Protocolli di sicurezza

Alessandro Armando

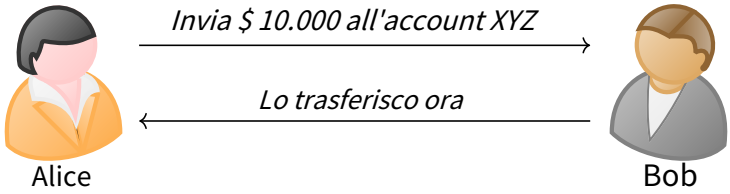
Laboratorio di sicurezza informatica (CSec)
DIBRIS, Università di Genova

Sicurezza del computer

Contorno

Motivazione

Esempio: Protezione di un'applicazione di e-banking.



- Come fa *Bob* conoscere il messaggio originato da *Alice*? Come
- fa *Bob* sapere *Alice* l'hai appena detto?

Altri esempi:

- Protezione di una rete di sensori
- Associazione di dispositivi wireless
- Un sistema di controllo accessi per impianti di risalita di tutto il
- comprensorio Sistemi di pagamento online

Come costruiresti algoritmi distribuiti per farlo?

Le soluzioni prevedono protocolli come: IPsec, SSH, PGP, SSL, Kerberos, ecc. Ci concentreremo sulle idee sottostanti.

Altri esempi:

- Protezione di una rete di sensori
- Associazione di dispositivi wireless
- Un sistema di controllo accessi per impianti di risalita di tutto il
- comprensorio Sistemi di pagamento online

Come costruiresti algoritmi distribuiti per farlo?

Le soluzioni prevedono protocolli come: IPsec, SSH, PGP, SSL, Kerberos, ecc. Ci concentreremo sulle idee sottostanti.

Altri esempi:

- Protezione di una rete di sensori
- Associazione di dispositivi wireless
- Un sistema di controllo accessi per impianti di risalita di tutto il
- comprensorio Sistemi di pagamento online

Come costruiresti algoritmi distribuiti per farlo?

Le soluzioni prevedono protocolli come: IPsec, SSH, PGP, SSL, Kerberos, ecc. Ci concentreremo sulle idee sottostanti.

Contorno

Definizioni

- UN protocollo consiste in un insieme di regole (convenzioni) che determinano lo scambio di messaggi tra due o più mandanti.
  In breve, a algoritmo distribuito con enfasi sulla comunicazione.
- Sicurezza (o crittografico) utilizzano meccanismi crittografici per raggiungere obiettivi di sicurezza.
  Esempi: Autenticazione dell'entità o del messaggio, creazione della chiave, integrità, tempestività, scambio equo, non ripudio, ...
- Piccole ricette, ma non banali da ideare e capire.
- Analogo a**programmare il computer di Satana**.[1]

[1] https://www.cl.cam.ac.uk/~rja14/Papers/satan.pdf

Messaggi

- I costruttori di messaggi sono:

  Nomi: UN, B o *Alice*, *Bob*, ...

  chiavi: K e chiavi inverse K^{-1}

  Tasti simmetrici: $\{M\}_{K_{UN\ B}}$, dove K_{AB} è condiviso tra (cioè noto solo a) UN e B

  Crittografia: $\{M\}_K$, ad es. crittografia con UNa chiave pubblica di: $\{M\}_{K_{UN}}$

  Firma: $\{M\}_{K^{-1}}$, ad es. "firma" con UNa chiave privata di: $\{M\}_{K^{-1}_{UN}}$

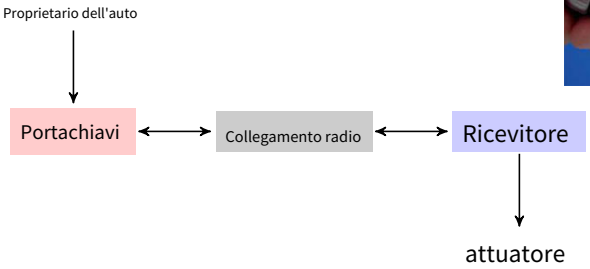
  Nonce: n_{UN}, nuovi elementi di dati utilizzati per la sfida/risposta.**Timestamp:**

  T, denotare il tempo, ad esempio, utilizzato per la scadenza della chiave.

  Concatenazione dei messaggi: $\{m_1, m_2\}$, $m_1 \mathbin{\|} m_2$, o $[m_1, m_2]$.

- Esempio: $\{A_{UN}, K_{AB}\}_{K_B}$

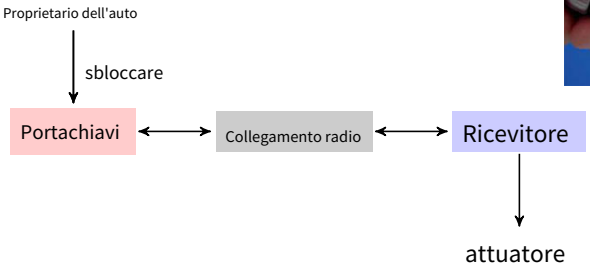
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

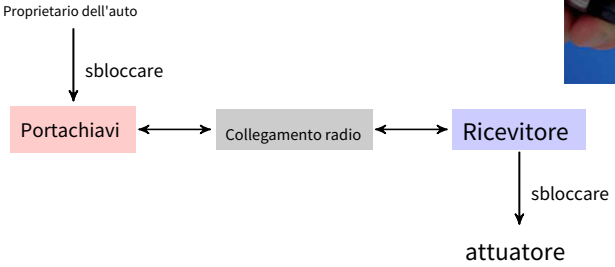
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

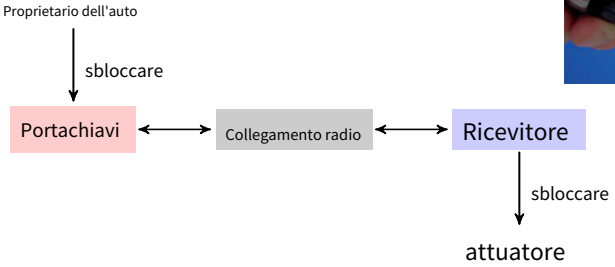
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

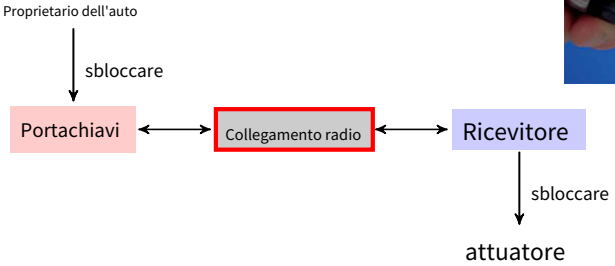
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

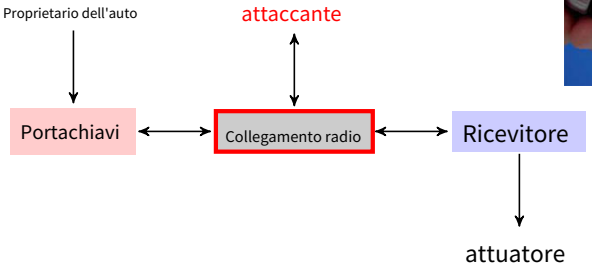
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

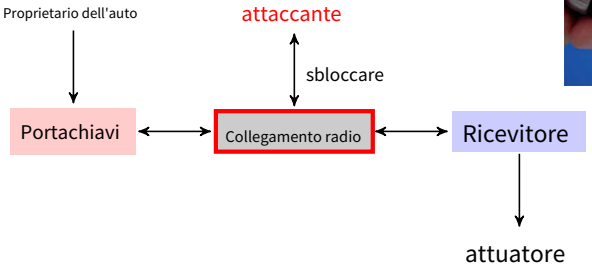
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

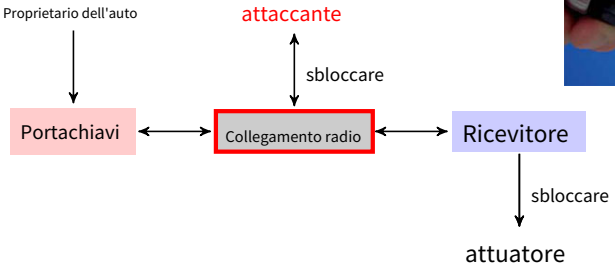
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

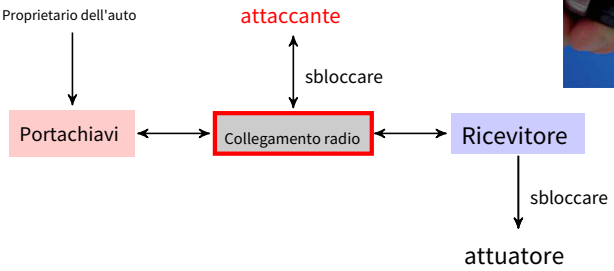
Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

Esempio: sistema remoto senza chiave



Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.

Protocollo Remote Keyless System (1° tentativo)

Assumi numero di serie (SN)
è un segreto condiviso tra KF e R.

KF invia SN a R:

1. $KF \rightarrow R$: *sbloccare, SN*

- Cattiva idea: l'attaccante può facilmente origliare SN e *rigiocare* esso successivamente. I
- problemi:
  - Segretezza di SN compromessa.
  - R non può verificare l'autenticità della richiesta

Protocollo Remote Keyless System (1° tentativo)

Assumi numero di serie (SN)
è un segreto condiviso tra KF e R.

KF invia SN a R:

1. $KF \rightarrow R$: *sbloccare, SN*

- Cattiva idea: l'attaccante può facilmente origliare SN e *rigiocare* esso successivamente.
- problemi:
  - Segretezza di SN compromessa.
  - R non può verificare l'autenticità della richiesta

Protocollo Remote Keyless System (1° tentativo)

Assumi numero di serie (SN)
è un segreto condiviso tra KF e R.

KF invia SN a R:

1. $KF \rightarrow R$: *sbloccare, SN*

- Cattiva idea: l'attaccante può facilmente origliare SN e *rigiocare* esso successivamente. I
- problemi:
  - Segretezza di SN compromessa.
  - R non può verificare l'autenticità della richiesta

Protocollo Remote Keyless System (2° tentativo)

Idea: proteggere la segretezza di SN

KF crittografa la richiesta con la chiave condivisa (K) e invia i risultati a R.

1. $KF \rightarrow R$: $\{sblocca, SN\}_K$

- L'aggressore può facilmente ascoltare la richiesta crittografata e rigiocare esso successivamente.
- Segretezza di SN assicurata ma attacco possibile comunque. R può verificare
- l'autenticità della richiesta; ma questo non basta...

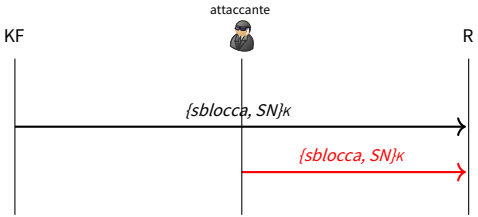
Protocollo Remote Keyless System (2° tentativo)

Idea: proteggere la segretezza di SN

KF crittografa la richiesta con la chiave condivisa (K) e invia i risultati a R.

1. $KF \rightarrow R$: $\{sblocca, SN\}_K$

- L'aggressore può facilmente ascoltare la richiesta crittografata e rigiocare esso successivamente.
- Segretezza di SN assicurata ma attacco possibile comunque. R può verificare
- l'autenticità della richiesta; ma questo non basta...



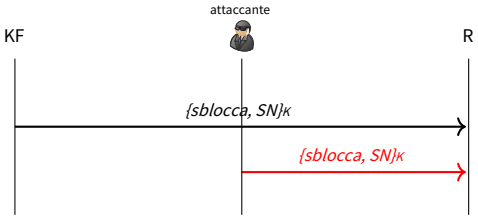
Protocollo Remote Keyless System (2° tentativo)

Idea: proteggere la segretezza di SN

KF crittografa la richiesta con la chiave condivisa (K) e invia i risultati a R.

1. $KF \rightarrow R : \{sblocca, SN\}_K$

- L'aggressore può facilmente ascoltare la richiesta crittografata e rigiocare esso successivamente.
- Segretezza di SN assicurata ma attacco possibile comunque.R può verificare
- l'autenticità della richiesta; ma questo non basta...



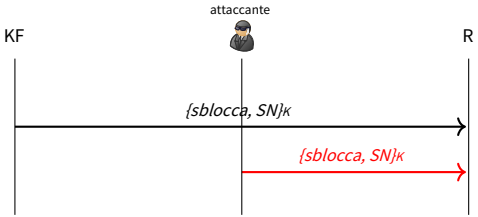
Protocollo Remote Keyless System (2° tentativo)

Idea: proteggere la segretezza di SN

KF crittografa la richiesta con la chiave condivisa (K) e invia i risultati a R.

1. $KF \rightarrow R$: $\{sblocca, SN\}_K$

- L'aggressore può facilmente ascoltare la richiesta crittografata e rigiocare esso successivamente.
- Segretezza di SN assicurata ma attacco possibile comunque. R può verificare
- l'autenticità della richiesta; ma questo non basta...

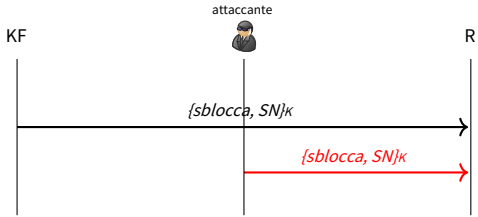


Obiettivo di sicurezza: requisito di freschezza

La proprietà:

Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.



è rispettato dal protocollo.

Eppure, il protocollo soffre di a *rigioca attacco* .

Obiettivo di sicurezza (rivisto)

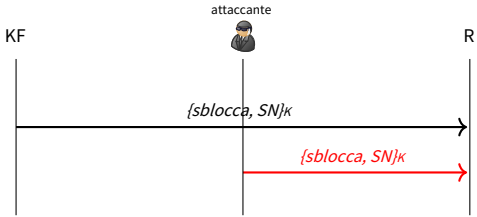
Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *recentemente* premuto il pulsante di sblocco sul portachiavi.

Obiettivo di sicurezza: requisito di freschezza

La proprietà:

Obiettivo di sicurezza (1° tentativo)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *in precedenza* premuto il pulsante di sblocco sul portachiavi.



è rispettato dal protocollo.

Eppure, il protocollo soffre di a *rigioca attacco* .

Obiettivo di sicurezza (rivisto)

Il ricevitore invia il comando di sblocco all'attuatore *solo se* Proprietario dell'auto *recentemente* premuto il pulsante di sblocco sul portachiavi.

Protocollo Remote Keyless System (3° tentativo)

KF crittografa un timestamp con la chiave condivisa K e invia il risultato a R.

1. $KF \rightarrow R$: $\{sblocca, T\}_K$

Protocollo Remote Keyless System (3° tentativo)

KF crittografa un timestamp con la chiave condivisa K e invia il risultato a R.

1. $KF \rightarrow R$: $\{sblocca, T\}_K$

- Non è necessario inviare SN poiché KF può essere identificato tramite la chiave condivisa.
- Il timestamp impedisce l'attacco di replay, ma richiede orologi sincronizzati su KF e R.

Protocollo Remote Keyless System (4° tentativo)

Il ricevitore (R) invia una chiave al telecomando (KF) (un nonce, N) e KF restituisce N crittografato con la chiave condivisa (K).

1. $KF \rightarrow R$: *Ciao2.*
$R \rightarrow KF$: n
3. $KF \rightarrow R$: $\{sblocca, N\}_K$

Protocollo Remote Keyless System (4° tentativo)

Il ricevitore (R) invia una chiave al telecomando (KF) (un nonce, N) e KF restituisce N crittografato con la chiave condivisa (K).

1. $KF \rightarrow R$: *Ciao2.*
$R \rightarrow KF$: n
3. $KF \rightarrow R$: $\{sblocca, N\}_K$

- L'uso del nonce impedisce gli attacchi di replay.
- Non sono necessari orologi sincronizzati su KF e R, ma sono necessari passaggi aggiuntivi.

Letture consigliate

- "Come funziona l'ingresso remoto", Howstuffworks.https:// auto.howstuffworks.com/remote-entry2.htm (Accesso il 25 agosto 2019)

- "Gli hacker possono rubare una Tesla Model S in pochi secondi clonando il suo portachiavi". Cablato, 2018.https://www.wired.com/story/hackers-steal-tesla-model-s-seconds-key-fob/
- "Tesla Model S rubata in 30 secondi utilizzando Keyless Hack", PC Magazine, 2019.https://www.pcmag.com/news/370359/ tesla-model-s-stolen-in-30-seconds-using-keyless-hack

Notazione del protocollo (notazione di Alice e Bob)

- Gli eventi fondamentali sono la comunicazione tra i presidi:

$$1.\ io \to R : \{ESSO_{io}, K\}_{K_R}$$
$$2.\ R \to io : \{R, io\}_K$$

- *io* (iniziatore) e *R* (risponditore) nome ruoli.
  Può essere istanziato da qualsiasi principale che gioca nel ruolo, ad es

$$1.\ un \to B : \{a_{un}, K\}_{K_B}$$
$$2.\ B \to un : \{b, a\}_K$$

$$1.\ B \to C : \{b, t_B, K_{io}\}_{K_C}$$
$$2.\ C \to B : \{c, b\}_{K_{io}}$$

- La comunicazione è asincrona
- Il protocollo specifica le azioni dei principali.
  Equivalentemente, il protocollo definisce un insieme di sequenze di eventi (tracce).

Notazione del protocollo (notazione di Alice e Bob)

- Gli eventi fondamentali sono la comunicazione tra i presidi:

$$1.\ io \rightarrow R : \{ESSO_{io}, K\}_{K_R}$$
$$2.\ R \rightarrow io : \{R, io\}_K$$

- *io* (iniziatore) e *R* (risponditore) nome ruoli.
  Può essere istanziato da qualsiasi principale che gioca nel ruolo, ad es

$$1.\ un \rightarrow B : \{a_{un}, K\}_{K_B}$$
$$2.\ B \rightarrow un : \{b, a\}_K$$

$$1.\ B \rightarrow C : \{b, t_B, K_{io}\}_{K_C}$$
$$2.\ C \rightarrow B : \{c, b\}_{K_{io}}$$

- La comunicazione è asincrona
- Il protocollo specifica le azioni dei principali.
  Equivalentemente, il protocollo definisce un insieme di sequenze di eventi (tracce).

Ipotesi e obiettivi

Ipotesi (per i mandanti):

- I principali conoscono le loro chiavi private e le chiavi pubbliche degli altri
- I principali possono generare/controllare nonce e timestamp, crittografare e decrittografare con chiavi note
- I presidi (onesti) implementano correttamente il protocollo

Obiettivi: Cosa dovrebbe ottenere il protocollo. Per esempio,

- Autenticare messaggi, legandoli al loro mittente.
- Garantire tempestività di messaggi (recenti, freschi, ...) Garanzia
- segretezza di determinati elementi (ad es. chiavi generate).

Ipotesi: attaccante

Come modelliamo l'attaccante? Possibilità:

- Conosce il protocollo ma non può violare le criptovalute. (Standard)
- egli è passivo ma ascolta tutte le comunicazioni. egli èattivo e
- può intercettare e generare messaggi. Potrebbe anche essere uno
- dei principali responsabili del protocollo!

Modello attaccante standard

- L'attaccante è attivo. Vale a dire:
  - Può intercettare e leggere tutti i messaggi.
  - Può scomporre i messaggi nelle loro parti.
    Ma la crittografia è sicura: la decrittazione richiede chiavi inverse.
  - Può costruire nuovi messaggi con i diversi costruttori. Può inviare
  - messaggi in qualsiasi momento.
- A volte chiamato il Dolev-Yao modello attaccante.
- Ipotesi più forti possibili sull'attaccante

$=?$

i protocolli corretti funzionano nella più ampia gamma di ambienti.

Tipi di attacco

- Rigiocare (o freschezza) attacco: riutilizza parti dei messaggi precedenti.
- Uomo nel mezzo (o sessioni parallele) attacco: $UN\ M \leftrightarrow B$.
- Attacco di riflessione inviare le informazioni trasmesse all'originatore.
- Tipo attacco difetto: sostituisce un diverso tipo di campo messaggio. Esempio: usa un nome (o una chiave o ...) come nonce.

Contorno

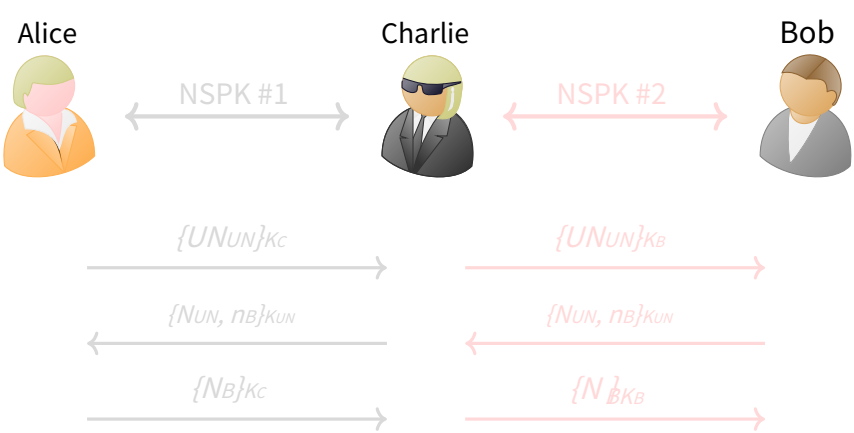
Protocollo NSPK

1. $UN \rightarrow B : \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN : \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B : \{N_B\}_{K_B}$

- **Obiettivo:** autenticazione reciproca (di entità).
- Argomento di correttezza (informale).
  1. Questa è Alice e ho scelto un nonce n_{Alice}.
  2. Ecco il tuo Nonce n_{Alice}. Dal momento che ho potuto leggerlo, devo essere Bob. Ho anche una sfida n_{Bob} per te.
  3. Tu mi hai mandato n_{Bob}. Dal momento che solo Alice può leggerlo e l'ho rimandato indietro, devo essere Alice.
- Le entità di richiamo possono essere coinvolte in più esecuzioni. L'obiettivo dovrebbe essere mantenuto in tutte le esecuzioni del protocollo interleaved.

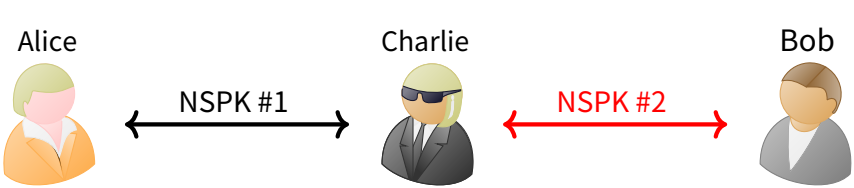
Protocollo proposto negli anni '70 e utilizzato da decenni.

Attacco a NSPK



1. $UN \rightarrow B$: $\{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN$: $\{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B$: $\{N_B\}_{K_B}$

Attacco a NSPK



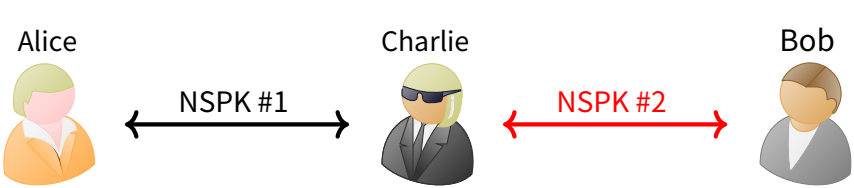
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$

$\{UN_{UN}\}_{K_C}$ $\quad$ $\{UN_{UN}\}_{K_B}$

$\{N_{UN}, n_B\}_{K_{UN}}$ $\quad$ $\{N_{UN}, n_B\}_{K_{UN}}$

$\{N_B\}_{K_C}$ $\quad$ $\{N_B\}_{K_B}$

Attacco a NSPK



1. $UN \to B: \{UN_{UN}\}_{K_B}$
2. $B \to UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \to B: \{N_B\}_{K_B}$

$\{UN_{UN}\}_{K_C}$

$\{UN_{UN}\}_{K_B}$

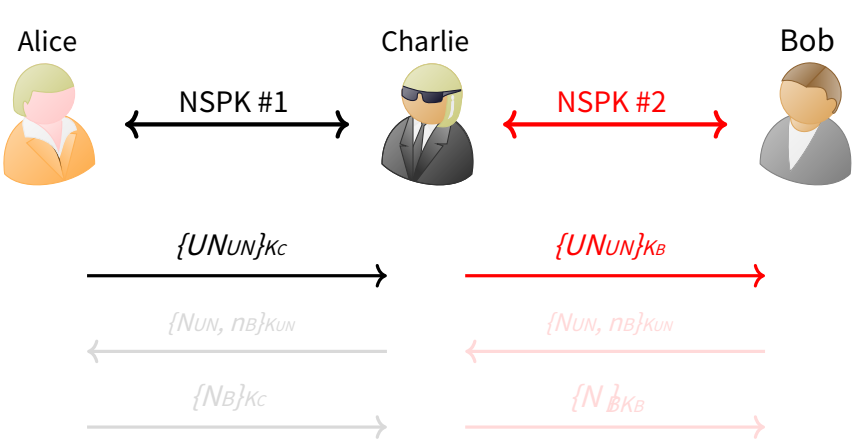
$\{N_{UN}, n_B\}_{K_{UN}}$

$\{N_{UN}, n_B\}_{K_{UN}}$

$\{N_B\}_{K_C}$

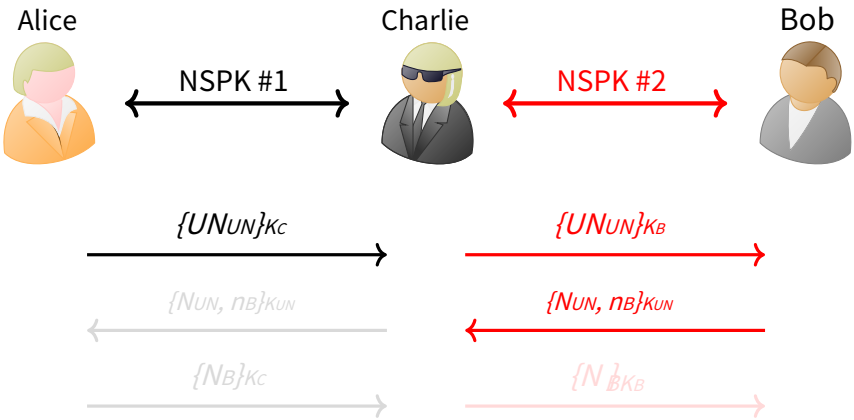
$\{N_B\}_{K_B}$

Attacco a NSPK



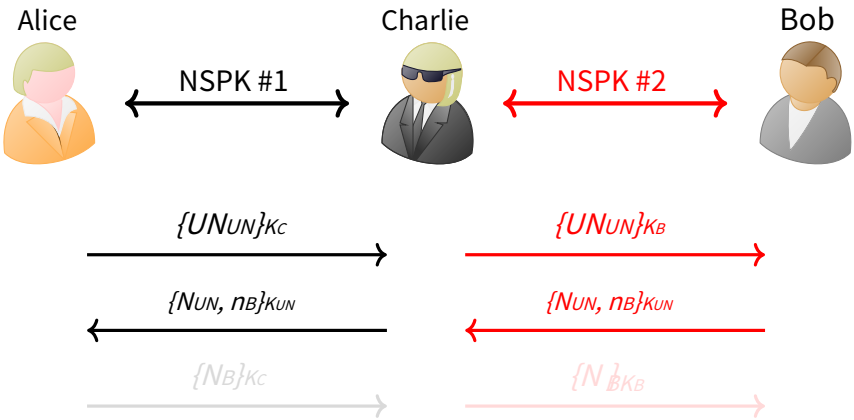
1. $UN \to B: \{UN_{UN}\}_{K_B}$
2. $B \to UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \to B: \{N_B\}_{K_B}$

Attacco a NSPK



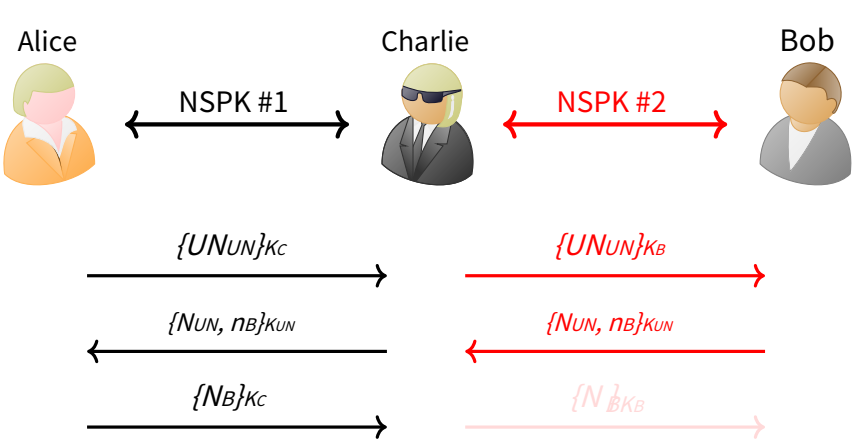
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$

Attacco a NSPK



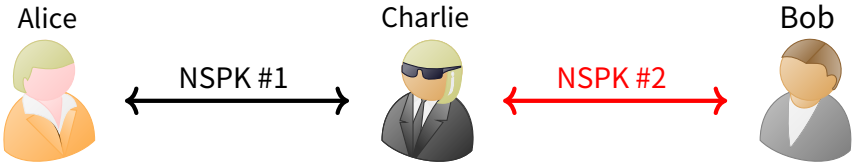
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$

Attacco a NSPK



1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$

Attacco a NSPK



1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$

B crede di parlare con *UN*!

Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

1. $UN \rightarrow B : \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN : \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B : \{N_B\}_{K_B}$

Protocollo NSPK con Fix di Lowe

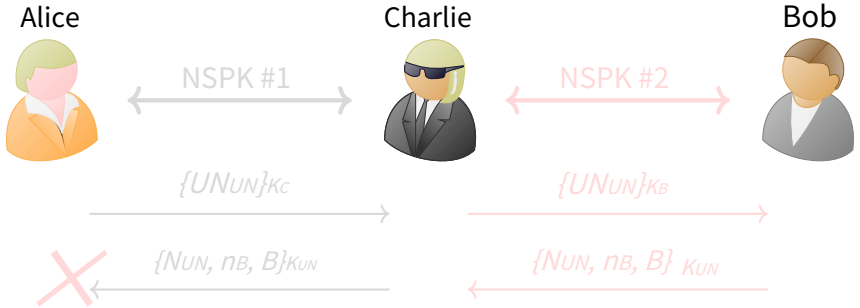
Come possiamo proteggerci dall'attacco precedente?

1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$

Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

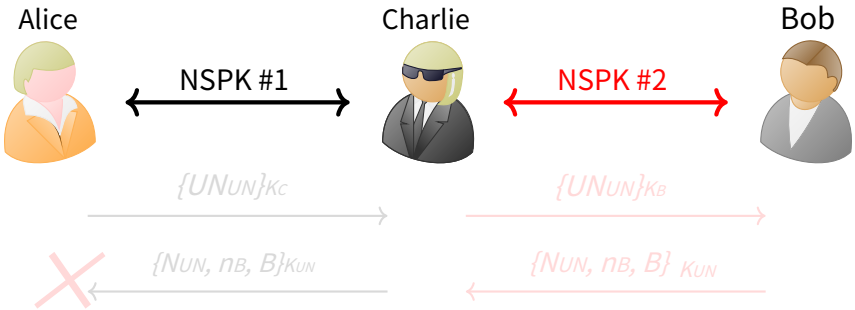
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$



Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

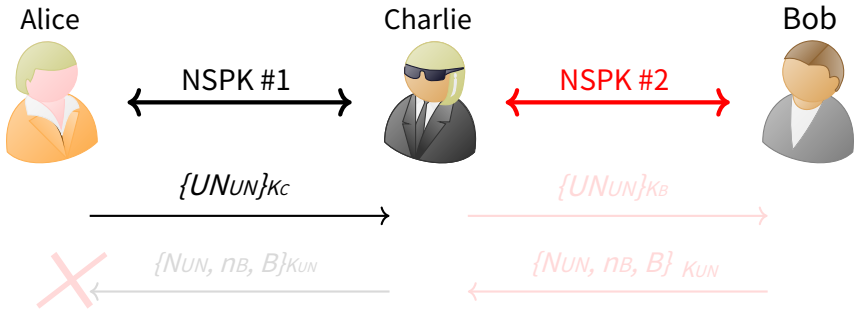
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$



Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

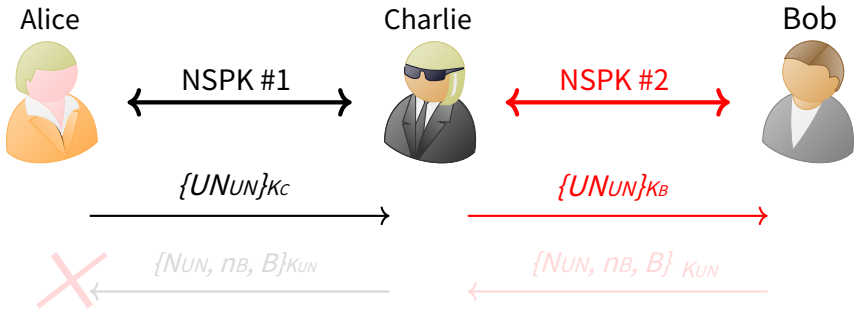
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$



Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

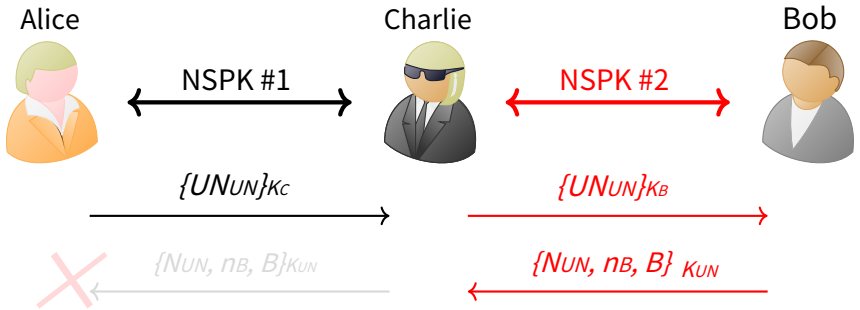
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$



Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

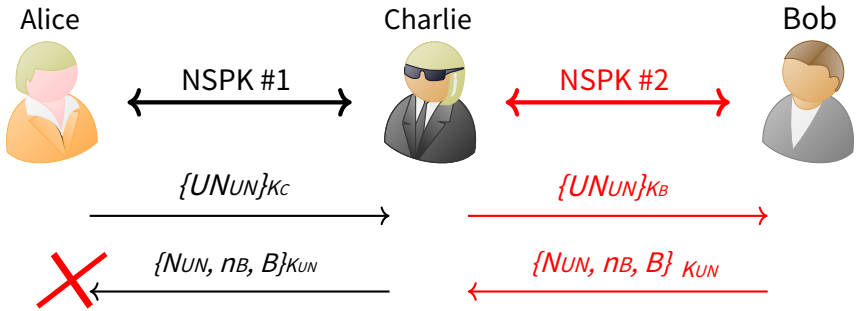
1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$



Protocollo NSPK con Fix di Lowe

Come possiamo proteggerci dall'attacco precedente?

1. $UN \rightarrow B: \{UN_{UN}\}_{K_B}$
2. $B \rightarrow UN: \{N_{UN}, n_B, B\}_{K_{UN}}$
3. $UN \rightarrow B: \{N_B\}_{K_B}$



- Un messaggio è costituito da una sequenza di sottomessaggi.
  Esempi: il nome di un preside, un nonce, una chiave, ...
- Messaggi inviati come stringhe di bit. Nessuna informazione sul tipo.

1011 0110 0010 1110 0011 0111 1010 0000

- Difetto di tipo è quando $UN \rightarrow B$: m e B accetta m come valido ma lo analizza in modo diverso. Cioè, B interpreta i bit in modo diverso rispetto a UN.
- Consideriamo un paio di esempi.

Contorno

Il protocollo Otway-Rees

Protocollo basato su server che fornisce la distribuzione della chiave autenticata (con autenticazione della chiave e aggiornamento della chiave) ma senza autenticazione dell'entità o conferma della chiave.

M1. $UN \rightarrow B$: $io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \rightarrow S$: $io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $S \rightarrow B$: $IN_{UN}, K_{AB}\}_{K_{UN\ S}}, \{n_B, K_{AB}\}_{K_{BS}}$
M4. $B \rightarrow UN$: $IN_{UN}, K_{AB}\}_{K_{COME}}$

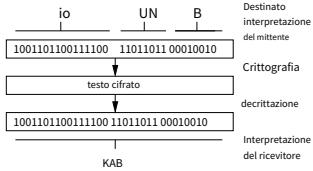
dove le chiavi del server sono già note e *io* è l'identificatore di esecuzione del protocollo (ad esempio, un numero intero).

Perché dovrebbe (non) avere le proprietà di cui sopra?

Tipo attacco difetto sul protocollo Otway-Rees

M1. $UN \rightarrow B$: $Io, A, B, \{N_{UN}, io, A, B\}K_{COME}$
M2. $B \rightarrow S$: $Io, A, B, \{N_{UN}, io, A, B\}K_{UN\ S}, \{n_B, io, A, B\}K_{BS}$
M3. $S \rightarrow B$: $IN_{UN}, K_{AB}\}K_{COME}, \{n_B, K_{AB}\}K_{BS}$
M4. $B \rightarrow UN$: $IN_{UN}, K_{AB}\}K_{COME}$

supponiamo $|\{Io, A, B\}| = |\{K_{AB}\}|$, per esempio, *io* è 32 bit, *UN* e *B* sono 16 bit, e K_{AB} è 64 bit.



Attacco 1 (Riflessione/difetto di tipo): Mallory l'attaccante riproduce parti del messaggio 1 come messaggio 4 (omettendo i passaggi 2 e 3).

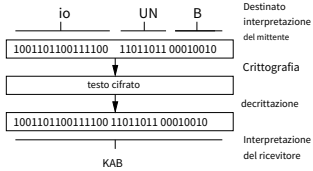
M1. $UN \rightarrow M(B)$: $Io, A, B, \{N_{UN}, io, A, B\}K_{COME}$
M4. $m(B) \rightarrow UN$: $IN_{UN}, io, A, B\}K_{COME}$

UN vede n_{UN} e accetta erroneamente $\{io, A, B\}$ come chiave di sessione.

Tipo attacco difetto sul protocollo Otway-Rees

M1. $UN \rightarrow B$: $Io, A, B, \{N_{UN}, io, A, B\}K_{COME}$
M2. $B \rightarrow S$: $Io, A, B, \{N_{UN}, io, A, B\}K_{UN\ S}, \{n_B, io, A, B\}K_{BS}$
M3. $S \rightarrow B$: $IN_{UN}, K_{AB}\}K_{COME}, \{n_B, K_{AB}\}K_{BS}$
M4. $B \rightarrow UN$: $IN_{UN}, K_{AB}\}K_{COME}$

supponiamo $|\{Io, A, B\}| = |\{K_{AB}\}|$, per esempio, *io* è 32 bit, *UN* e *B* sono 16 bit, e K_{AB} è 64 bit.



Attacco 1 (Riflessione/difetto di tipo): Mallory l'attaccante riproduce parti del messaggio 1 come messaggio 4 (omettendo i passaggi 2 e 3).

M1. $UN \rightarrow M(B)$: $Io, A, B, \{N_{UN}, io, A, B\}K_{COME}$
M4. $m(B) \rightarrow UN$: $IN_{UN}, io, A, B\}K_{COME}$

UN vede n_{UN} e accetta erroneamente $\{io, A, B\}$ come chiave di sessione.

Tipo attacco difetto sul protocollo Otway-Rees (segue)

M1. $UN \rightarrow B$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \rightarrow S$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{UN\ S}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $S \rightarrow B$: $IN_{UN}, K_{AB}\}_{K_{COME}}, \{n_B, K_{AB}\}_{K_{BS}}$
M4. $B \rightarrow UN$: $IN_{UN}, K_{AB}\}_{K_{COME}}$

Attacco 2: Mallory può interpretare il ruolo di *S* in M2 e M3 riflettendo i componenti crittografati di M2 di nuovo in *B*. Vale a dire:

M1. $UN \rightarrow B$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \rightarrow M(S)$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $m(S) \rightarrow B$: $IN_{UN}, io, A, B\}_{K_{UN\ S}}, \{n_B, io, A, B\}_{K_{BS}}$
M4. $B \rightarrow UN$: $IN_{UN}, io, A, B\}_{K_{COME}}$

? *UN* e *B* accettare la chiave sbagliata e *m* possono decifrare la loro successiva comunicazione! Quindi l'autenticazione della chiave (e la segretezza) fallisce!
Come possiamo proteggerci da questo tipo di attacco?

Tipo attacco difetto sul protocollo Otway-Rees (segue)

M1. $UN \rightarrow B$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \rightarrow S$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{UN\ S}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $S \rightarrow B$: $IN_{UN}, K_{AB}\}_{K_{COME}}, \{n_B, K_{AB}\}_{K_{BS}}$
M4. $B \rightarrow UN$: $IN_{UN}, K_{AB}\}_{K_{COME}}$

Attacco 2: Mallory può interpretare il ruolo di *S* in M2 e M3 riflettendo i componenti crittografati di M2 di nuovo in *B*. Vale a dire:

M1. $UN \rightarrow B$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \rightarrow M(S)$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $m(S) \rightarrow B$: $IN_{UN}, io, A, B\}_{K_{UN\ S}}, \{n_B, io, A, B\}_{K_{BS}}$
M4. $B \rightarrow UN$: $IN_{UN}, io, A, B\}_{K_{COME}}$

? *UN* e *B* accettare la chiave sbagliata e *m* possono decifrare la loro successiva comunicazione! Quindi l'autenticazione della chiave (e la segretezza) fallisce!

Come possiamo proteggerci da questo tipo di attacco?

Tipo attacco difetto sul protocollo Otway-Rees (segue)

M1. $UN \to B$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \to S$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{UN\ S}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $S \to B$: $IN_{UN}, K_{AB}\}_{K_{COME}}, \{n_B, K_{AB}\}_{K_{BS}}$
M4. $B \to UN$: $IN_{UN}, K_{AB}\}_{K_{COME}}$

Attacco 2: Mallory può interpretare il ruolo di *S* in M2 e M3 riflettendo i componenti crittografati di M2 di nuovo in *B*. Vale a dire:

M1. $UN \to B$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}$
M2. $B \to M(S)$: $Io, A, B, \{N_{UN}, io, A, B\}_{K_{COME}}, \{n_B, io, A, B\}_{K_{BS}}$
M3. $m(S) \to B$: $IN_{UN}, io, A, B\}_{K_{UN\ S}}, \{n_B, io, A, B\}_{K_{BS}}$
M4. $B \to UN$: $IN_{UN}, io, A, B\}_{K_{COME}}$

? *UN* e *B* accettare la chiave sbagliata e *m* possono decifrare la loro successiva comunicazione! Quindi l'autenticazione della chiave (e la segretezza) fallisce!

Come possiamo proteggerci da questo tipo di attacco?

Contorno

Il protocollo Andrew Secure RPC

Obiettivo: Scambia una chiave nuova, autenticata, segreta e condivisa, tra due principali che condividono una chiave simmetrica.

M1. $UN \rightarrow B$: $UN_{UN}\}_{K_{AB}}$
M2. $B \rightarrow UN$: $\{N_{UN} + 1, n_B\}_{K_{AB}}$
M3. $UN \rightarrow B$: $\{N_B + 1\}_{K_{AB}}$
M4. $B \rightarrow UN$: $\{K_{io_{AB}}, n_{io_B}\}_{K_{AB}}$

Stabilisce la chiave di sessione K_{io_AB} più nonce n_{io_B} (per una prossima sessione).

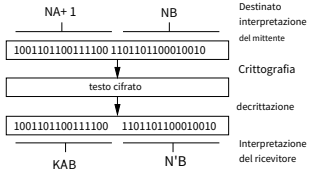
Tipo attacco difetto sul protocollo Andrew Secure RPC

M1. $UN \rightarrow B$: $UN_{UN}\}_{K_{AB}}$
M2. $B \rightarrow UN$: $\{N_{UN}+1, n_B\}_{K_{UN\ B}}$
M3. $UN \rightarrow B$: $\{N_B+1\}_{K_{AB}}$
M4. $B \rightarrow UN$: $\{K_{iaB}, n_{io\ B}\}_{K_{AB}}$

M1. $UN \rightarrow B$: $UN_{UN}\}_{K_{AB}}$
M2. $B \rightarrow UN$: $\{N_{UN}+1, n_B\}_{K_{AB}}$
M3. $UN \rightarrow M(B)$: $\{N_B+1\}_{K_{AB}}$
M4. $m(B) \rightarrow UN$: $\{N_{UN}+1, n_B\}_{K_{AB}}$

Assumiamo: nonce e chiavi sono rappresentati come sequenze di bit della stessa lunghezza (es. 64 bit).

Mallory può quindi registrare M2, intercettare M3 e riprodurre M2 come M4.



UN è ingannato nell'accettare il valore nonce $n_{UN}+1$ come nuova chiave di sessione. Questa chiave è non autenticato.

Questo attacco però non viola la segretezza (cfr. Otway-Rees).

Contorno

Scambio chiavi con CA (Denning & Sacco)

$$
\begin{aligned}
UN \rightarrow S: &\quad A, B \\
S \rightarrow UN: &\quad C_{UN}, C_B \\
UN \rightarrow B: &\quad C_{UN}, C_B, \{\{T_{UN}, K_{AB}\}_{K^{-1}_{UN}}\}_{K_B}
\end{aligned}
$$

Spiegazioni:

S: Server per un'autorità di certificazione. K_{AB}:

Chiave di sessione segreta.

K_B: Chiave pubblica di B, il partner di comunicazione previsto.

K^{-1}_{UN}: Firma con chiave privata di UN.

C_{UN}, C_B: Certificati per UN e B (nome, chiave pubblica, ...) T_{UN}: Un timestamp generato da UN.

Denning & Sacco (II)

$$UN \rightarrow S: \quad A, B$$
$$S \rightarrow UN: \quad C_{UN}, C_B$$
$$UN \rightarrow B: \quad C_{UN}, C_B, \underset{UN}{\{\{T_{UN}, K_{AB}\}_{K^{-1}}\}_{K_B}}$$

Scopo previsto:

- K_{AB} è noto solo a UN e B (chiave di sessione) B può
- essere sicuro che è stato inviato da UN:
  - può decifrarlo usando la chiave pubblica K_{UN}.
  - K_{UN} è destinato a UN dal certificato C_{UN}.
- B sa che il messaggio era destinato a lui/lei perché viene utilizzata la sua chiave pubblica.

- UN conosce ...?
- Il timestamp può essere utilizzato per limitare l'uso della chiave di sessione.

Denning & Sacco (III)

- L'attacco

$$UN \to S: \quad A, B$$
$$S \to UN: \quad C_{UN}, C_B$$
$$UN \to B: \quad C_{UN}, C_B, \{\{T_{UN}, K_{AB}\}_{K^{-1}_{UN}}\}_{K_B}$$

$$alice \to Charlie: \quad C_{alice}, C_{Charlie}, \{\{T_{alice}, K_{alice\ charlie}\}_{C K^{-1}_{alice}}\}_{K_{Charlie}}$$
$$Charlie \to bob: \quad alice, C_{bob}, \{\{T_{alice}, K_{alice\ charlie}\}_{K^{-1}_{alice}}\}_{K_{bob}}$$

- Bob crede che l'ultimo messaggio sia stato inviato da Alice
  - Quando poi usa $K_{alice\ charlie}$, Charlie può ascoltare. Quindi
  - la chiave non è né autenticata né segreta!
- Come possiamo difenderci da questo attacco man-in-the-middle? Sii esplicito sullo scopo: nominare i principali nell'ultimo passaggio.

$$UN \to B: C_{UN}, C_B, \{\{A, B, T_{UN}, K_{AB}\}_{K^{-1}_{UN}}\}_{K_B}$$

Denning & Sacco (III)

- L'attacco

$$UN \to S: \quad A, B$$
$$S \to UN: \quad C_{UN}, C_B$$
$$UN \to B: \quad C_{UN}, C_B, \{\{T_{UN}, K_{AB}\}_{K^{-1}_{UN}}\}_{K_B}$$

$$alice \to Charlie: \quad C_{alice}, C_{Charlie}, \{\{T_{alice}, K_{alice\ charlie}\}_{K^{-1}_{alice}}\}_{K_{Charlie}}$$
$$Charlie \to bob: \quad alice, C_{bob}, \{\{T_{alice}, K_{alice\ charlie}\}_{K^{-1}_{alice}}\}_{K_{bob}}$$

- Bob crede che l'ultimo messaggio sia stato inviato da Alice
  - Quando poi usa $K_{alice\ charlie}$, Charlie può ascoltare. Quindi
  - la chiave non è né autenticata né segreta!
- Come possiamo difenderci da questo attacco man-in-the-middle? Sii esplicito sullo scopo: nominare i principali nell'ultimo passaggio.

$$UN \to B: C_{UN}, C_B, \{\{A, B, T_{UN}, K_{AB}\}_{K^{-1}_{UN}}\}_{K_B}$$

Contorno

Ingegneria prudente dei protocolli di sicurezza

Principi proposti da Abadi e Needham (1994, 1995):

1. Ogni messaggio dovrebbe dire cosa significa.
2. Devono essere indicate le condizioni per l'attuazione di un messaggio.
3. Menzionare esplicitamente il nome del mandante nel messaggio se è essenziale per il significato.
4. Sii chiaro sul motivo per cui viene eseguita la crittografia. Riservatezza, autenticazione dei messaggi, associazione dei messaggi, ad es. $\{X, Y\}_{K^{-1}}$ contro $\{X\}_{K^{-1}}, \{Y\}_{K^{-1}}$

Ingegneria prudente (II)

6. Sii chiaro su quali proprietà stai assumendo sui nonces.
7. Le quantità prevedibili utilizzate per la risposta al problema dovrebbero essere protette dal replay.
8. I timestamp devono tenere conto della variazione dell'orologio locale e dei meccanismi di manutenzione dell'orologio.
9. Una chiave potrebbe essere stata usata di recente, ma essere vecchia.
10. Se viene utilizzata una codifica per presentare il significato di un messaggio, dovrebbe essere possibile dire quale codifica viene utilizzata.
11. Il progettista del protocollo dovrebbe sapere da quali relazioni di fiducia dipende il suo protocollo.

Ingegneria prudente (III)

Buon consiglio, ma...

- Sei sicuro di averli seguiti tutti?
- Il protocollo è quindi garantito per essere
- sicuro? È ottimale e/o minimo?

Ingegneria prudente: conclusione

- I protocolli di sicurezza possono ottenere proprietà che le primitive crittografiche da sole non possono offrire.
- Gli esempi mostrati sono semplici, ma le idee sono generali.
  Protocolli più avanzati in arrivo a breve.
- Anche tre fodere mostrano quanto sia difficile l'arte del design corretto.
  Possiamo elevarlo a scienza?

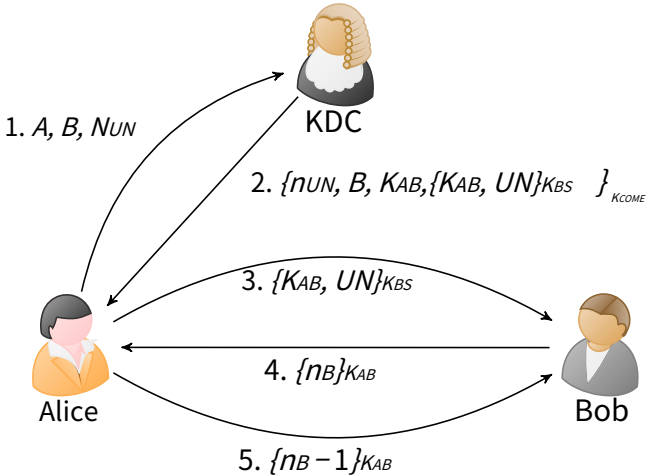
Contorno

Analisi automatizzata dei protocolli di sicurezza

- I protocolli di sicurezza sono notoriamente difficili da applicare correttamente.
- Le tecniche di verifica tradizionali (ad es. ispezione umana, test) non garantiscono il livello di garanzia necessario.
- Le tecniche automatizzate possono
  - esplorare sistematicamente lo spazio degli stati del sistema e garantire una copertura completa dei suoi comportamenti
  - supportare l'implementazione di soluzioni/meccanismi sicuri
- Negli anni sono emersi numerosi strumenti per l'analisi automatica dei protocolli di sicurezza: Casper/FDR, NRL, AVISPA Tool (che comprende OFMC, CL-AtSe, SATMC e TA4SP), AVANTSSAR Platform, SPACIoS Tool, Proverif, Scyther, . ..

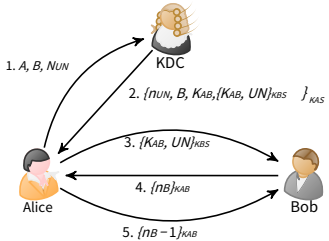
Contorno

Protocollo a chiave condivisa Needham-Schroeder



Obiettivo di sicurezza: scambio di chiavi autenticate.

Debolezza di NSSK



- B non può controllare l'aggiornamento del messaggio inviato al passaggio 3.
- I passaggi 4 e 5 impediscono di assicurarsi (a B) che A sia attualmente impegnato a utilizzare la chiave di sessione K_{AB}.
- Tuttavia, se una chiave di sessione precedente, dire K_{io} $_{AB}$ viene compromesso, poi l'attaccante fa B accettare K_{AB} di nuovo riproducendo $\{K_{io}$ $_{AB}, UN\}_{K_{BS}}$
- Questa debolezza può essere risolta con l'inclusione di un timestamp.

Contorno

Kerberos

- Protocollo per autenticazione/controllo accessi per applicazioni client/server.
- Nella mitologia greca, Kerberos è un cane a tre teste che fa la guardia all'ingresso dell'Ade.
  Kerberos moderno intendeva avere tre componenti per proteggere il cancello di una rete: autenticazione, contabilità e controllo.
  Ultime due teste mai implementate.
- Sviluppato come parte del Progetto Athena (MIT, anni '80).
  - Versione V (1993) ora standard.
  - Ampiamente usato, ad es. Microsoft Windows e Microsoft Active Directory.

Protocollo di autenticazione Kerberos: obiettivi (1/2)

(A partire dal: https://www.kerberos.org/software/tutorial.html)

- La password dell'utente non deve mai viaggiare in rete;
- La password dell'utente non deve mai essere memorizzata in nessuna forma sulla macchina client: deve essere immediatamente scartata dopo essere stata utilizzata;
- La password dell'utente non deve mai essere memorizzata in forma non crittografata nemmeno nel database del server di autenticazione;
- Single Sign-On: all'utente viene chiesto di inserire una password una sola volta per sessione di lavoro. Pertanto gli utenti possono accedere in modo trasparente a tutti i servizi per i quali sono autorizzati senza dover reinserire la password durante questa sessione;

Protocollo di autenticazione Kerberos: obiettivi (2/2)

- Le informazioni di autenticazione risiedono solo sul server di autenticazione. I server delle applicazioni non devono contenere le informazioni di autenticazione per i propri utenti. Ciò è essenziale per ottenere i seguenti risultati:
  - L'amministratore può disabilitare l'account di qualsiasi utente agendo da un'unica postazione senza dover agire sui più server applicativi;
  - Quando un utente cambia la sua password, questa viene cambiata per tutti i servizi
  - contemporaneamente; Non c'è ridondanza delle informazioni di autenticazione che altrimenti dovrebbero essere salvaguardate in vari luoghi;
- Autenticazione reciproca: non solo gli utenti devono dimostrare di essere chi dicono, ma, quando richiesto, anche gli application server devono dimostrare la loro autenticità al client.
- Dopo il completamento dell'autenticazione e dell'autorizzazione, il client e il server devono essere in grado di stabilire una connessione crittografata.

Kerberos – requisiti

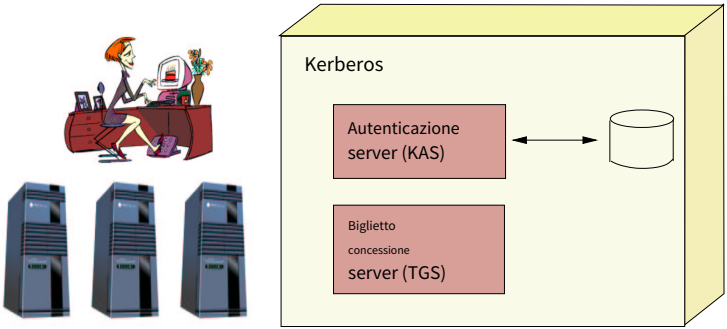
Sicuro: Un intercettatore non dovrebbe essere in grado di ottenere informazioni per impersonare un utente.

Affidabile: Poiché molti servizi dipendono da Kerberos per il controllo degli accessi, deve essere altamente affidabile e supporta un'architettura distribuita, in cui un sistema può eseguire il backup di un altro.

Trasparente: Ogni utente deve inserire un'unica password per ottenere i servizi di rete; altrimenti non dovrebbe essere a conoscenza dei protocolli sottostanti.

Scalabile: Il sistema dovrebbe scalare per supportare un gran numero di utenti e server; questo suggerisce un'architettura modulare e distribuita.

Kerberos versione IV: Architettura

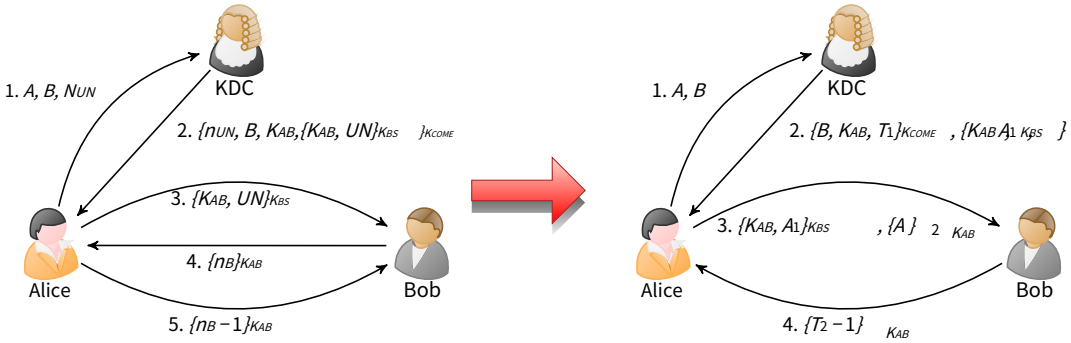


Autenticazione usando KAS, il Server di autenticazione Kerberos.
Autorizzazione usando TGS, il Server di assegnazione dei biglietti.
Controllo di accesso dove i server controllano i ticket TGS.

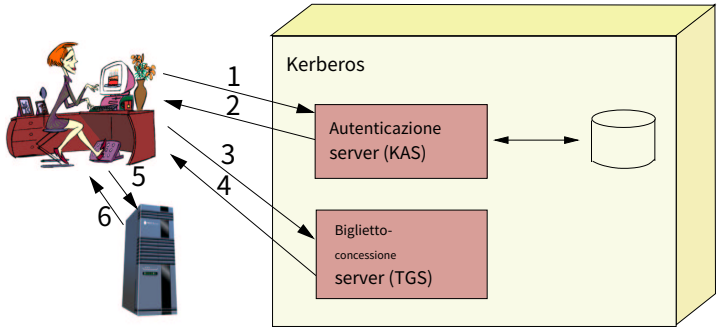
Protocollo di autenticazione Kerberos

Liberamente basato sul protocollo a chiave condivisa Needham-Schroeder:

- Timestamp invece di nonces per assicurare l'aggiornamento delle chiavi di sessione.
- Rimozione della crittografia annidata.



Kerberos IV: protocollo



Autenticazione	messaggi 1 e 2.	Una volta per sessione di accesso
Autorizzazione	messaggi 3 e 4.	utente.Una volta per tipo di servizio.
Servizio	messaggi 5 e 6.	Una volta per sessione di servizio.

Di seguito presentiamo le tre parti (leggermente semplificate).

Fase di autenticazione

1. $UN \rightarrow KAS \quad : \quad A, TGS$
2. $KAS \rightarrow UN \quad : \quad \{K_{A,TGS}, TGS, T1\}_{K_{COME}} \; , \underbrace{\{A, TGS, K_{A,TGS}, T1\}_{K_{KAS,TGS}}}_{AuthTicket}$

- *UN* accede alla workstation e richiede le risorse di rete.
- KAS accede al database e invia *UN* una chiave di sessione $K_{A,TGS}$ e un biglietto crittografato *AuthTicket*.
- $K_{A,TGS}$ ha una durata di diverse ore (a seconda dell'applicazione). K_{COME} è derivato
- dalla password dell'utente, ad es $K_{COME} = h(Parola\ d'ordine_{UN} io_{UN})$. Sia le chiavi
- utente che quelle server devono essere registrate nel database.
- *UN* digita la password sul client per decrittografare i risultati. Il ticket e la chiave di sessione vengono salvati. La password dell'utente è stata dimenticata. *UN* viene disconnesso quando $K_{A,TGS}$ scade.

Fase di autorizzazione

3. A → TGS: $\{A, TGS, K_{A,TGS}, T_1\}_{K_{AS,TGS}}$, $\{UN, T_2\}_{K_{A,TGS}}$, B

AuthTicket *autenticatore*

4. TGS → UN : $\{K_{AB}, B, T_3\}_{K_{A,TGS}}$, $\{A, B, K_{AB}, T_3\}_{K_{BS}}$

ServTicket

Prima *UN*l primo accesso di risorse di rete *B*:

- *UN* regali *AuthTicket* dal messaggio 2 al TGS insieme a un nuovo *autenticatore* ,con durata breve (secondi).
  - Ruolo di autenticatore? La validità breve impedisce gli attacchi di replay. I server
  - memorizzano gli autenticatori recenti per impedire la riproduzione immediata.
- Problemi con il TGS *UN* una nuova chiave di sessione K_{AB} (durata di pochi minuti) e un nuovo biglietto *ServTicket* . K_{BS} è la chiave condivisa tra TGS e la risorsa di rete.

Fase di autorizzazione

3. A → TGS: $\{A, TGS, K_{A,TGS}, T_1\}_{K_{KAS,TGS}}$, $\{UN, T_2\}_{K_{A,TGS}}$, B

AuthTicket *autenticatore*

4. TGS → UN : $\{K_{AB}, B, T_3\}_{K_{A,TGS}}$ $\{A, B, K_{AB}, T_3\}_{K_{BS}}$

ServTicket

Prima *UN*l primo accesso di risorse di rete *B*:

- *UN* regali *AuthTicket* dal messaggio 2 al TGS insieme a un nuovo *autenticatore* ,con durata breve (secondi).
  - Ruolo di autenticatore? La validità breve impedisce gli attacchi di replay. I server
  - memorizzano gli autenticatori recenti per impedire la riproduzione immediata.
- Problemi con il TGS *UN* una nuova chiave di sessione K_{AB} (durata di pochi minuti) e un nuovo biglietto *ServTicket* . K_{BS} è la chiave condivisa tra TGS e la risorsa di rete.

Fase di servizio

5. A → B : $\{A, B, K_{AB}, T_3\}_{K_{BS}}$ $\{UN, T_4\}_{K_{AB}}$

ServTicket *autenticatore*

6. Si → UN : $\{T_4 + 1\}_{K_{AB}}$

Per *UN* per accedere alle risorse di rete *B*:

- *UN* regali K_{AB} dalle 4 alle *B* insieme a nuovo *autenticatore*.

  In pratica, potrebbero essere inviate anche altre informazioni per il server.

- *B* risposte, servizio di autenticazione.

Scalabilità: regni multipli/Kerberi

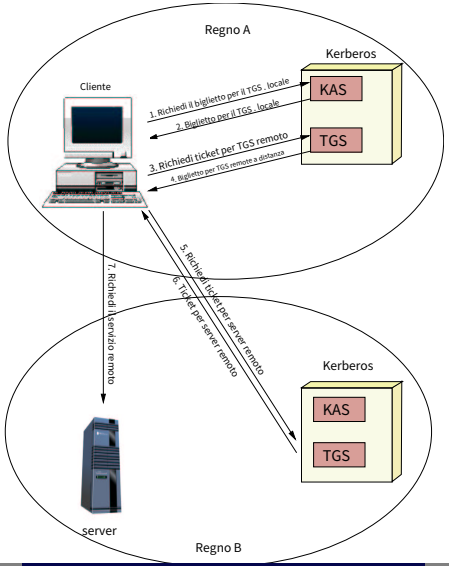
- UN regno è definito da un server Kerberos.

  Il server memorizza le password dell'utente e del server delle applicazioni per il realm.

- Le grandi reti possono essere suddivise in ambiti
- amministrativi. Kerberos supporta i protocolli tra regni.
  - I server si registrano tra loro.
  - Per *UN* accedere *B* in un altro regno, il *TGS* in *UM*l regno di 's riceve la richiesta e concede il biglietto per l'accesso *TGS* in *B*'s regno.
- Estensione del protocollo semplice: due passaggi aggiuntivi.

  Ma per n regni, $oh(m_2)$ problema di distribuzione delle chiavi (nella versione IV).

Comunicazione tra regni



Limitazioni di Kerberos IV

- M1: la crittografia non è necessaria, ma l'attaccante può inondare S

$$UN \rightarrow KAS : A, TGS$$

- M2: crittografia annidata non necessaria; eliminato in Kerberos V.

$$KAS \rightarrow UN : \{K_{A,TGS}, TGS, T_1, \{A, TGS, K_{A,TGS}, T_1\}_{K_{KAS,TGS}} \}_{K_{COME}}$$

- Si basa su orologi (approssimativamente) sincronizzati e senza compromessi.
  Se l'host è compromesso, l'orologio può essere compromesso e la riproduzione è facile.

Alcune di queste limitazioni si applicano anche a Kerberos V.

Ottenere e usare Kerberos

- Ottenere Kerberos:
  - MIT: http://web.mit.edu/kerberos/www/.
  - Distribuzioni per Linux e Windows 2000.
- Molti protocolli/applicazioni sono stati “kerberizzati” tra cui:
  - IP mobile (SeaMoby), protocollo di prenotazione delle risorse (RSVP),
  - Telnet, FTP, SNMP, TLS,
  - DHCP, GSS-API, SASL, DMS (tramite TKEY), Windows 2000 per
  - tutte le procedure di autenticazione e IKE (che esegue IKE
  - Phase 2, su Artifical Kerberos IKE SA),

Conclusioni

I protocolli di sicurezza sono

- chiave per proteggere i sistemi distribuiti
- onnipresenti: coprono praticamente tutti i domini applicativi e l'intero stack di protocollo
- (apparentemente) semplice: utilizzare protocolli stabiliti quando disponibili/possibile, ma essere pronti a progettarne/svilupparne/utilizzarne uno nuovo se necessario.