Protocolli di sicurezza storici

Il protocollo Needham-Schroder

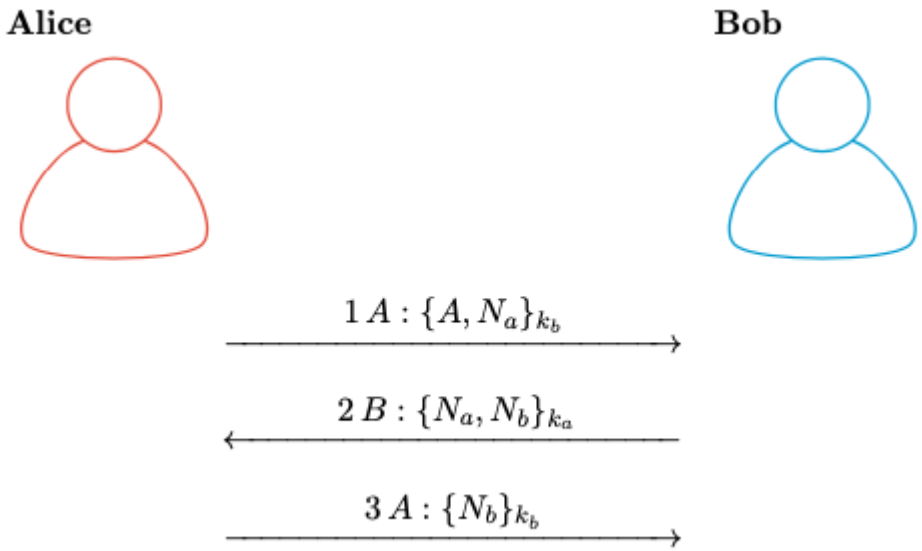
Il protocollo Needham-Schroder è basato su crittografia **asimmetrica** e su $Alice$ e Bob, i quali desiderano auteticarsi all'interno di una rete ostile Dolev-Yao. In questo scenario, esiste un infrastruttura *PKI* secondo cui tutti gli agenti conoscono le rispettive chiavi pubbliche.
Cosa aggiunge il protocollo Needham-Schroder rispetto un protocollo basilare? Il protocollo Needham-Schroder aggiunge essenzialmente due cose in più rispetto un protocollo basilare: (i) la **mutualità**, per cui $Alice$ può autenticare Bob e viceversa, e (ii) la **freshness**.

> [$Esame$] Com'è possibile aggiungere la freshness ad un protocollo basilare per l'autenticazione?

Il protocollo propone una schema *simile* al **protocollo basilare** per l'autenticazione, in cui $Alice$, per autenticare Bob, invia un messaggio a Bob cifrato con la chiave pubblica di quest'ultimo. Poiché l'unico a poter leggere il contenuto del messaggio è solo Bob (attraverso la sua chiave privata), se Bob re-invia il contenuto del messaggio ricevuto ad $Alice$, allora egli dimostra di essere effettivamente Bob.
Il protocollo, in aggiunta all'autenticazione prevede **freshness**, la quale viene implementata secondo la $Nonce$. Per questo motivo, il contenuto del messaggio di cui abbiamo scritto in precedenza diventa effettivamente la $Nonce$ generata da $Alice$.
Inviare la $Nonce$ generata da $Alice$ a Bob indica evitare un possibile *replay-attack*, mentre criptare il messaggio con la chiave pubblica di Bob funge da *challenge* ed è un modo per autenticare Bob.

> All'interno di protocollo che aggiunge freshness all'autenticazione, il secondo messaggio inviato da Bob potrebbe essere tranquillamente in chiaro. Questo non deve sorprendere poiché la $Nonce$ una *challenge-response* a cui solo l'effettivo destinatario può rispondere.

Il protocollo basilare che prevede l'aggiunta della freshness all'autenticazione potrebbe concludersi qui, con un secondo ed ultimo messaggio inviato da Bob contenente N_a.
Il protocollo Needham-Schroder però aggiunge altro, ovvero la **mutualità** nell'autenticazione tra $Alice$ e Bob.
Per garantire mutualità, è necessario che non appena Bob riceve il messaggio di $Alice$ (1), egli non invii esclusivamente N_a, bensì concateni al messaggio anche una sua $Nonce$ (i.e. N_b). A questo punto, Bob esegue l'encrypt del messaggio secondo la chiave pubblica di $Alice$ ed invia il tutto (2).
Il protocollo Needham-Schroder si conclude con un terzo ed ultimo messaggio da parte di $Alice$, in cui dimostra di essere lei inviando N_b a Bob.



Abbiamo trattato la spiegazione del protocollo Needham-Schroder a partire dalla trattazione di un protocollo basilare per l'autenticazione, poi con freshness, poi con mutualità; tuttavia, un passaggio che non abbiamo ancora spiegato al meglio è 1, lo stesso che da inizio al protocollo. All'interno del messaggio al passaggio 1, possiamo notare che oltre ad N_a, vi è anche l'identità di $Alice$ stessa.
Cosa succederebbe se all'interno del messaggio 1, non vi fosse l'identità di $Alice$?
A prima vista, potremmo pensare che l'identità di Alice posta all'interno del messaggio sia *futile* e che Bob possa conoscere il destinatario per il messaggio 2 analizzando l'intestazione del pacchetto ricevuto.
Tuttavia, non aggiungere l'identità potrebbe far sì che un attaccante $Charlie$, cambi l'intestazione del pacchetto a livello di trasporto, generando un attacco. In questo modo, infatti, Bob legge l'intestazione precedente ed invia a $Charlie$ il messaggio contenente $\{Nonce_A, Nonce_B\}$.

$$1.\ Alice \longrightarrow Bob\ :\ \{Alice, N_A\}_{k_B}$$

$$Charlie\ intercetta\ il\ messaggio$$

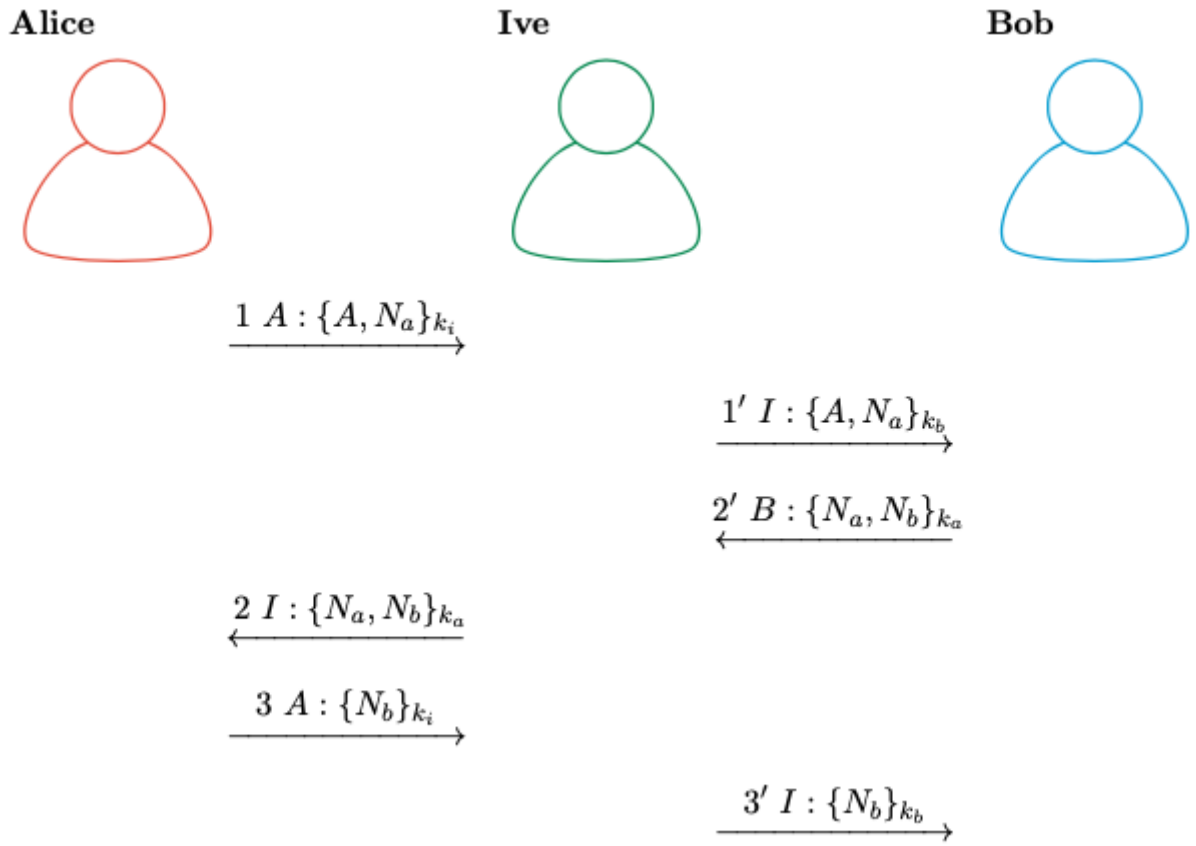
$$1^{'}.\ Charlie \longrightarrow Bob\ :\ \{Alice, N_C\}_{k_B}$$

$$2.\ Bob \longrightarrow Alice\ :\ \{N_A, N_B\}_{k_A}$$

Quali sono le conseguenze rispetto l'applicazione del protocollo? All'applicazione del protocollo, $Alice$ è in grado di autenticarsi con Bob e viceversa; inoltre, le $Nonce$ utilizzate e rimaste segrete, possono essere utilizzate per codificare i messaggi successivi tra $Alice$ e Bob.

Ad un certo punto, nel 1995, qualcuno osserva l'esistenza di almeno uno scenario di attacco, l'**attacco di Lowe**.
Ipotizziamo che $Alice$ voglia autenticarsi con Ive, motivo per cui da inizio al protocollo un messaggio contenente la sua identità e $Nonce_{Alice}$ (1).
Ive si comporta da *Dolev-Yao*, infatti, legge N_a all'interno del messaggio ricevuto da $Alice$, ed invia un messaggio Bob, inviando $\{Alice, N_a\}_{k_b}$ a Bob, applicando il protocollo NS con Bob ma impersonando $Alice$ $(1^{'})$.
A questo punto, Bob replica con un messaggio ad $Alice$ contenente la $Nonce$ di Alice (i.e. N_a) e la sua $Nonce$ (i.e. N_b), codificando la chiave pubblica di $Alice$ (i.e. k_a) $(2^{'})$.
Ive, quindi, interrompe l'invio del messaggio da parte di Bob ed intercetta il messaggio, ma non essendo in grado di rompere la cifratura non può far altro che continuare l'invio del messaggio verso $Alice$ (2).
$Alice$ riceve il messaggio da parte di Ive ed autentica Ive, notando anche la *challenge-response* generata da Bob, ovvero N_b. Si noti che al dì là della notazione utilizzata, $Alice$, in realtà, pensa che la N_b appartenga ad Ive, e non Bob.
$Alice$ continua ad eseguire il protocollo inviando ad Ive la "*sua*" $Nonce$, cifrando con la chiave di Ive (3).
A questo punto, Ive replica il messaggio ricevuto da $Alice$ a Bob e segna la fine del protocollo $(3^{'})$.

Ive, in definitiva, ottiene sia $Nonce_{Alice}$ che $Nonce_{Bob}$; in particolare, Ive si autentica con Bob impersonando $Alice$.
Si noti che la conoscenza di $Nonce_{Alice}$ da parte di Ive, non è un attacco alla confidenzialità. L'attacco avviene nel momento in cui Ive scopre $Nonce_{Bob}$, generata da Bob per $Alice$.



> Rispetto l'attacco di Lowe, la vittima diretta che desume qualcosa di erroneo è Bob, la vittima rispetto il furto d'identità è $Alice$.

Quali sono i possibili **fix** rispetto l'**attacco di Lowe**? Un possibile *fix* riguarda la modifica del protocollo stesso e prevede che all'interno del messaggio 2, oltre ad N_a ed N_b, venga inserito anche l'identità del mittente, ossia Bob. Esistono altri *fix*.

Rispetto l'attacco di Lowe, Needham arrivò ad affermare che l'attacco non avesse senso, affermando che il modello di attaccante su cui si basa l'attacco è diverso rispetto al modello di attaccante che Needham e Schroder avevano *dato per implicito* all'interno della documentazione del protocollo. D'altra parte, la nascita del protocollo è precedente a quella relativa al modello di attaccante Dolev-Yao.
Da questo evento, non esistette più un protocollo che non presentasse una sezione relativa al modello di attaccante all'interno della sua documentazione.

> [$Osservazione$] Needham affermò che il modello di attaccante (dato per implicito) prevedesse che ogni persona si fidasse dell'altra, tuttavia questa affermazione si scontra con l'utilizzo dell'identità di Alice all'interno del punto 1, la quale era necessaria affinché la Nonce di Bob non finisse in mani sbagliate.

> Per questo motivo, possiamo concludere con due diverse affermazioni: (i) la risposta data da Needham rispetto il modello di attaccante utilizzato è vera e quindi l'aggiunta di Alice è stata un no-sense (impossibile, dati gli autori), oppure (ii) il contrario, per cui il modello di

> attaccante descritto è stato effettivamente utilizzato e Needham e Schroder hanno dimenticato di aggiunere l'identità di Bob all'interno del messaggio al punto 2.

Il protocollo Woo-Lam

Il protocollo Woo-Lam è basato su crittografia **simmetrica** ed ha l'obiettivo di autenticare $Alice$ con Bob e non necessariamente viceversa. Poiché il protocollo è basato su crittografia asimmetrica, ciascun utente condivide la propria chiave simmetrica (i.e. a lungo termine), con una fortuna assunzione circa la protezione offerta da parte del server TTP (acronimo di *Trusted Third Party*).
L'utilizzo del TTP all'interno dei passaggi nel protocollo è del tutto normale, poiché laddove vi sia un crittotesto creato con una chiave a lungo termine del mittente, il ricevente non può che affidarsi al server TTP per il decrypt del messaggio.

$$1.\ A \longrightarrow B\ :\ A$$
$$2.\ B \longrightarrow A\ :\ N_b$$
$$3.\ A \longrightarrow B\ :\ \{N_b\}_{k_a}$$
$$4.\ B \longrightarrow TTP\ :\ \{Alice, \{N_b\}_{k_a}\}_{k_b}$$
$$5.\ TTP \longrightarrow B\ :\ \{N_b\}_{k_b}$$

Il passaggio (1) prevede l'invio di un messaggio contenente l'identità di $Alice$, da parte di $Alice$ verso Bob. Bob, quindi, suppone vi possa essere all'altro capo ci sia $Alice$, motivo per cui emette un challenge rappresentato una $Nonce$.
Per accettare la sfida, al passo 3, $Alice$ non può che cifrare il messaggio con la sua chiave a lungo termine. Al passo successivo (4), Bob ricorre al TTP, inviando un messaggio contenente l'identità di $Alice$ ed $\{N_b\}_{k_a}$, il tutto cifrato con la chiave a lungo termine di Bob.
Il TTP è programmato per leggere l'identità della prima parte del messaggio (i.e. $Alice$), cercare la chiave per l'identità ed effettuare il decrypt della seconda parte del messaggio utilizzando la chiave trovata nel database. Il *risultato del decrypt* viene cifrato con la chiave a lungo termine di Bob ed inviato a quest'ultimo all'interno del punto 6.
A questo punto, Bob può effettuare il decrypt del messaggio inviato dal TTP, ed autenticare $Alice$ se ciò che è contenuto nel messaggio è uguale alla $Nonce$ generata da egli al punto 2.