

# Lecture: protocollo di Needham-Schroeder

Riccardo Torre

4 dicembre 2025

$$\begin{array}{ll} A \rightarrow S: & A, B \\ S \rightarrow A: & K_{AB} \\ B \rightarrow A: & K_{AB}, A \end{array}$$

**Assunzione di sicurezza 1** *L'avversario è capace di ascoltare tutti i messaggi in un protocollo di sicurezza.*

Dunque è necessaria la crittografia dei messaggi.

---

**Assunzione di sicurezza 2 (Perfect cryptography assumption)** *L'attaccante in generale non può rompere la crittografia.*

$$\begin{array}{llll} A \rightarrow S: & A, B & & \\ S \rightarrow A: & E_{K_{AS}}(K_{AB}), E_{K_{BS}}(K_{AB}) & & \\ A \rightarrow B: & E_{K_{BS}}(K_{AB}), A & \longrightarrow & \text{C intercetta il messaggio} \\ C \rightarrow B: & E_{K_{BS}}(K_{AB}), D & & \end{array}$$

C'è un attacco peggiore di questo. In questo caso, C è autenticato (è un insider del sistema, perché S accetta la sua richiesta).

$$\begin{array}{llll} A \rightarrow S: & A, B & \text{C intercetta il messaggio} & \\ C \rightarrow S: & A, C & & \\ S \rightarrow C: & E_{K_{AS}}(K_{AC}), E_{K_{CS}}(K_{AC}) & & \\ C \rightarrow A: & E_{K_{AS}}(K_{AC}), E_{K_{CS}}(K_{AC}) & \text{A non si accorge che C sta nel mezzo} & \\ A \rightarrow C: & E_{K_{CS}}(K_{AC}), A & & \end{array}$$

Le chiavi non sono legate crittograficamente all'identità di chi deve farne uso (**binding attack**). L'attacco appena descritto è inoltre un **man-in-the-middle** e uno **spoofing attack**.

**Assunzione di sicurezza 3** *L'avversario può essere un partecipante legittimo del protocollo (un insider), o una parte esterna (un outsider) o una combinazione delle precedenti.*

---


$$\begin{aligned}
A \rightarrow S: & \quad A, B \\
S \rightarrow A: & \quad E_{K_{AS}}(K_{AB}, B), E_{K_{BS}}(K_{AB}, A) \\
A \rightarrow B: & \quad E_{K_{BS}}(K_{AB}, A)
\end{aligned}$$

Questo fix lega crittograficamente la chiave all'entità che deve utilizzarla. Ma questo protocollo è ancora vulnerabile ad un **replay attack**.

$$\begin{aligned}
A \rightarrow S: & \quad A, B \quad \longrightarrow C \text{ intercetta la richiesta, la blocca e si finge di essere il server} \\
C \rightarrow A: & \quad E_{K_{AS}}(K_{AB}', B), E_{K_{BS}}(K_{AB}', A) \quad \longrightarrow \text{dove } K_{AB}' \text{ è una vecchia chiave di sessione intercettata da } C \text{ in una vecchia interazione tra } A \text{ e } S \\
A \rightarrow B: & \quad E_{K_{BS}}(K_{AB}', A)
\end{aligned}$$

La risposta in seguito alla richiesta di A ad S è di una vecchia interazione tra A e S, solo che A non se ne accorge perché manca questa informazione nel protocollo. Bisogna legare le richieste alle risposte con i **nonce** o **challenge-response**. Arriviamo così al protocollo di Needham-Schroeder.

---

### Il protocollo di Needham-Schroeder

$$\begin{aligned}
A \rightarrow S: & \quad A, B, N_A \\
S \rightarrow A: & \quad E_{K_{AS}}(K_{AB}, B, N_A, E_{K_{BS}}(K_{AB}, A)) \\
A \rightarrow B: & \quad E_{K_{BS}}(K_{AB}, A) \\
B \rightarrow A: & \quad E_{K_{AB}}(N_B) \quad \longrightarrow \text{viene fatto una sorta di acknowledgement} \\
A \rightarrow B: & \quad E_{K_{AB}}(N_B - 1)
\end{aligned}$$

Anche questo protocollo è vulnerabile al **replay attack** sul messaggio  $B \rightarrow A: E_{K_{AB}}(N_B)$ . A ha veramente la garanzia che la chiave è fresh, ma B no!

**Attacco al protocollo Needham-Schroeder** In questo scenario, viene indicata con  $K_{\overline{AB}}$  la chiave compromessa, che C utilizza per compiere il **replay attack** verso B.

$$\begin{aligned}
A \rightarrow S: & \quad A, B, N_A \\
S \rightarrow A: & \quad E_{K_{AS}}(K_{AB}, B, N_A, E_{K_{BS}}(K_{AB}, A)) \\
C \rightarrow B: & \quad E_{K_{\overline{BS}}}(K_{\overline{AB}}, A) \\
B \rightarrow C: & \quad E_{K_{\overline{AB}}}(N_B) \quad \longrightarrow \text{viene fatto una sorta di acknowledgement} \\
C \rightarrow B: & \quad E_{K_{\overline{AB}}}(N_B - 1)
\end{aligned}$$

B è convinto che sia stato A a mandargli la  $K_{\overline{AB}}$ . Qui il nonce  $N_B$  viene inviato dopo che è stata ricevuta la chiave! Questo non permette di concludere che la chiave ricevuta provenga da un'interazione recente!

Il problema è che B non si accorge che la chiave è vecchia perché il nonce  $N_B$  non è legato crittograficamente alla chiave  $K_{AB}$  e viene usato più come un acknowledgement che come un qualcosa per verificare la freschezza della chiave di sessione. Invece A può rendersi conto che la chiave di sessione è fresca perché il nonce che ha inviato,  $N_A$ , è legato crittograficamente alla chiave  $K_{AB}$  e non può essere modificato da un attaccante nell'ipotesi che valga l'assunzione di sicurezza 2.

### Fix a questo attacco

$$\begin{array}{ll} B \rightarrow A: & B, N_B \\ A \rightarrow S: & A, B, N_A, N_B \\ S \rightarrow A: & E_{K_{AS}}(K_{AB}, B, N_A), E_{K_{BS}}(K_{AB}, A, N_B) \\ A \rightarrow B: & E_{K_{BS}}(K_{AB}, A, N_B) \end{array}$$

In questo caso i nonce inviati da  $A$  e  $B$  vengono legati crittograficamente alla chiave e un attacco come il precedente non può avvenire.

---

**Il protocollo di Needham-Schroeder a chiave pubblica** Questa è la versione di Needham-Schroeder usando chiavi asimmetriche! L'obiettivo di Needham-Schroeder in generale è di creare una sorta di *ping autentificato*. In questo caso, l'interazione con il server viene omessa.

$$\begin{array}{ll} A \rightarrow B: & E_{K_B}(N_A, A) \\ B \rightarrow A: & E_{K_A}(N_A, N_B) \\ A \rightarrow B: & E_{K_B}(N_B) \end{array}$$

**Attacco man-in-the-middle a Needham-Schroeder a chiave pubblica**  $C$  è un insider e  $A$  non sa che è cattivo.  $A$  decide di aprire una connessione con lui.

$$\begin{array}{ll} A \rightarrow C: & E_{K_C}(N_A, A) \\ C \rightarrow B: & E_{K_B}(N_A, A) \\ B \rightarrow C: & E_{K_A}(N_A, N_B) \quad \longrightarrow \quad \begin{array}{l} C \text{ non può aprire il messaggio per leggere } N_B \\ \text{perché codificato con } K_A \end{array} \\ C \rightarrow A: & E_{K_A}(N_A, N_B) \\ A \rightarrow C: & E_{K_C}(N_B) \\ C \rightarrow B: & E_{K_B}(N_B) \end{array}$$

L'attaccante ha fatto uno spoofing attack nei confronti di Bob, spacciandosi per Alice. Inoltre questo è un **binding attack**.

### Soluzione di Lowe

$$\begin{array}{ll} A \rightarrow C: & E_{K_C}(N_A, A) \\ C \rightarrow B: & E_{K_B}(N_A, A) \\ B \rightarrow C: & E_{K_A}(N_A, N_B, B) \quad \longrightarrow \quad \begin{array}{l} B \text{ risponde con un messaggio corretto ovvero lega} \\ \text{crittograficamente il nonce da lui prodotto alla sua identità} \end{array} \\ C \rightarrow A: & E_{K_A}(N_A, N_B, B) \quad \longrightarrow \quad \text{Bob? Ma io volevo parlare con Charlie!} \end{array}$$

**! Abort del protocollo !**

Il protocollo non è sound.

**Type flaw attack** Questo è un attacco man-in-the-middle condotto da  $M$  che consiste in un **oracle attack** per fare un **type-flaw attack**.

$$\begin{array}{ll}
M \rightarrow B: & E_{K_B}(N_A, A) \\
B \rightarrow S: & B, A \\
S \rightarrow B: & E_{K_{S^{-1}}}(A, K_A) \\
B \rightarrow M: & E_{K_A}(N_A, \underbrace{\boxed{N_B, B}}_{\text{id}}) \\
M \rightarrow A: & E_{K_A}(N_A, \boxed{N_B, B}) \quad \longrightarrow \quad \text{Oracle attack + Type flaw attack!} \\
A \rightarrow M: & A, \boxed{N_B, B} \quad \longrightarrow \quad N_B, B \text{ viene mandato in chiaro!} \\
M \rightarrow B: & E_{K_B}(N_B)
\end{array}$$

L'**oracle attack** consiste nell'inoltare il messaggio ad  $A$  per farselo decifrare ( $A$  è l'unico a poterlo fare!). Il **type flaw attack** invece, consiste nello sfruttare una vulnerabilità nel protocollo, ovvero che i messaggi non sono tipati, dunque il nonce di  $B$  e l'identificatore di  $B$ , che sono una stringa di bit, vengono confusi da  $A$  per un identificativo perché  $A$  pensa che  $N_B, B$  stia aprendo il protocollo con lui, dicendo “*questo è il mio nonce, voglio parlare con B*”. Quindi  $A$  invia la richiesta di certificato in chiaro. Ma in realtà  $A$  sta inviando informazioni che non dovrebbero essere divulgate ossia il nonce di  $B$  e l'identificativo di  $B$ .