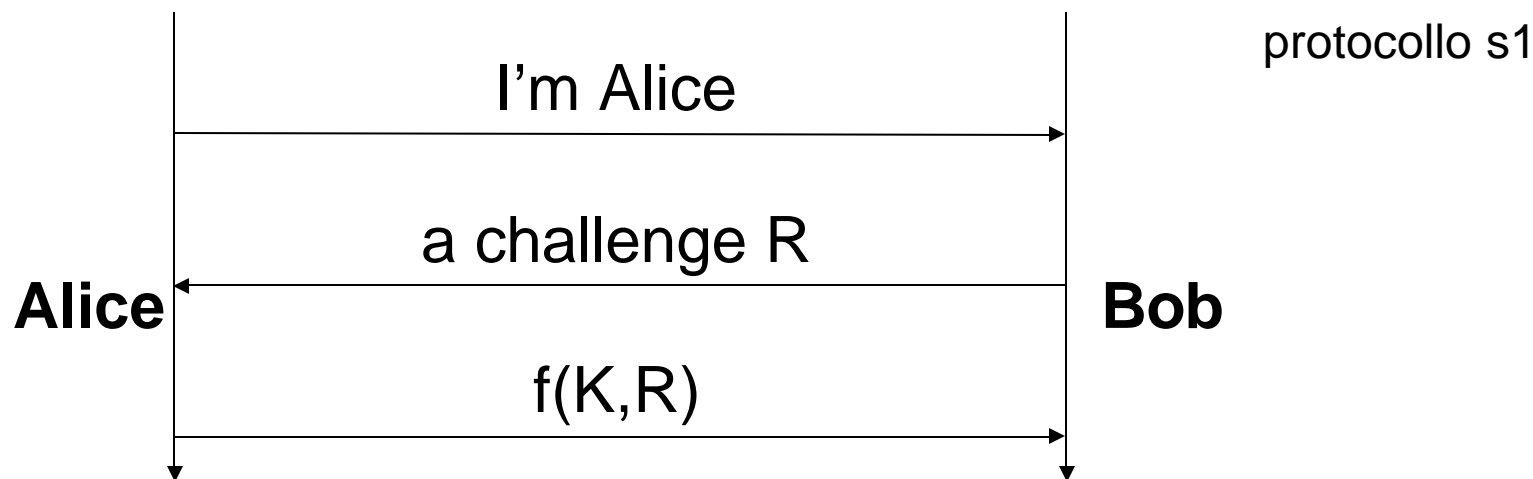


# tecniche crittografiche e protocolli

# obiettivi

- autenticazione one-way e mutua
- scambio di chiavi di sessione
- scambio dei dati
  - integrità
  - confidenzialità

# autenticazione one-way con shared secret (s1)



- $K$  è un segreto condiviso
- $f(K,R)$  è  $K\{R\}$  oppure  $h(R|K)$
- problemi
  - Cindy può sniffare e installare un attacco
    - know plaintext
    - se  $K$  è derivata da una password un off-line password guessing
  - chi legge il key db di A o B può impersonare A

# problemi dell'autenticazione one-way senza altre contromisure

- Alice non autentica Bob che può essere facilmente impersonato
  - si deve fare mutua autenticazione

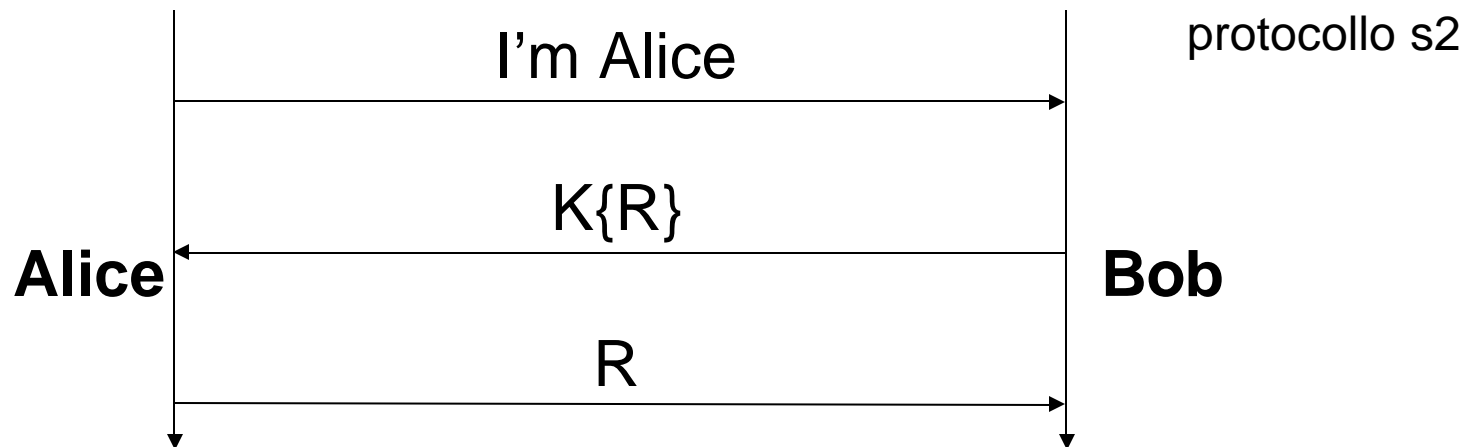
# replay attack

- Cindy registra il/i messaggi e li re-invia in una sessione successiva
  - Cindy non ha bisogno di conoscere  $K$  per l'attacco
- se un valore di  $R$  viene riutilizzato allora il prot.  $s1$  è vulnerabile al “replay attack”

# nonces

- un *nonce* è un valore o stringa che è usato una sola volta
  - tutti i challenge devono essere nonces altrimenti i protocolli diventano vulnerabili ad un replay attack
- esempi di nonces
  - timestamp
    - predicibile
    - dipende dalla vulnerabilità dei meccanismi di settaggio del clock
  - numero di sequenza
    - predicibile
    - che succede dopo il boot?
    - che succede dopo un overflow?
  - random number
    - devono essere molto grandi altrimenti si ripetono

# autenticazione one-way con shared secret (s2)



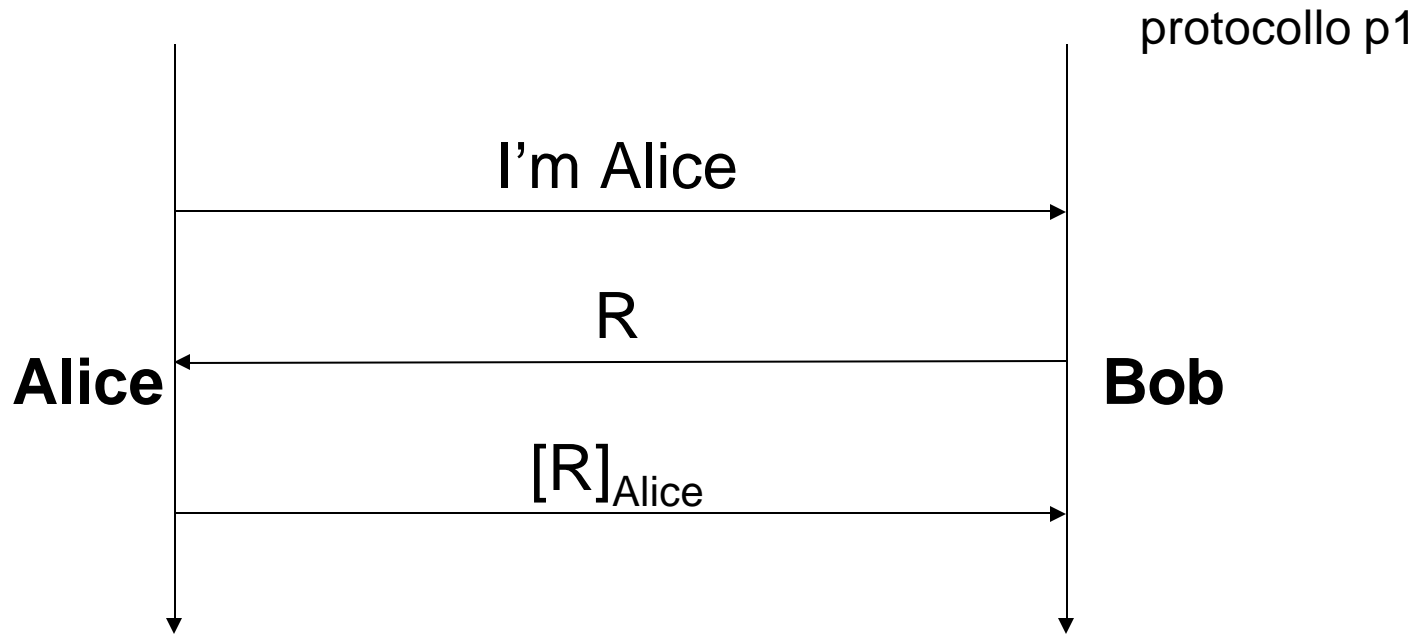
- $R$  non deve essere predicibile da Alice
- richiede crittografia reversibile (non si può usare l'hash)
- problemi
  - Cindy può sniffare e installare un attacco
    - know plaintext
    - se  $K$  è derivata da una password un off-line password guessing
  - chi legge il key db di A o B può impersonare A

# uso promiscuo della chiave

- chiunque impersoni Bob può ottenere
  - cifratura con  $K$  di qualsiasi cosa (prot.  $s1$ )
  - decifratura con  $K$  di qualsiasi cosa (prot.  $s2$ )
- in generale una chiave dovrebbe essere usata per un solo scopo (un solo protocollo) altrimenti...
  - protocolli indipendentemente sicuri possono essere vulnerabili se usati assieme
  - l'introduzione di nuovi protocolli che usano la stessa chiave può rendere vulnerabili i vecchi
- in alternativa usare nonces strutturati per ciascuna funzionalità

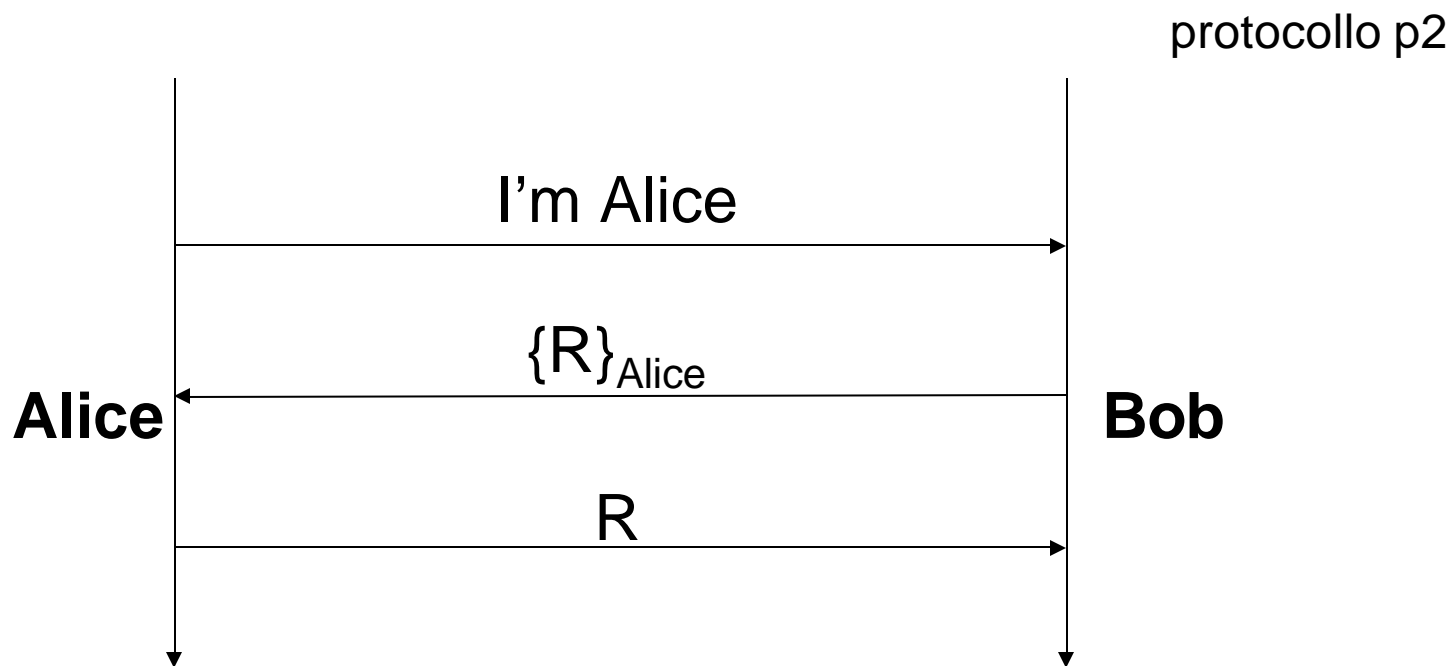


# autenticazione one-way con chiave pubblica (p1)



- vulnerabilità: come s1 ma chi legge il key db di B non può impersonare A

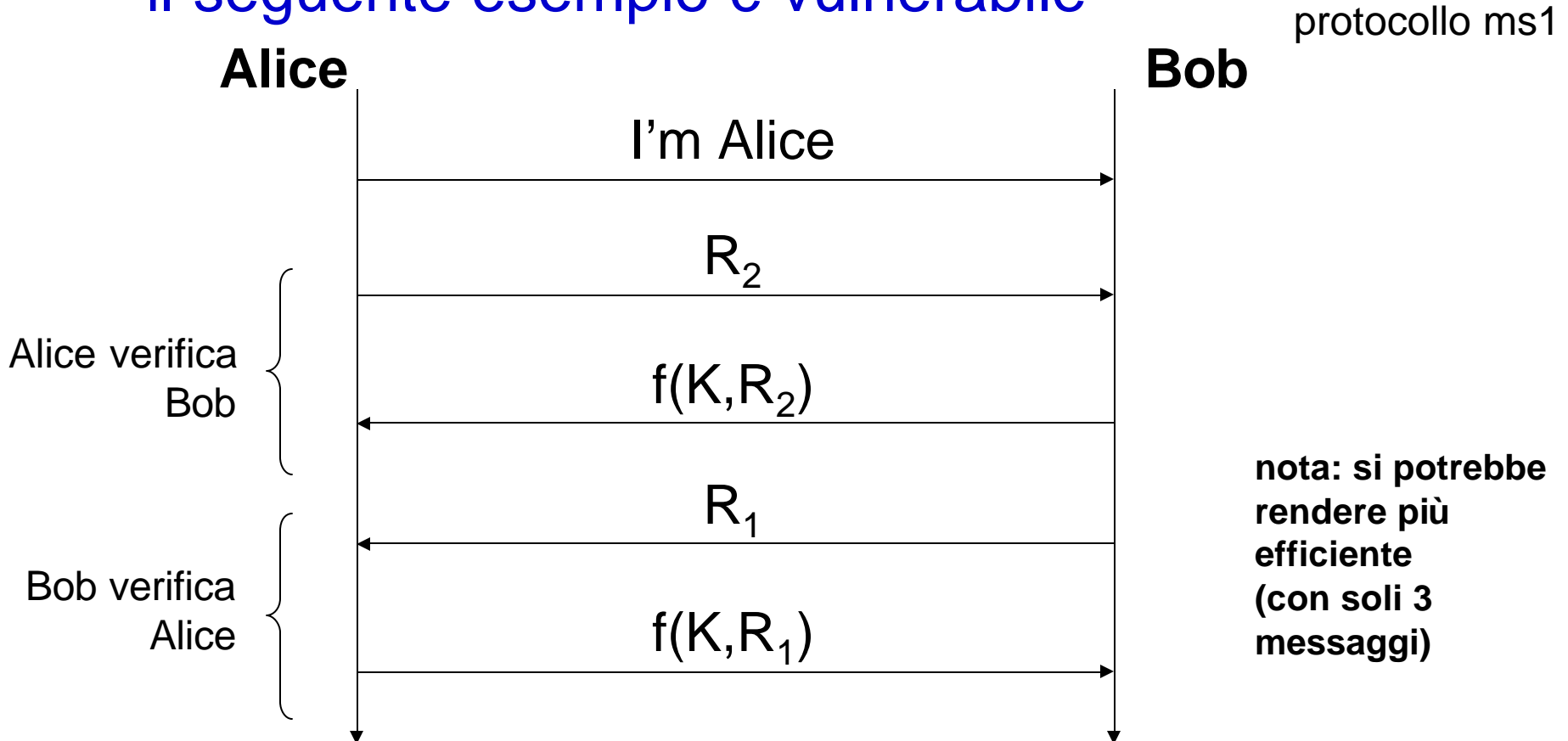
# autenticazione one-way con chiave pubblica (p2)



- vulnerabilità: come s2 ma chi legge il key db di B non può impersonare A

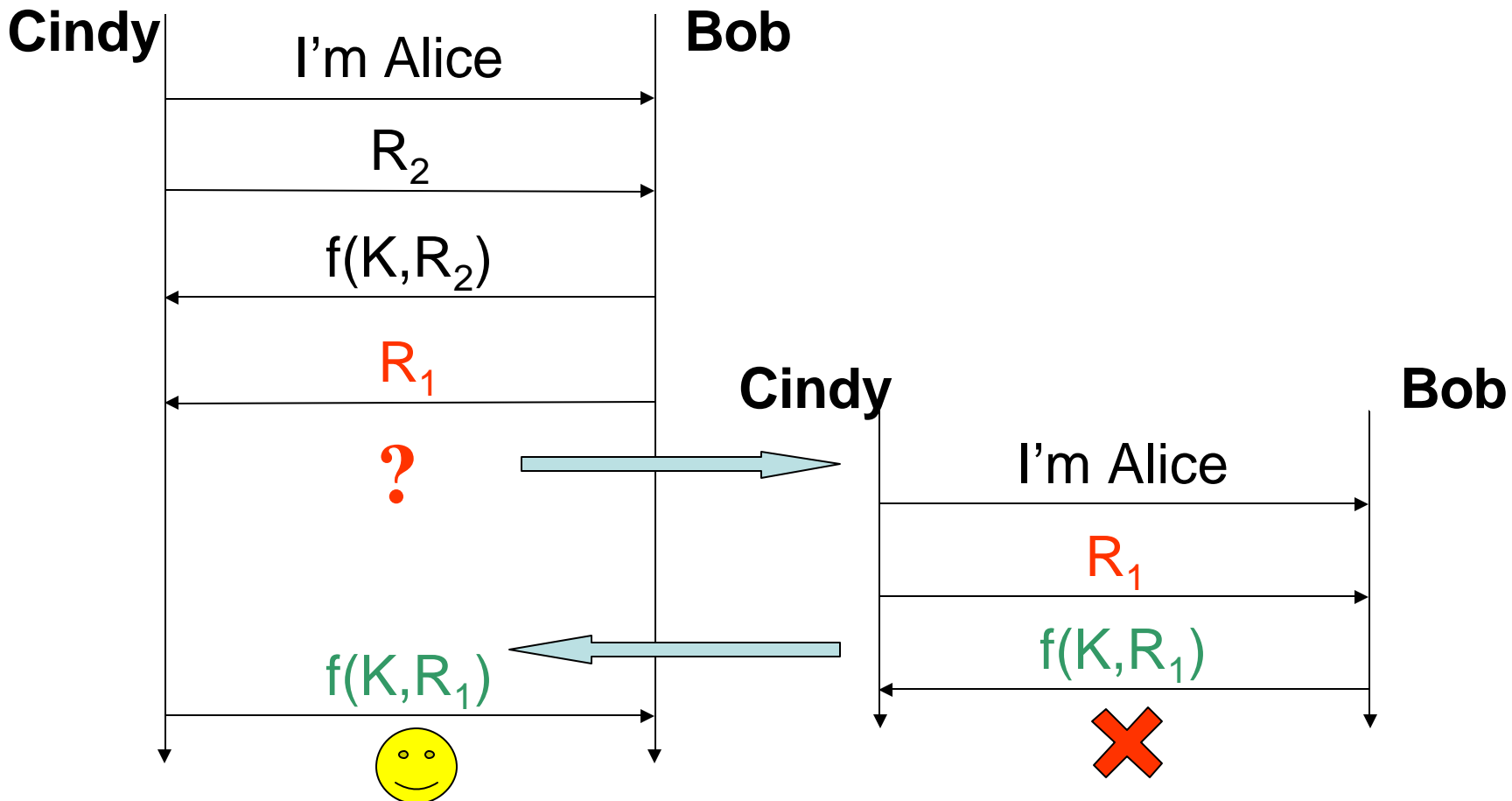
# autenticazione mutua con shared secret (ms1)

- per l'autenticazione mutua non basta avere due autenticazioni una dopo l'altra
- il seguente esempio è vulnerabile

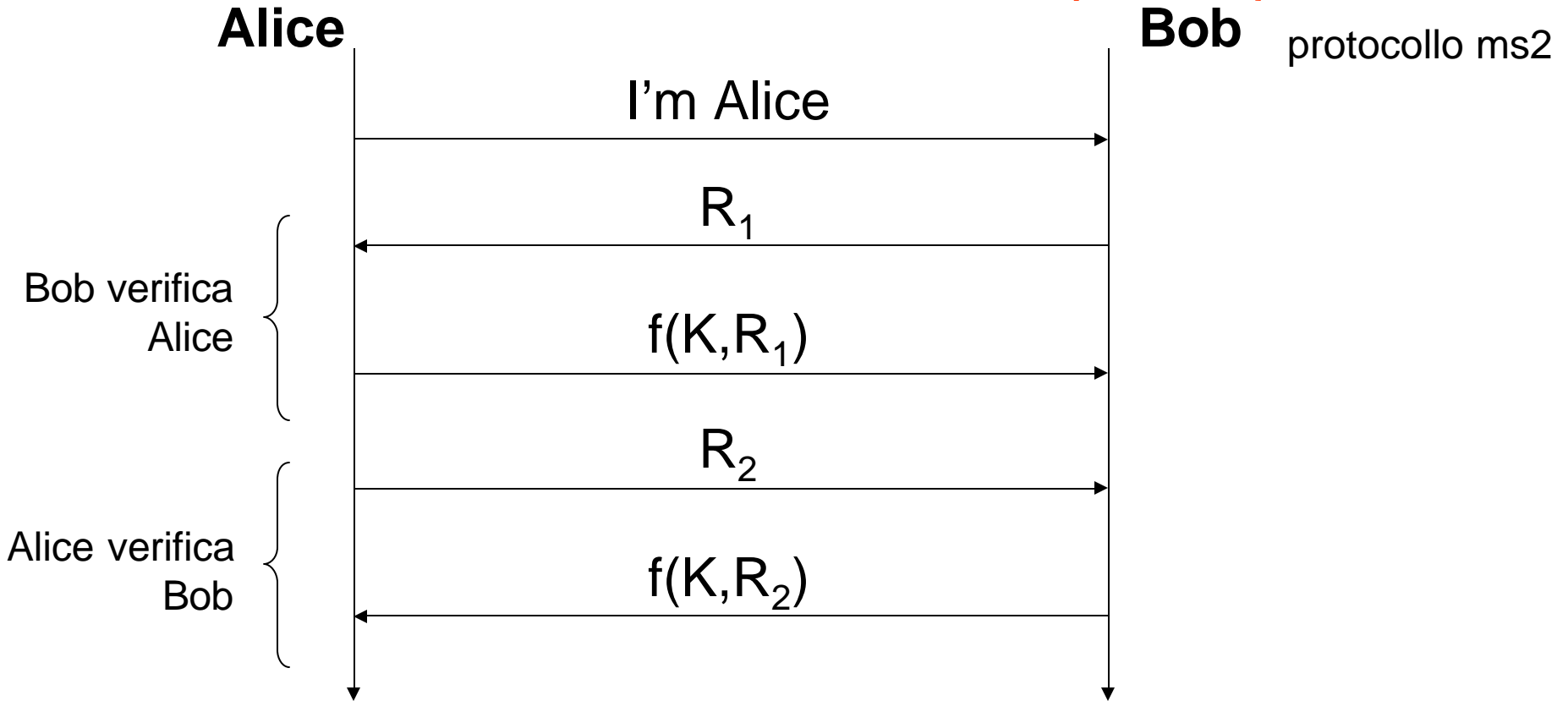


# reflection attack

- nel reflection attack Cindy può sfruttare il protocollo stesso (su un'altra sessione) per ottenere le informazioni per impersonare A



# autenticazione mutua con shared secret (ms2)



- Cindy può impersonare Alice con un reflection attack?
- Cindy può impersonare Bob?

# reflection attack: contromisura

- A e B non devono fare la stessa cosa
- es. usare chiavi differenti nei due versi
  - es. totalmente differenti
  - es. chiavi derivate
    - es.  $K$  e  $K+1$
- es. usare challenge strutturalmente differenti
  - es.  $R_1$  pari e  $R_2$  dispari
  - es. concatenare il nome
    - es. Alice| $R_1$  e Bob| $R_2$
    - es. server| $R_1$  e client| $R_2$

# cifratura dei dati

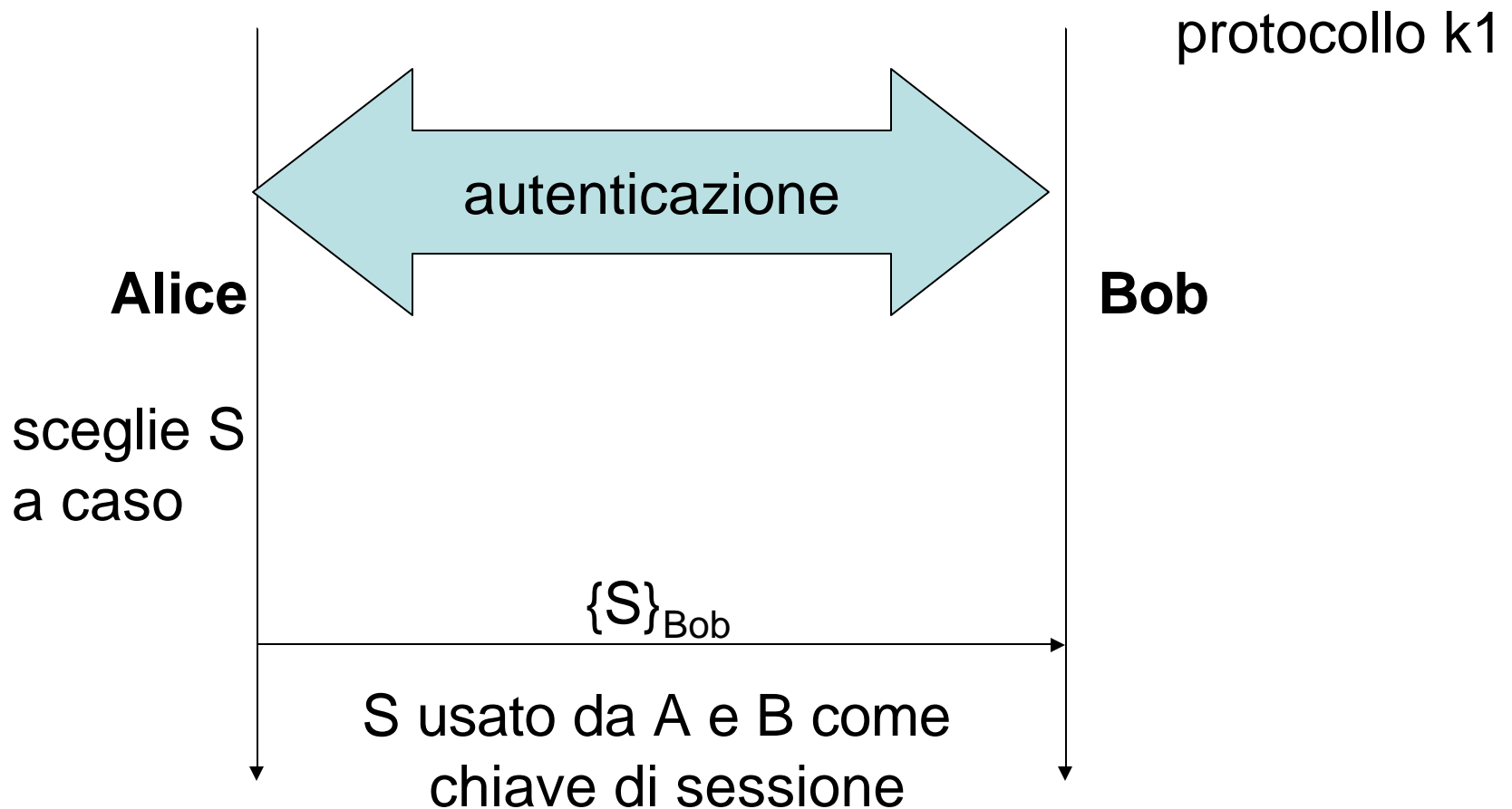
# chiavi di autenticazione e chiavi di sessione

- la chiave simmetrica usata per la cifratura è detta *session key*
- la session key è bene che sia diversa dalla/e chiave/i usata/e per l'autenticazione
  - la/le chiave/i per l'autenticazione deve/devono durare nel tempo (***long term secret***)
  - la chiave di sessione si usa molto e “si deteriora” (***short term secret***)
- una buona session key deve essere
  - diversa per ciascuna sessione
  - non predicibile da un eavesdropper
    - deve essere derivata anche (ma non necessariamente solo) da un numero random

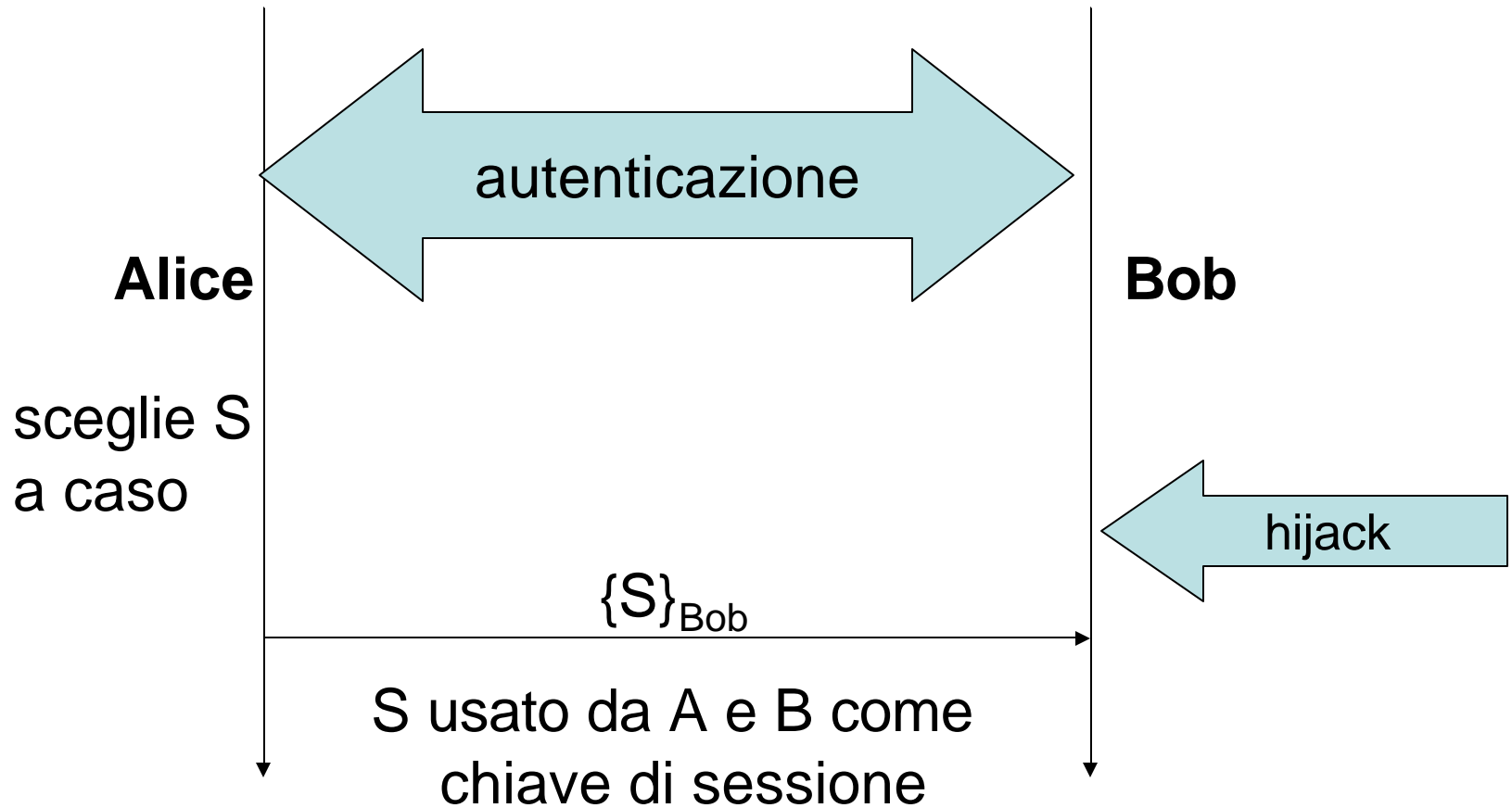


# autenticazione e sessione

- trova la vulnerabilità



# hijack



- nulla mi assicura che  $S$  sia stato generato da Alice

# scambio di chiave di sessione

- è necessario che ci sia la prova che la chiave di sessione provenga dallo stesso soggetto che è stato autenticato
- protocollo k2: come k1 ma la chiave è autenticata in qualche modo
  - $[ \{S\}_{Bob} ]_{Alice}$  oppure  $K\{S\}$
  - non vulnerabile ad hijacking
- oppure scambio di chiave nell'ambito dell'autenticazione
- se la sessione dura molto la chiave di sessione va cambiata periodicamente
  - le tecniche di crittoanalisi hanno bisogno di una certa quantità di ciphertext per trovare la chiave
  - la chiave va cambiata prima che si raggiunga una quantità di ciphertext che renda possibile l'attacco

# key rollover

- il cambiamento periodico della chiave di sessione è detto *key rollover*
  - la nuova chiave di sessione può essere negoziata come la prima (inefficiente) oppure calcolata
  - tipicamente tutte le chiavi di sessione sono calcolate a partire da un “master secret” tramite la combinazione di vari possibili strumenti
    - contatori, shuffling, hash, crittografia, ecc
  - il master secret è tipicamente generato a partire da un numero pseudo-casuale (di qualità e quindi di generazione inefficiente) e da altri segreti a lungo termine
    - meglio se entrambe le parti concorrono alla creazione del master secret
  - il master secret è solitamente usato solo per generare le chiavi
    - la quantità pubblica di ciphertext prodotto cifrando con il master secret deve essere il minimo possibile (possibilmente niente)