Breaking the Needham–Schroeder Protocol

Chloe Mills, Ryan Sowa
McGill University, Montréal, Canada
`chloe.mills@mail.mcgill.ca, ryan.sowa@mail.mcgill.ca`

Abstract. In this paper, we present the Needham-Schroeder Public-Key Protocol along with an attack on the protocol presented by Lowe. We focus on verification methods used to prove the protocol insecure. In particular, we analyze verification with FDR, the inductive method, and the NRL Protocol Analyzer. We analyze the attacks found by these verification tools and reason about why they show insecurity of the Needham-Shroeder protocol.

1 Introduction

A cryptographic protocol is a program that must satisfy certain security properties. An example of a security property is authentication [2]. Authentication mutually verifies the identity of the parties involved. This is to ensure we are indeed speaking to a desired party and not an unintended third party pretending to be someone else [6].

In this paper, we consider the Needham-Schroeder Public-Key Protocol [6], which was designed to provide mutual authentication for parties communicating on a network. Exchange of messages occurs after the parties have mutually authenticated each other in order to not disclose private information to an unintended party. However, we demonstrate that the Needham-Schroeder public-key protocol has a fatal flaw, namely, that is an imposter can impersonate another agent in order to establish a connection with a third party. In particular, the protocol fails to ensure mutual authentication. This attack is known as a man-in-the-middle attack.

2 Terminology

Public-key cryptography is a system that uses a pair of keys for the encryption scheme, a public key and a corresponding private key [1]. $PK(A)$ and $PK(B)$ denote the public keys of A and B, respectively. Likewise, $SK(A)$ and $SK(B)$ denote the private keys of A and B respectively. If A wants to start a communication session with B, he will send a message *message* encrypted with B's public key, denoted by $\{message\}_{PK(B)}$. Only B will be able to decrypt the message with their corresponding private key to uncover the original message: $\{message\}_{PK(B)SK(B)} == \{message\}$.

The protocol involves sending a nonce, a randomly-generated number only used once per run of the protocol, between two parties. Nonces can be used to prevent replay attacks, such as impersonating someone based on their past communications. N_a and N_b are the nonces denoted by A and B, respectively.

An Authentication Server, in the context of public key algorithms, sends information based on a requested principal's secret key. Each party in the communicating network has a public key and a private key, but they will need to communicate with the authentication server to obtain another network user's public key in order to communicate with them. Only the users who have verified themselves to the network will have access to the authentication server's public key.

3 The Protocol

The protocol is comprised of seven steps, which includes an authentication server S which delivers the exchange of public keys:

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{PK(B), B\}_{SK(S)}$
3. $A \rightarrow B : A.B.\{N_a, A\}_{PK(B)}$
4. $B \rightarrow S : B, A$
5. $S \rightarrow B : \{PK(A), A\}_{SK(S)}$
6. $B \rightarrow A : B.A.\{N_a, N_b\}_{PK(A)}$
7. $A \rightarrow B : A.B.\{N_b\}_{PK(B)}$

In step 1, A requests B's public key from the authentication server. In step 2, the server responds with the public key of B along with B's identity and encrypts the message with its own private key $SK(S)$. It is presumed that all users who have verified themselves to the authentication server have the public key of the server. Step 2 is encrypted to ensure the integrity of the information by letting A know that this information did indeed come from the authentication server and not from a malicious third party. Step 3 begins the communication between A and B, where A sends his nonce N_a along with his identity A to B, encrypted using B's public key, $PK(B)$. This step acts as a way to let B know that someone claiming to be A wishes to start communication. Steps 4 and 5 are similar to steps 1 and 2, except it is now B who is requesting the public key of A. Step 6 continues the communication between A and B. B replies to A with the nonce N_a that A sent in step 3, along with B's nonce N_b, encrypted with $PK(A)$. The final step, step 7, concludes the authentication protocol by A sending to B the nonce, N_b, encrypted using $PK(B)$.

We can reduce the protocol to the following three steps, with the assumption that A and B already have each other's public keys using, for example, a cache of common public keys:

1. $A \rightarrow B : A.B\{N_a, A\}_{PK(B)}$

2. $B \rightarrow A : B.A\{N_a, N_b\}_{PK(A)}$

3. $A \rightarrow B : A.B\{N_b\}_{PK(B)}$

4 Verification Techniques

The Needham–Schroeder protocol has long been considered insecure, however, a large array of methods have been employed to prove this insecurity. In this paper, we will consider verification using FDR as demonstrated by Lowe [3], using inductive definitions as demonstrated by Paulson [7], and finally using the NRL Protocol Analyzer as demonstrated by Meadows [4, 5].

5 Verification With FDR

As the Needham–Schroeder protocol was considered secure for decades, Lowe's work proving the protocol insecure using FDR [3] had an enormous impact on cryptography as we know it today.

FDR, short for Failures Divergences Refinement Checker, is a model checker which takes as input two CSP processes, a specification and an implementation, and checks whether the implementation refines the specification. In the context of the Needham-Schroeder protocol, the agents interacting in the protocol are modeled as CSP processes by FDR. Lowe used FDR to first model the intruder, then analyze the system with an intruder, and finally discover how an attack can be realized.

The intruder was modeled as a process that has the same capabilities as a real-world intruder. Namely, the intruder has the ability to:

1. Overhear and intercept messages being passed into the system

2. Decrypt messages encrypted with his own public key

3. Introduce new messages into the system

4. Replay messages that pass through the system

.

In FDR, the system takes as input a specification and an implementation and tests whether the implementation refines the specification. Lowe specifically checked whether each trace of the implementation was also a trace in the specification. This is modeled using two actors, A, the initiator, and B, the responder.

Let $I\_commit.A.B$ represent the event that the initiator commits to a session and $R\_running.A.B$ represent the event that the responder takes part in the run with the initiator. If

$$AR_0 \ \widehat{=}\ R_{running}.A.B \rightarrow I_{commit}.A.B \rightarrow AR_0$$

represents that an $I_{commit}.A.B$ should occur only after $R_{running}.A.B$ and A_1 is defined as

$$A_1 \ \widehat{=}\ \{|R_{running}.A.B, I_{commit}.A.B|\}$$

, we define AR as

$$AR \ \widehat{=}\ AR_0|||RUN(\Sigma \setminus A_1)$$

.

Similarly, let $R\_commit.A.B$ represent the event that the responder commits to a session and $I\_running.A.B$ represent the event that the initiator takes part in the run with the responder. If

$$AI_0 \ \widehat{=}\ I_{running}.A.B \to R_{commit}.A.B \to AI_0$$

represents that a $R_{commit}.A.B$ should occur only after $I_{running}.A.B$ and A_2 is defined as

$$A_2 \ \widehat{=}\ \{|I_{running}.A.B, R_{commit}.A.B|\}$$

, we define AI as

$$AI \ \widehat{=}\ AI_0|||RUN(\Sigma \setminus A_2)$$

.

For the protocol to be secure, we would need to have that $SYSTEM$ refines AR and $SYSTEM$ refines AI. However, although FDR verified that $SYSTEM$ refines AI, FDR could not verify that $SYSTEM$ refines AI. In particular, there exists a trace of the protocol which leads to $SYSTEM$ refines AI being violated.

5.1 Details of the Attack

Let A be the initiator, B be the responder, and I be the intruder. The attack on the Needham–Schroeder which will be outlined is based on the event in which B commits to a session with A even though A is not trying to establish a session with B. Obviously, if this happens, the protocol is not secure. We will examine in detail the “simplified” (by Lowe) trace that FDR discovered which violates $SYSTEM$ refines AI.

5.2 The Attack

The first message sent in the attack is Message α.1.

$$\text{Message } \alpha.1. \quad A \to I \quad : A.I.\{N_a.A\}_{PK(I)}$$

This represents A trying to establish a session with I, sending the nonce N_a as well as his identity encrypted with I's public key.
The second message sent in the attack is Message β.1.

$$\text{Message } \beta.1. \quad I(A) \to B \quad : A.B.\{N_a.A\}_{PK(B)}$$

Upon receiving the message from A, instead of responding to A, I forwards the message to B and encrypts it with B's public key.
The third message sent is Message $\beta.2$.

$$\text{Message } \beta.2. \quad B \rightarrow I(A) \quad : B.A.\{N_a.N_b\}_{PK(A)}$$

Upon receiving the message from I, B opens the message and sees A's identity. B therefore thinks he is talking with A and sends his nonce encrypted with A's public key back to I.
The fourth message sent is Message $\alpha.2$.

$$\text{Message } \alpha.2. \quad I \rightarrow A \quad : I.A.\{N_a.N_b\}_{PK(A)}$$

Since I cannot open the message which B sent to him (it is encrypted with A's public key), I forwards the message to A, hoping A will decrypt the message and send it back to him.
The fifth message of the attack is Message $\alpha.3$.

$$\text{Message } \alpha.3. \quad A \rightarrow I \quad : A.I.\{N_b\}_{PK(I)}$$

A decrypts the message, finds his own nonce, N_b, and sends the message encrypted with I's public key back to I.
The sixth and final message of the attack is Message $\beta.3$.

$$\text{Message } \beta.3. \quad I(A) \rightarrow B \quad : A.B.\{N_b\}_{PK(B)}$$

I decrypts the message and obtains N_b, which he returns to B. Now, B believes that he is talking with A when he is actually talking to I, a clear violation of security.

5.3 Corrected Protocol Proposed by Lowe

We now analyze the corrected protocol proposed by Lowe [3] on which FDR fails to find any attacks. The corrected protocol proposes that we include the identity of the responder, B, in the response to a message. The response to the message, therefore, becomes

$$B \rightarrow A : B.A.\{N_a.N_b.B\}_{PK(A)}$$

. Thus, Message $\beta.2$ becomes

$$\text{Message } \beta.2. \quad B \rightarrow I(A) \quad : B.A.\{N_a.N_b.B\}_{PK(A)}$$

. This prevents an attack as described above, since I cannot replay this message in Message $\alpha.2$ as A is not expecting a message with B's identity.

6 Verification Using Inductive Definitions

We next analyze verification of the Needham-Shoeder protocol using inductive definitions [7]. The "inductive method" uses inductive definitions to list the possible actions that an agent or system can perform and corresponding induction rules to reason about arbitrary finite sequences of actions. In the case of the Needham-Shoreder protocol (as well as other security protocols), the inductive method must specify the capabilities of the attacker. Then, the inductive method models the behaviour of honest agents operating the midst of an attacker.

There are, of course, downsides to performing inductive verification of protocols. In particular, inductive verification of protocols involves long proofs in which each safety property is proved by induction over the protocol. We will be primarily analyzing machine proofs produced by Isabelle, an instantiation of the generic theorem-prover Isabelle [7].

As with FDR, we case analyze guarantees for A, the initiator, and for B, the responder. If we can both ensure guarantees for A and guarantees for B, then we can claim that the protocol is secure.

6.1 Guarantees for A

The guarantees for A is that if A is communicating with B and a spy is present, this spy is not able to discover A's nonce. A proof of this would require unicity properties (stated in the form of lemmas or theorems) for N_a saying that N_a is only used once. In particular, to ensure guarantees for A, the theorems in Fig.1 (formulated in Isabelle/HOL) would need to be proven. A brief overview of these theorems is presented below:

1. N_a is not also used as N_b.
2. If N_a is secret, then its presence in a message 1 determines message 1.
3. Corollary of 2.
4. N_a must be secret.
5. If A has started a run of the protocol with B and receives a message, then that message must be from B.

In Isabelle, a proof for all five of these theorems comprises 27 commands and executes in ten seconds, or equivalently, two seconds per theorem.

6.2 Guarantees for B

Similar as the situation for guarantees of A, the guarantees for B is that if B is communicating with A and a spy is present, this spy is not able to discover B's nonce. However, in this situation, N_b does not remain secret. In particular, an attempt to prove the secrecy of N_b in Isabelle/HOL leads to a subgoal which has no proof as illustrated in Fig. 2, which

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

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

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

4  [| Says A B (Crypt (pubK B) {|Nonce NA, Agent A|}) ∈ set evs;
      A ∉ bad;  B ∉ bad;  evs ∈ ns_public |]
   ⟹ Nonce NA ∉ analz(spies evs)

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

Figure 1: Lemmas for guarantees for A formulated in Isabelle

illustrates that if A is communicating with B and C, then the claim that N_b remains secret even after A has sent the message Crypt(pubK A)$\{N_b\}$ is false, since the message reveals N_b to the spy.

6.3 Analyzing Lowe's Corrected Protocol

In the strengthed protocol proposed by Lowe [3], the identity of the responder, B, is included in the response to a message. Using the inductive method, we are able to prove the secrecy of N_b. In particular, we can reason that if someone sent the message

$$\text{Cryp(pubK A)}\{N_a, N_b, C\}$$

and B sent the message

$$\text{Cryp(pubK A)}\{N_a, N_b, B\}$$

, then the unicity theorem for N_b implies that $B = C$. However, this is a contradiction, since C is compromised and B is not. Therefore, the subgoal outlined in Fig. 2 yields a contradiction with the strengthed protocol and, thus, does not return False.

```
[| A∉bad;  B∉bad;  C∈bad;
   evs3∈ns_public;
   Says A  C (Crypt (pubK C) {|Nonce NA, Agent A|})  ∈ set evs3;
   Says B' A (Crypt (pubK A) {|Nonce NA, Nonce NB|}) ∈ set evs3;
   Says B  A (Crypt (pubK A) {|Nonce NA, Nonce NB|}) ∈ set evs3;
   Nonce NB∉analz (spies evs3) |]
==> False
```

Figure 2: A failed attempt to prove the secrecy of N_b in Isabelle

7 The NRL Protocol Analyzer

The NRL Protocol Analyzer [4, 5] is a verification tool used for analysis of cryptographic protocols. The assumption is that agents communicate over a hostile network, where an intruder can read, modify and destroy message traffic. However, there are some things that the intruder does not know, such as encrypted messages that the intruder does not have the corresponding private key to decrypt. Protocols of the NRL Protocol Analyzer are specified as transitions of state machines. The state transitions are actions of the legitimate agents which are specified by the user of the Analyzer. Messages received by the legitimate agents are input to the transitions as well as values of the local state variables. We assume that the former has been either generated by an intruder or forwarded from the intruder. The output of the transitions are new local state variables and messages from the legitimate agents.

In a specification, the local state variables as well as the transitions, are indexed by the following: the name of the agent involved, an identifier for the local execution, the current protocol step, and the agent's local time.

A key difference between the NRL Protocol Analyzer and other model checkers is that the Protocol Analyzer works backwards from a final state. The analyzer is used to prove a security property by stating an insecure (final) state by

1. words that could be known to the intruder

2. values of state variables

3. events (ones that have or have not occurred)

. The analyzer then returns a description of all immediate preceding states. This process is continuously repeated on each of those states. Note that NRL deals with possibly infinite state space, where it then exhaustively searches the space. The user can adjust the search space to account for that concern. The techniques that Meadows discusses to narrow the search space are: an inductive proof of unreachability of infinite classes of states by use of formal languages, remembering conditions on reachability of states, and querying subsets of state descriptions [4].

The user begins by using the tool to generate lemmas that identify a number of unreachable states. Then, the user specifies an insecure final state for the analyzer to perform a backwards search. Each encountered state is tested against the lemmas described above in order to determine if the state is unreachable. If the state satisfies one of the lemmas, the analyzer discards that state. If the state does not satisfy any of the lemmas, the Analyzer keeps the state and then attempts to determine how the state was reached. The Analyzer records all paths previously generated and returns to the user which paths begin in initial states or if certain paths begin in unreachable ones.

In addition to the attacks Lowe presented, the NRL Protocol Analyzer was able to find new attacks by assuming that nonces can become compromised by confusion between identities and nonces.

7.1 Attacks With Compromised Nonces and Use of an Authentication Server

We now analyze the NRL Protocol Analyzer by considering the possibility that previous nonces could become compromised and the possibility that a nonce can be confused for a name (or identity). At the very least, this shows which steps of the protocol require unambigious types as being a requirement for security. This is why nonces *can* be used to prevent replay attacks, but the use of nonces does not imply a protocol is immune to replay attacks. A transition was included, which states that nonces can be compromised after it is created. Therefore, we assume the intruder can have access to newly created nonces.

Meadows presents the following attack on the Needham-Schroeder Public-Key Protocol found by the NRL Protocol Analyzer. Note that the first six steps are identical to the original Needham-Schroeder public-key protocol using an authentication server, as presented earlier.

6. $B \to A : B.A.\{N_a, N_b\}_{PK(A)}$

   I intercepts this message and sends it to A, pretending his identity is N_b in order to begin the first authentication step of the protocol with A as some other party. This is where the second run of the protocol begins.

3.′ $I(N_b) \to A : I.A.\{N_a, N_b\}_{PK(A)}$

   A requests the identity N_b from the authentication server S to get the public key of N_b.

4.′ $A \to S : A, N_b$

   I then receives N_b from A and is now able to send it to B impersonating A. This causes B to think A successfully completed the protocol.

7. $I(A) \to B : A.B.\{N_b\}_{PK(B)}$

This attack outlines how the first run of the protocol was completed by an intruder. Thus, the protocol did not ensure mutual authentication as A did not complete the last step. If these nonces are used as authenticators during a normal message exchange, I can continue to impersonate A to B.

We also present the following attack presented by Meadows which shows an agent acting as both the initiator and the responder, believing he successfully responded to himself:

1. $A \rightarrow S : A, A$

2. $S \rightarrow A : \{PK(A), A\}_{SK(S)}$

3. $A \rightarrow A : A.A.\{N_a, A\}_{PK(A)}$

   The intruder I intercepts this message and sends the following message to the responder A, thus impersonating A as the initiator.

6. $I(A) \rightarrow A : A.A.\{N_a, A\}_{PK(A)}$

   A believes that he has successfully responded to himself and assumes the identity A is actually a nonce. He sends the following message to himself.

7. $A \rightarrow A : A.A.\{A\}_{PK(A)}$

Although these attacks may appear to be odd since they assume there is confusion between identities and nonces (which can be avoided), they still offer some interesting ideas in terms of security. When an assumption is changed, the protocol can be shown to be insecure. Note that the NRL Protocol Analyzer did not find any attacks on Lowe's corrected protocol, even with the assumption that nonces and identity could be confused.

8 Conclusion

In this paper, we analyzed the Needham-Shroeder protocol, a tool used to provide authentication for parties communicating across a network. We first introduced the protocol, outlining the steps which comprise the protocol. Then, we analyzed an attack presented by Lowe as well as several verification methods for proving insecurity of the Needham-Shroeder protocol. First we analyzed verification using FDR, a model checker which models actors as CSP processes. Then, we analyzed verification using inductive definitions where we analyzed guarantees for parties A and B with the help of machine proofs produced by Isabelle. Finally, we explored attacks found by using the NRL Protocol Analyzer. Notably, these attacks have different assumptions about nonces, such as assuming there is confusion between identities and nonces. Although we can ensure that types can be distinguished rather easily, it is still useful to see what happens when we introduce the above assumption. When Needham and Schroeder proved that their protocol was correct, they assumed that all participating agents in the protocol would ensure the nonces are kept secret. This is why the attack Lowe discovered, which requires that the intruder not keep their nonce secret [4], was a counterexample to their proof.

References

[1] Martin E Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 40(5):42–49, 2002.

[2] Steve Kremer. Formal Verification of Cryptographic Protocols. Invited tutorial, 7th School on Modelling and Verifying Parallel Processes (MOVEP'06), Bordeaux, France, June 2006. 5 pages.

[3] Gavin Lowe. Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[4] Catherine Meadows. Analyzing the Needham-Schroeder Public-Key Protocol: A Comparison of Two Approaches. In *Proceedings of the 4th European Symposium on Research in Computer Security: Computer Security*, ESORICS '96, page 351–364, Berlin, Heidelberg, 1996. Springer-Verlag.

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

[6] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM*, 21(12):993–999, dec 1978.

[7] Lawrence C Paulson. The inductive approach to verifying cryptographic protocols. *Journal of computer security*, 6(1-2):85–128, 1998.