An Attack on the Needham-Schroeder Public-Key Authentication Protocol

Gavin Lowe

August 22, 1995

Abstract

In this paper we present an attack upon the Needham-Schroeder public-key authentication protocol. The attack allows an intruder to impersonate another agent.

Keywords: distributed systems; security in digital systems; authentication protocols; public-key cryptography.

1 Introduction

In a distributed computer system, it is necessary to have some mechanism whereby a pair of agents can be assured of each other's identity—they should become sure that they really are talking to each other, rather than to an imposter impersonating the other agent. This is the role of an *authentication protocol.*

In this paper we consider the Needham-Schroeder public-key authentication protocol [NS78]. The protocol aims to provide mutual authentication, after which some session involving the exchange of messages can take place. However, we show that it fails to ensure authentication: we show that an intruder can impersonate an agent A during a run of the protocol, to trick another agent B into thinking that he really is talking to A.

The protocol uses *public key cryptography* [DH76, RSA78]. Each agent A possesses a *public key*, denoted K_a, which any other agent can obtain from a key server. It also possesses a *secret key*, K_a^{-1}, which is the inverse of K_a. We will write $\{m\}_k$ for message m encrypted with key k. Any agent can encrypt a message m using A's public key to produce $\{m\}_{K_a}$; only A can decrypt this message, so this ensures secrecy. A can sign a message m by encrypting it with its secret key, to produce $\{m\}_{K_a^{-1}}$; any other agent in possession of A's public key can then decrypt this message; the encryption using A's secret key should assure other agents that the message really did originate from A.

The protocol also uses *nonces*: random numbers generated with the purpose of being used in a *single* run of the protocol. We denote nonces by N_a and N_b: the subscripts are intended to denote that the nonces were generated by A and B, respectively.

2 The Needham-Schroeder public-key authentication protocol

The Needham-Schroeder public-key protocol involves seven steps, and can be described as follows:

$$\begin{array}{lrcl}
1. & A \rightarrow S & : & A, B \\
2. & S \rightarrow A & : & \{K_b, B\}_{K_s^{-1}} \\
3. & A \rightarrow B & : & \{N_a, A\}_{K_b} \\
4. & B \rightarrow S & : & B, A \\
5. & S \rightarrow B & : & \{K_a, A\}_{K_s^{-1}} \\
6. & B \rightarrow A & : & \{N_a, N_b\}_{K_a} \\
7. & A \rightarrow B & : & \{N_b\}_{K_b}.
\end{array}$$

Here A is in *initiator* who seeks to establish a session with *responder* B, with the help of trusted key server S. In step 1, A sends a message to the server, requesting B's public key. S responds in message 2 by returning the key K_b, along with B's identity (to prevent attacks based upon diverting key deliveries), encrypted using S's secret key (to assure A that this message originated from S). A then seeks to establish a connection with B by selecting a nonce N_a, and sending it along with its identity to B (message 3), encrypted using B's public key. When B receives this message, it decrypts the message to obtain the nonce N_a. It requests (message 4) and receives (message 5) A's public key. It then returns the nonce N_a, along with a new nonce N_b, to A, encrypted with A's key (message 6). When A receives this message he should be assured that he is talking to B, since only B should be able to decrypt message 3 to obtain N_a. A then returns the nonce N_b to B, encrypted with B's key. When B receives this message he should be assured that he is talking to A, since only A should be able to decrypt message 6 to obtain N_b.

This protocol can be considered as the interleaving of two logically disjoint protocols: messages 1, 2, 4 and 5 are concerned with obtaining public keys, whereas messages 3, 6 and 7 are concerned with the authentication of A and B.

Denning and Sacco [DS81] have pointed out that this protocol provides no guarantee that the public keys obtained are current, rather than replays of old, possibly compromised keys. This problem can be overcome in various ways, for example by including timestamps in the key deliveries.

In this paper we will assume that each agent initially has each other's public key, and restrict our attention to just the following messages:

$$\begin{array}{lrcl} 3. & A \to B & : & \{N_a, A\}_{K_b} \\ 6. & B \to A & : & \{N_a, N_b\}_{K_a} \\ 7. & A \to B & : & \{N_b\}_{K_b}. \end{array}$$

3 An attack on the protocol

We will consider how an intruder can interact with this protocol. We assume that the intruder I is a user of the computer network, and so is able to set up standard sessions with other agents, and other agents may try to set up sessions with I—indeed, the attack below starts with agent A trying to establish a session with I. We assume that the intruder can intercept any messages in the system, and introduce new messages. However, we have to make some assumptions about what sort of messages the intruder may introduce. We assume that the intruder cannot guess the value of nonces being passed in encrypted messages, unless those messages are encrypted with his own key. Thus the intruder can only produce new messages using nonces that it invented itself, or that it has previously seen and understood. It can also replay complete encrypted messages, even if it is unable to understand the contents.

The attack on the protocol allows an intruder I to impersonate another agent A to set up a false session with B. The attack involves two simultaneous runs of the protocol: in run 1, A establishes a valid session with I; in run 2, I impersonates A to establish a fake session with B. Below we write, for example, 1.3 to represent message 3 in run 1; we write $I(A)$ to represent the intruder I impersonating A:

$$\begin{array}{lrcl} 1.3. & A \to I & : & \{N_a, A\}_{K_i} \\ 2.3. & I(A) \to B & : & \{N_a, A\}_{K_b} \\ 2.6. & B \to I(A) & : & \{N_a, N_b\}_{K_a} \\ 1.6. & I \to A & : & \{N_a, N_b\}_{K_a} \\ 1.7. & A \to I & : & \{N_b\}_{K_i} \\ 2.7. & I(A) \to B & : & \{N_b\}_{K_b}. \end{array}$$

In step 1.3, A starts to establish a normal session with I, sending it a nonce N_a. In step 2.3, the intruder impersonates A to try to establish a false session with B, sending it the nonce N_a obtained in the previous message. B responds in message 2.6 by selecting a new nonce N_b, and trying to return it, along with N_a, to A. The intruder intercepts this message, but cannot decrypt it because it is encrypted with A's key. The intruder therefore seeks to use A as an oracle, by forwarding the message to A in message 1.6; note that this message is of the form expected by A in run 1 of the protocol. A decrypts the message to obtain N_b, and returns this to I in message 1.7. I can then decrypt this message to obtain N_b, which it returns to B in message 2.7, thus completing run 2 of the protocol. Hence B believes that A has correctly established a session with it.

We should consider the consequences of this attack. It has been suggested [NS78, BAN89] that because the nonces are *shared secrets*, they can be included within subsequent messages as authentication. However, after the above attack, the intruder knows the nonces, and so he may continue to impersonate A to send messages to B during the session. For example, the intruder may include the nonces within a subsequent message suggesting a session key, and B will believe that this message originated from A. Similarly, if B is a bank, then I could impersonate A to send a message such as:

$$I(A) \rightarrow B : \{N_a, N_b, \text{“Transfer £1000 from my account to } I\text{'s”}\}_{K_b}.$$

4 Conclusions

In this paper we have presented an attack on the well known Needham-Schroeder public-key authentication protocol; the attack allows an intruder to impersonate one agent in a session with another.

It is fairly easy to change the protocol so as to prevent the attack. If we include the responder's identity in message 6 of the protocol:

$$6. \quad B \rightarrow A \quad : \quad \{B, N_a, N_b\}_{K_a},$$

then step 2.6 of the attack would become

$$2.6. \quad B \rightarrow I(A) \quad : \quad \{B, N_a, N_b\}_{K_a},$$

and the intruder can not successfully replay this message in message 1.6, because A is expecting a message containing I's identity. This correction represents an instance of Principle 3 of [AN94]:

> If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

We conjecture that the revised protocol is safe against all attacks—at least, those attacks not dependent upon properties of the encryption method used. Proving this formally is the topic of current research.

References

[AN94] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. Research Report 125, Digital Equipment Corporation Systems Research Center, 1994.

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989. A preliminary version appeared as Digital Equipment Corporation Systems Research Center report No. 39, 1989.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, 1981.

[NS78] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[RSA78] 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.