabelle model. Lowe has since applied FDR to other protocols [5]. Model checking is fast and automatic, but can only deal with finitely many states. It can find attacks but cannot prove their absence.

Meadows's paper [6] makes direct comparisons with Lowe's. She sets up protocols corresponding precisely to Lowe's and discusses differences in speed between the NRL Protocol Analyzer and FDR (the latter is faster). She investigates many other variations on the protocol, including versions that handle the distribution of public keys. She considers the possibility of nonces being compromised. Not all of these attempts are successful: sometimes the state space becomes too large.

The Protocol Analyzer performs an unusual combination of search and proof. Ostensibly based on brute-force state enumeration, it can also prove by induction that given sets of states are unreachable. It thereby copes with infinite sets of states; a full analysis carries the same assurance as a formal proof. The precise relationship between Meadows's and my uses of induction needs to be examined.

Schneider [10], like Lowe, bases his work on the process calculus CSP [3]. But instead of using a model checker, he applies the laws of CSP in proofs. He has published detailed hand analyses of both the original protocol and Lowe's version. He considers a number of authentication properties in increasingly general settings, ultimately allowing concurrent runs.

A notable feature of Schneider's paper is his treatment of equational laws on messages. He shows how, if certain equations are allowed to hold (such as encryption distributing over concatenation), then attacks are possible. This is lacking in much other work, certainly my own, where all message constructors are assumed to be injective.

Equations must be precisely matched to the intended application. Encryption with K^{-1} is the inverse of encryption with K provided we use RSA [9], but this equation should not be assumed in general. Schneider assumes concatenation to be associative. It is, but in a most limited way, since each message component in a real protocol has a fixed size. The treatment of equations must be studied further: many protocols rely on equational properties of cryptosystems.

Type confusion is another matter. Meadows finds that allowing nonces to be confused with agent names allows additional attacks, a sober reminder of the need for explicitness [1]. But explicitness is cheap: for the Needham-Schroeder protocol, a mere two bits in each encrypted message prevents type confusion. Many type confusion attacks depend upon artificial assumptions;

it is not clear why they should continue to be studied. Who has a formal model of the careless implementor?

9 Conclusions

The two CSP-based methods, like mine, ultimately involve reasoning about traces. While they use CSP as the notation, I use logic and set theory. Which approach is better depends upon whether CSP turns out to be a beneficial tool or merely a burden. More investigation is needed to settle this question.

One advantage of CSP is that specifications can be analyzed automatically using a model checker, FDR. But the power of model checking can be seductive. Establishing the general case requires a formal proof. Lowe argues [4, §6] that the general case for Needham-Schroeder can be reduced to the finite case checked by FDR, but informal arguments are not convincing in such a slippery field as security. Mechanized provers do not have to be slow: for each version of the protocol, the full Isabelle proof script runs in under three minutes. The runtime exceeds FDR's but is comparable to that of the NRL Protocol Analyzer [6].

An interactive theorem prover gives its user much feedback. The cases of an inductive proof amount to a formal walk-through, covering each protocol step in turn. Even flawed protocols can be analyzed, and weak guarantees proved. When a protocol is modified, the proof scripts for the old version form the starting point for its analysis. The proofs described above took four normal working days to develop, including time spent developing general methods for reasoning about public-key protocols, but excluding polishing.

This experiment confirms the conclusions of my earlier work. Using the inductive approach, we can analyze protocols in days. The guarantees must be interpreted with caution—they describe specifications rather than implementations—but they have no restrictions for the sake of finiteness. Thanks to its simple foundations, the approach can probably be modified to cope with many other protocol types.

Acknowledgement Discussions with Andrew Gordon, Roger Needham and Kim Wagner were helpful. Giampaolo Bella, Fabio Massacci and Mark Staples scrutinized an early draft. The research was funded by the EPSRC grant GR/K77051 "Authentication Logics" and by the ESPRIT working group 21900 "Types".

References

[1] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

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

[3] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[4] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems. Second International Workshop, TACAS '96*, LNCS 1055, pages 147–166, 1996.

[5] Gavin Lowe. SPLICE/AS: A case study in using CSP to detect errors in security protocols. Technical report, Oxford University Computing Laboratory, 1996.

[6] Catherine A. Meadows. Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Computer Security — ESORICS 96*, LNCS 1146, pages 351–364. Springer, 1996.

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

[8] Lawrence C. Paulson. Proving properties of security protocols by induction. Technical Report 409, Computer Laboratory, University of Cambridge, December 1996.

[9] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[10] Steve Schneider. Using CSP for protocol analysis: the Needham-Schroeder public-key protocol. Technical Report CSD-TR-96-14, Royal Holloway, University of London, 1996.