

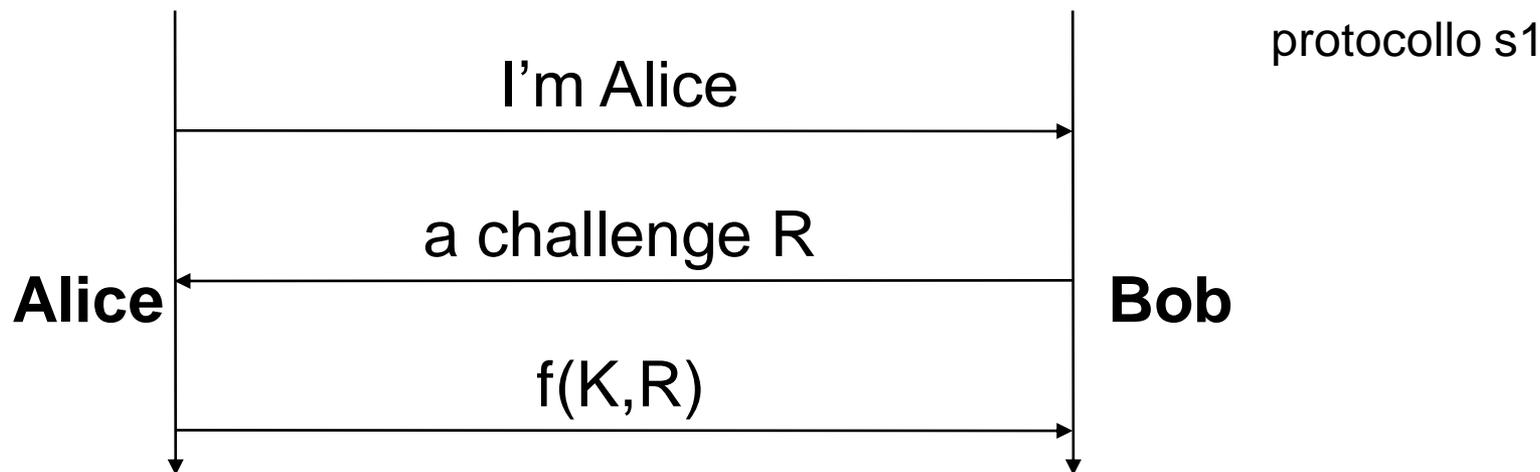
# tecniche crittografiche e protocolli

# obiettivi

- autenticazione
- scambio di *chiavi di sessione*
  - e conseguente scambio dei dati rispettando integrità e confidenzialità
- perfect forward secrecy

# autenticazione

# autenticazione one-way con shared secret



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

# problema dell'autenticazione one-way

- Alice non autentica Bob che può essere facilmente impersonato
  - in molti contesti è necessaria una mutua autenticazione

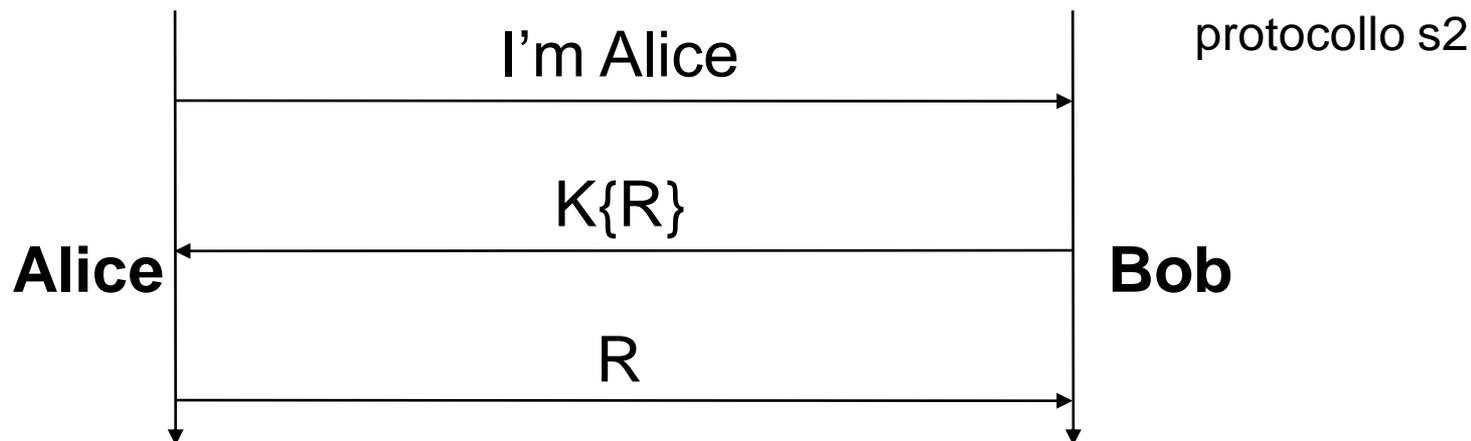
# replay attack

- se un valore di  $R$  viene riutilizzato allora il prot.  $s1$  è vulnerabile al *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

# nonces

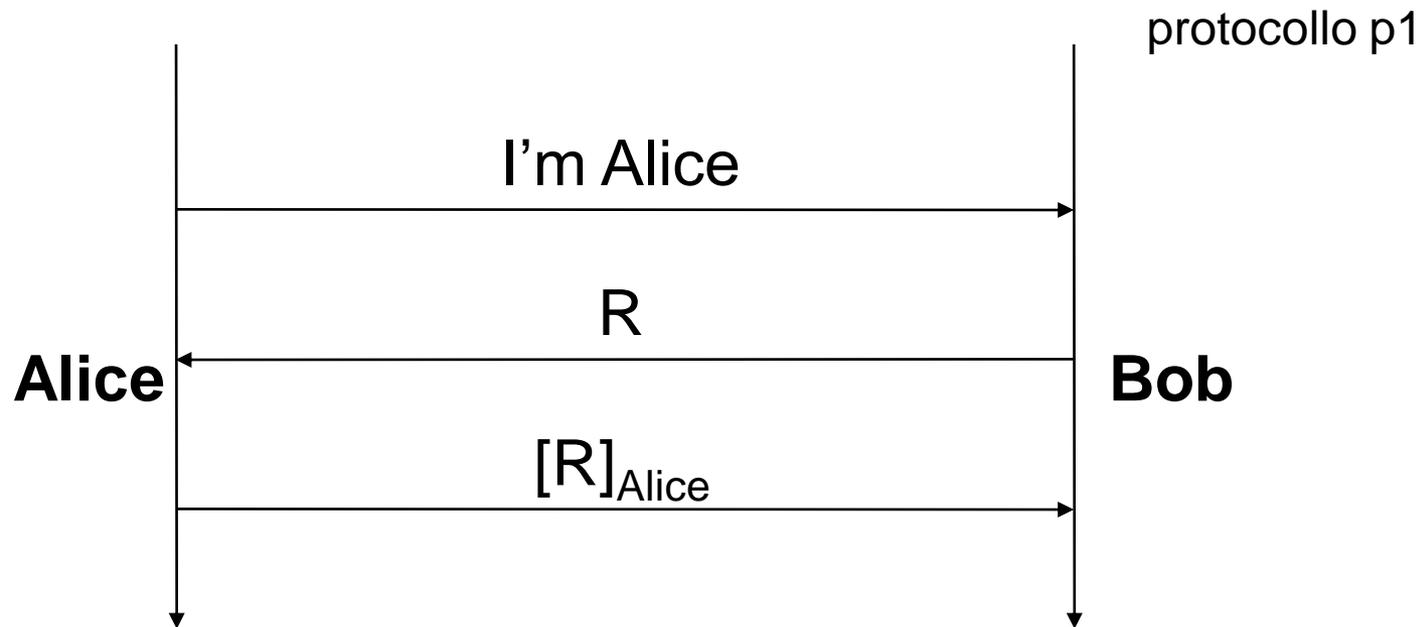
- un *nonce* è un valore o stringa che è **usato una sola volta**
  - tutti i challenge devono essere nonce altrimenti i protocolli diventano vulnerabili ad un replay attack
- alle volte deve anche essere **non predicibile**
- esempi di nonces
  - **timestamp**
    - predicibile
    - dipende dalla vulnerabilità dei meccanismi di settaggio del clock
  - **numero di sequenza**
    - predicibile
    - che succede dopo il boot?
    - che succede dopo un overflow?
  - **random number**
    - necessario un buon “seed”
    - necessario un buon generatore di numeri casuali

# autenticazione one-way con shared secret (s2)



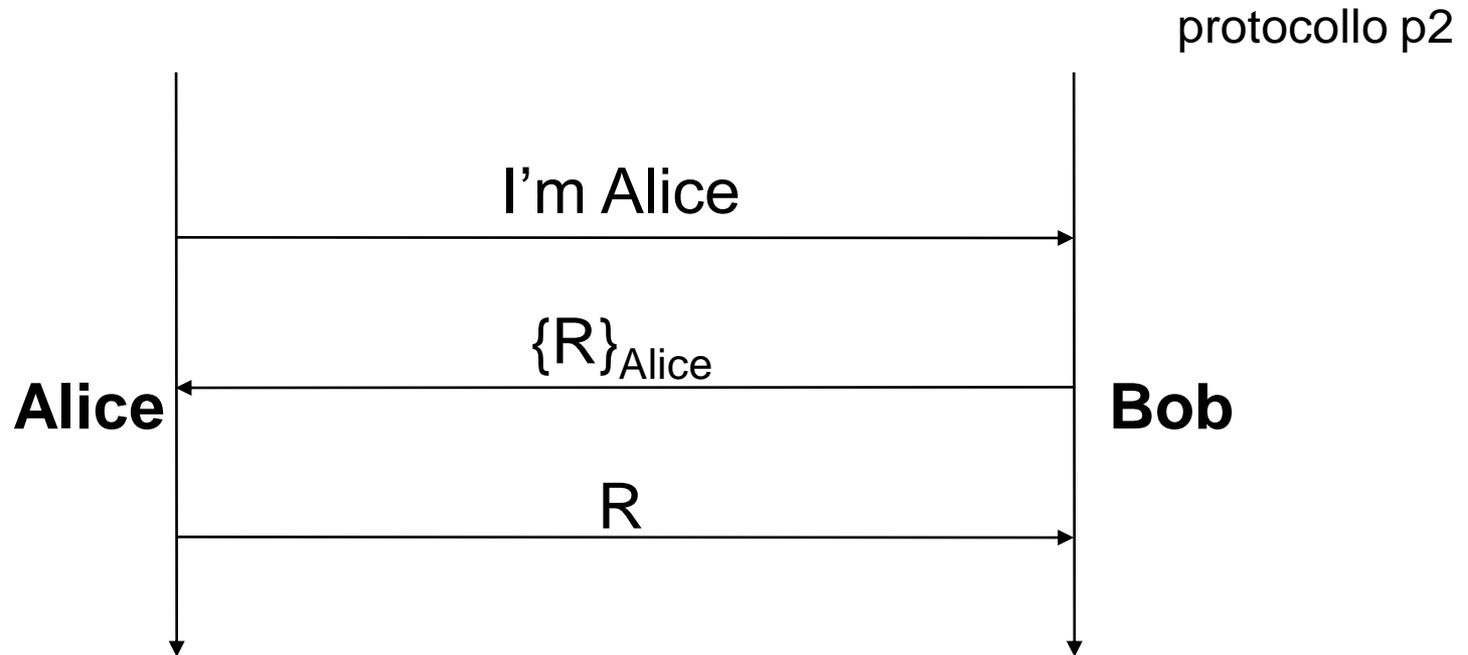
- $R$  deve essere **non predicibile** da Alice
- richiede crittografia reversibile (non si può usare l'hash)
- problemi
  - Cindy può sniffare e installare un attacco
    - know plaintext
    - se  $K$  è derivata da una password un off-line password guessing
  - chi legge il key db di A o B può impersonare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

# 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

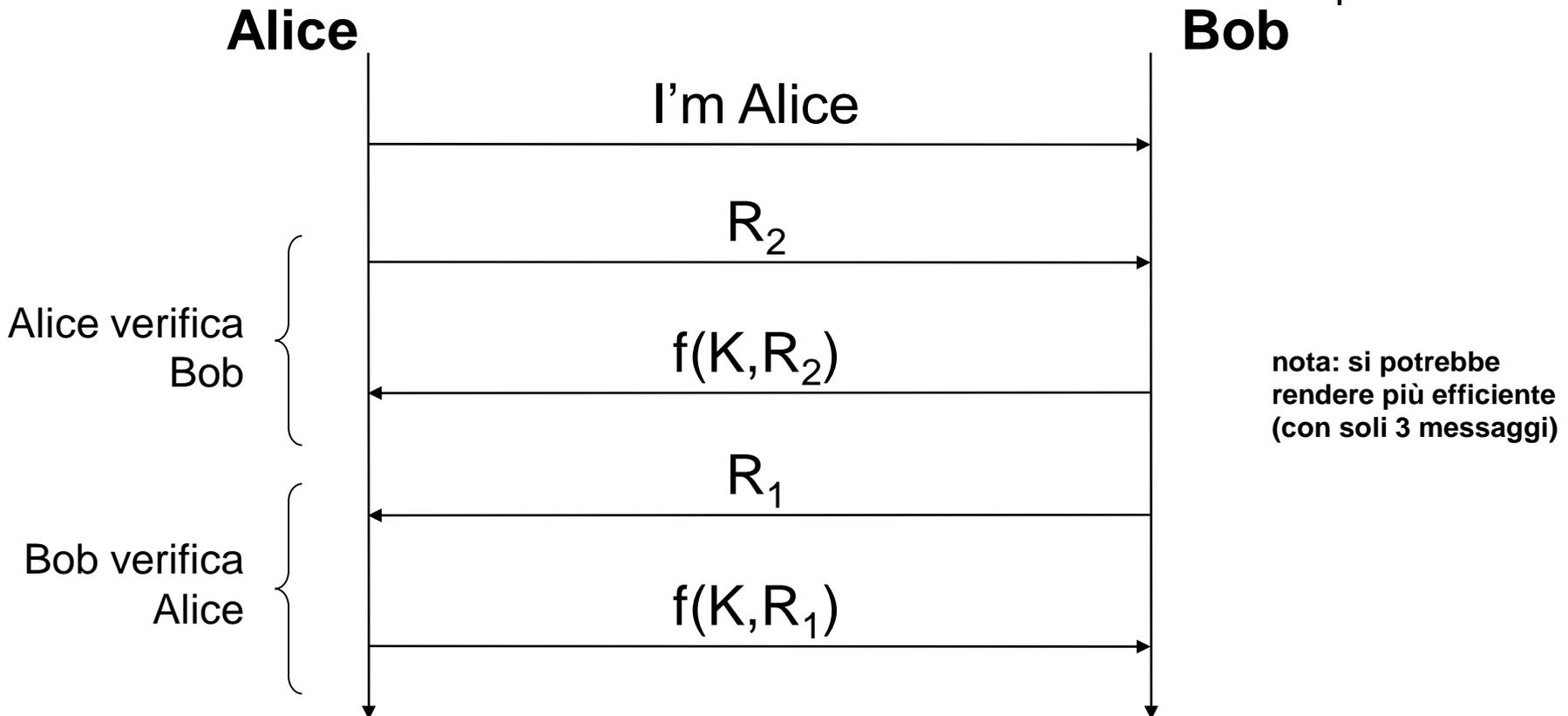
# evitare l'uso promiscuo delle chiavi

- chiunque impersoni Bob può scegliere un messaggio  $m$  e ottenere...
  - cifratura con  $K$  di  $m$  (prot.  $s_1$  e  $p_1$ )
  - decifratura con  $K$  di  $m$  (prot.  $s_2$  e  $p_2$ )
- in generale una chiave dovrebbe essere **usata per un solo scopo** (cioè con 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 nonce strutturati per ciascuna funzionalità

# autenticazione mutua con shared secret (ms1)

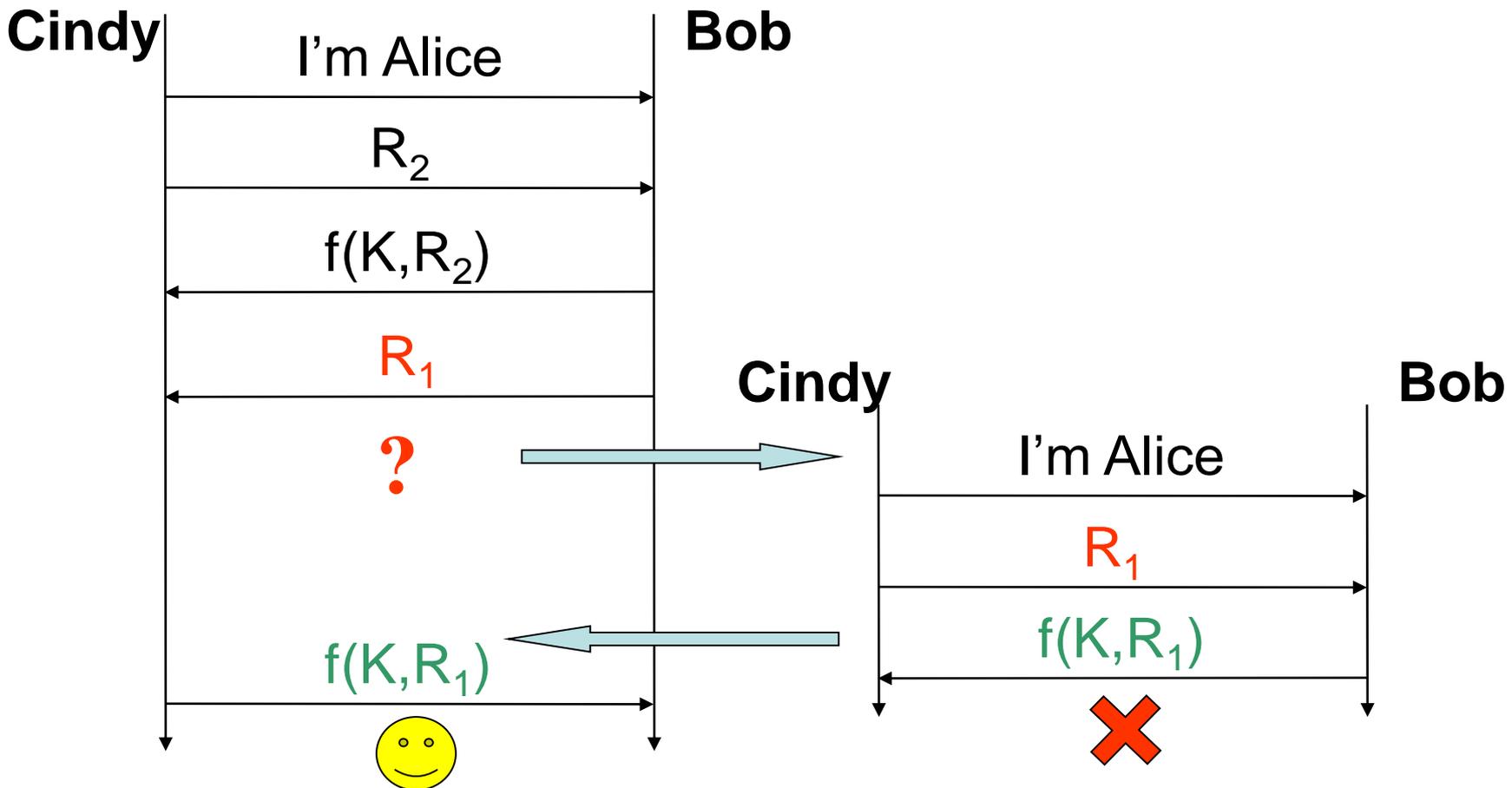
- per l'autenticazione mutua non basta avere due autenticazioni one-way una dopo l'altra
- il seguente esempio è vulnerabile

protocollo ms1

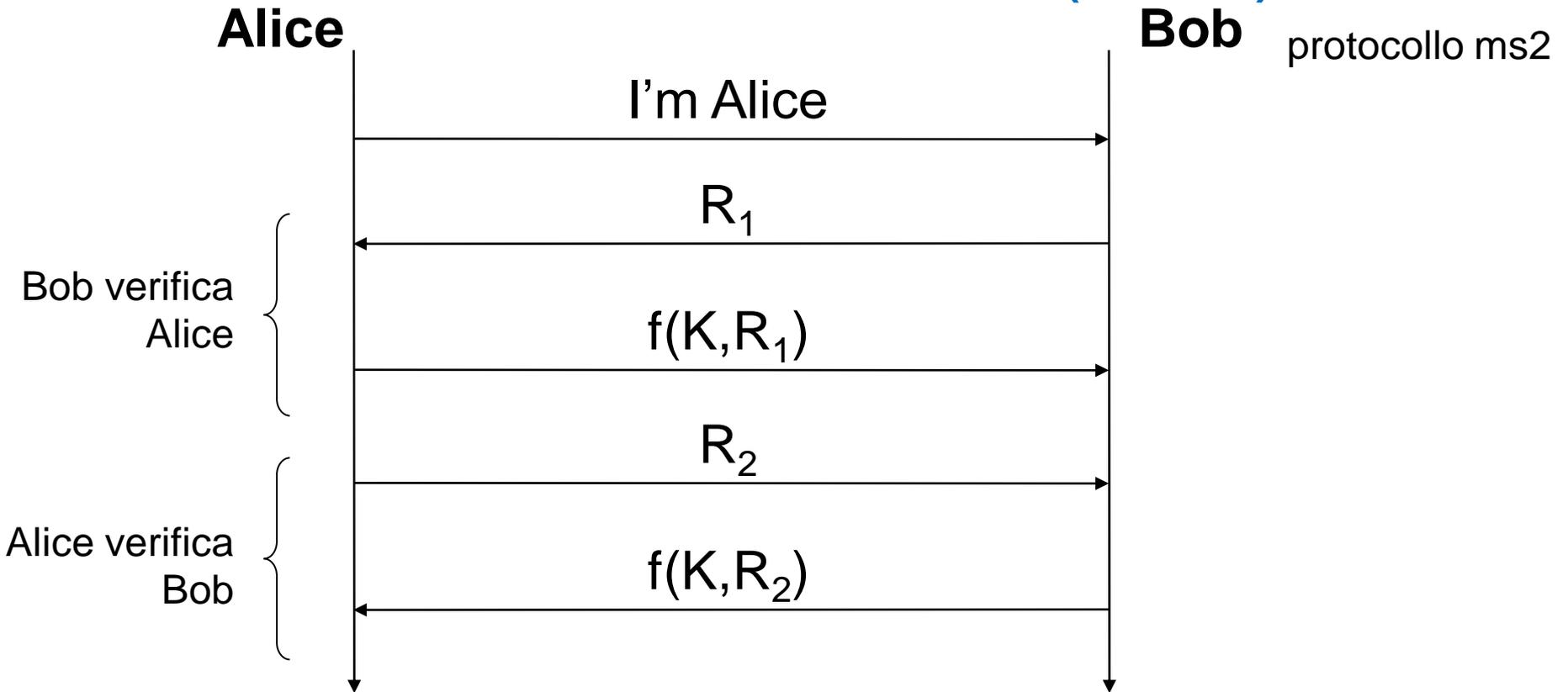


# reflection attack

- 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
  - cioè non devono far uso promiscuo della chiave
- es. usare chiavi differenti nei due versi
  - es. totalmente differenti
  - es. chiavi derivate
    - es.  $K$  e  $K+1$ , o  $K$  e  $h(K)$
- es. usare challenge strutturalmente differenti
  - es.  $R_1$  pari e  $R_2$  dispari
  - es. concatenare il nome o il ruolo
    - es. Alice| $R_1$  e Bob| $R_2$
    - es. server| $R_1$  e client| $R_2$

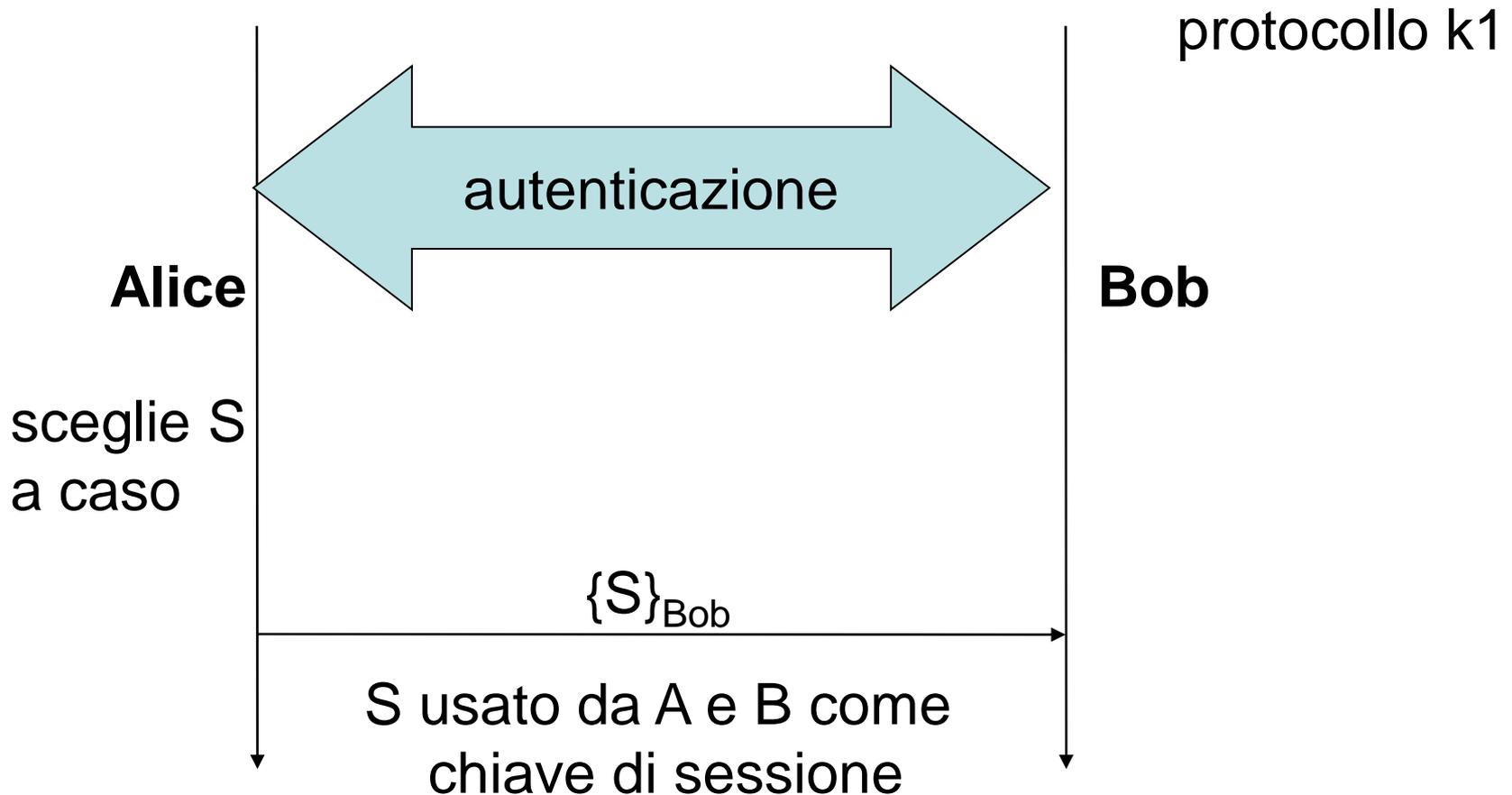
# cifratura dei dati

# chiavi di autenticazione e chiavi di sessione

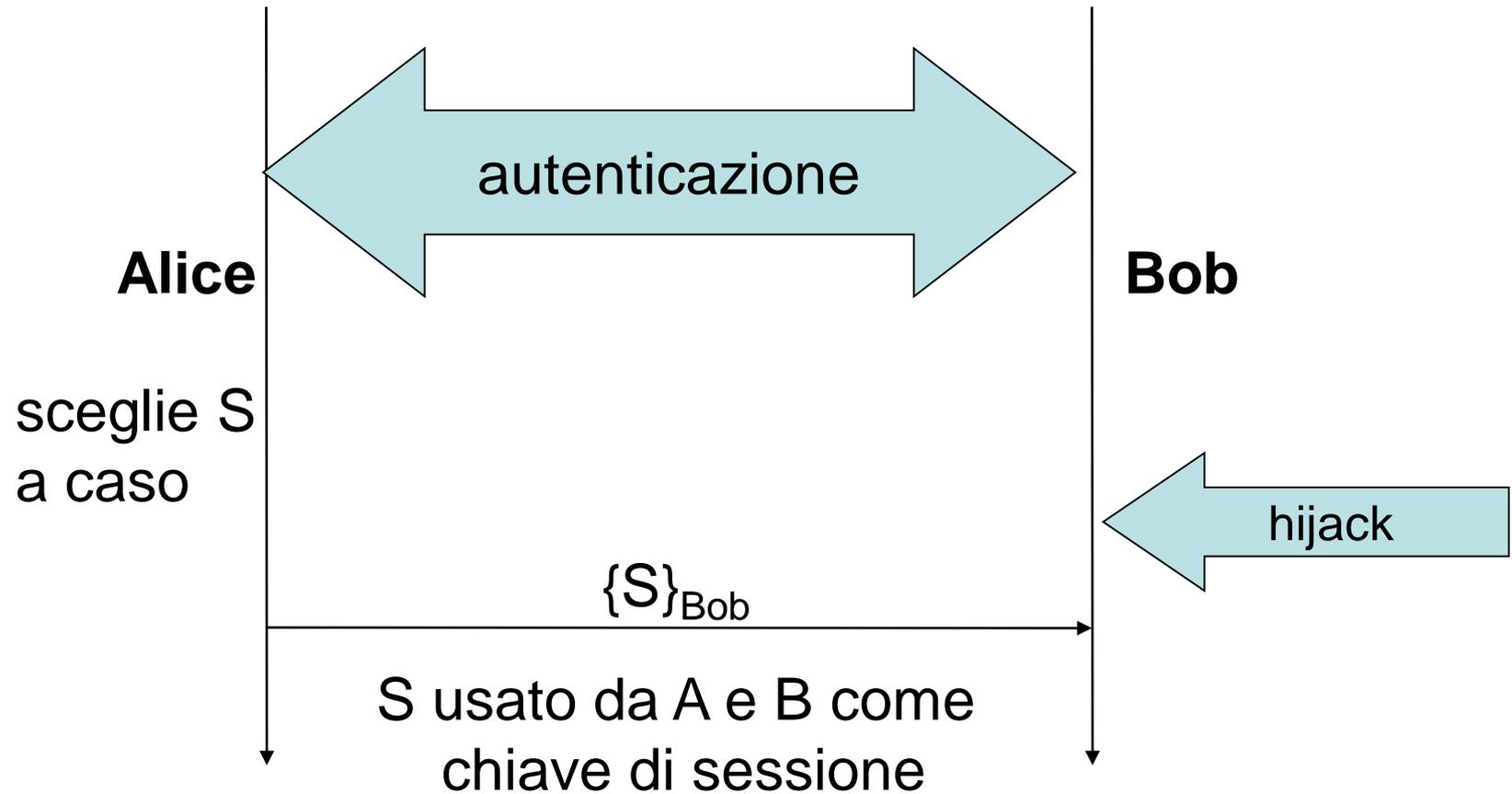
- la chiave simmetrica usata per la cifratura è detta **session key** (chiave di sessione)
- la session key è bene che sia diversa dalla/e chiave/i usata/e per l'autenticazione
  - la/le chiave/i per l'autenticazione deve/devono durare nel tempo (**long term secret**)
  - la chiave di sessione si usa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

# autenticazione e sessione

- trova la vulnerabilità



# hijack

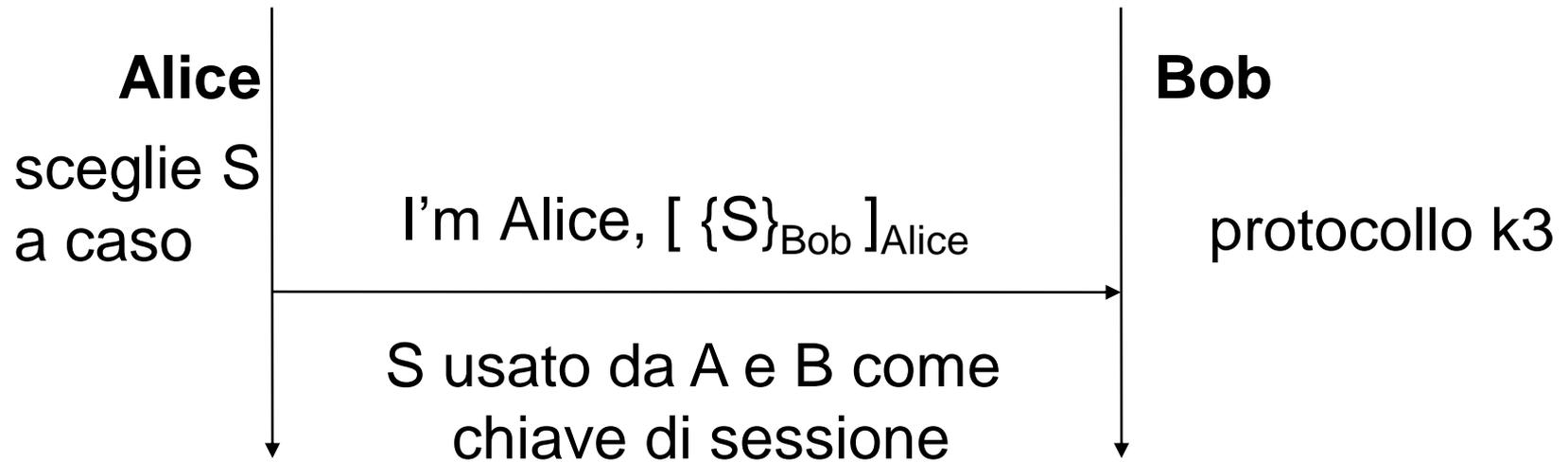


- nulla mi assicura che  $S$  sia stato generato da Alice

# 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  $k1'$ 
  - es. autenticazione mutua con due nonce  $R1$  e  $R2$
  - $\{S|R1|R2\}_{Bob}$
- protocollo  $k2$ 
  - come  $k1$  ma la chiave è autenticata in qualche modo
  - es.  $[\{S\}_{Bob}]_{Alice}$  oppure  $K\{S\}$
  - non vulnerabile ad hijacking
- in realtà  $k2$  è ridondante
  - lo scambio di  $[\{S\}_{Bob}]_{Alice}$  fa anche mutua autenticazione

# autenticazione mutua e scambio di session key in un solo messaggio



- Alice sa che solo Bob può essere la controparte poiché solo Bob può decifrare  $\{S\}_{Bob}$
- Bob è sicuro che solo Alice può essere la controparte poiché  $S$  è firmata da Alice
- un attaccante potrebbe fare replay del messaggio ma non potrebbe decifrare la sessione seguente
  - Bob non deve considerare la sola apertura della sessione come necessariamente proveniente da Alice

# le chiavi di sessione devono essere usate “poco”

- sessioni con molti dati sono un problema
  - le tecniche di crittoanalisi hanno bisogno di una certa quantità di ciphertext per trovare la chiave
  - molti dati semplificano la crittoanalisi
- la chiavi di sessione possono essere cambiate periodicamente
- **key rollover**: è il cambiamento periodico della chiave di sessione
  - la chiave va cambiata prima che si raggiunga una quantità di ciphertext che renda possibile l'attacco crittoanalitico

# key rollover e master secret

- rinegoziazione della nuova chiave di sessione: inefficiente
- nuove chiavi di sessioni calcolate: efficiente
  - aggiornate in maniera sincrona da entrambe le parti
- approccio tipico: chiavi di sessione calcolate da un **master secret** combinando vari strumenti
  - es. contatori, shuffling, hash, crittografia, ecc
  - es.  $M$ : master secret, chiave  $i$ -esima:  $K_i = h(i/M)$
- 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)

perfect forward secrecy e  
chiavi effimere

# problema 1: intercettazioni legali

- supponi che per legge esista un repository “fidato” di tutte le chiavi private
  - *key escrow*: terza parte depositaria delle chiavi
- la magistratura può autorizzare una intercettazione e richiedere le corrispondenti chiavi private
- la tecnologia dovrebbe permettere alla magistratura di...
  - ...decifrare le trasmissioni a partire dalla data di autorizzazione
  - ...impedire di decifrare trasmissioni precedenti alla data di autorizzazione
    - poiché le autorizzazioni di intercettazione non sono retroattive

# problema 2: pubblicazione delle chiavi private

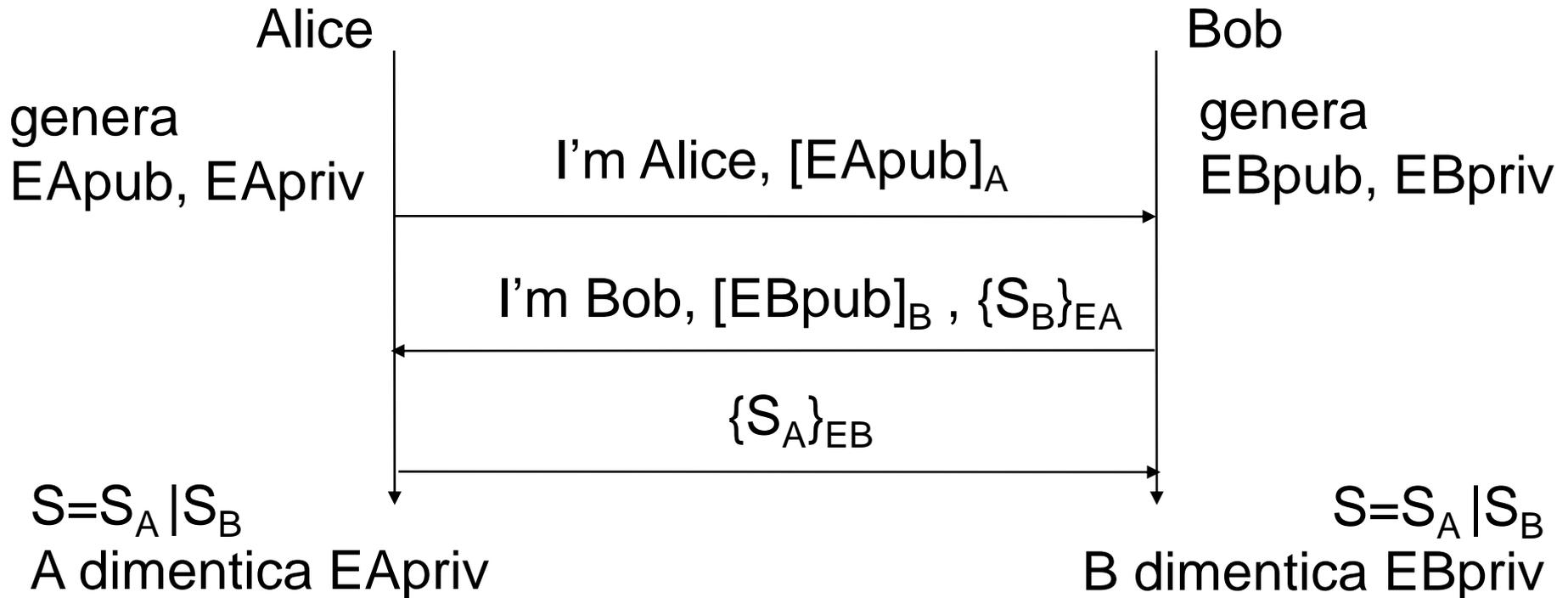
- dopo che le chiavi sono state usate possono essere pubblicate senza alterare la confidenzialità delle trasmissioni precedenti?
- la domanda è importante: considera i seguenti casi pratici
  - supponi che un eavesdropper **registri** una trasmissione e poi ottenga la/le chiavi di A, B o di entrambi, può risalire al contenuto della trasmissione?
  - tipicamente le chiavi hanno una **scadenza** dopo la quale vanno cambiate, alla scadenza le chiavi private sono pubblicabili?
  - la magistratura può ottenere il contenuto di registrazioni precedenti all'autorizzazione?
    - una volta ottenute le chiavi dal key escrow?

# (perfect) forward secrecy (PFS)

- un protocollo si dice avere la proprietà PFS se **non permette di decifrare una trasmissione registrata pur avendo i segreti a lungo termine** (chiavi di autenticazione) a disposizione.
- analizza i protocolli precedenti rispetto a questa proprietà

# chiavi effimere

- chiavi (RSA asimmetriche) **effimere**
  1. **generate** prima di ogni scambio di chiave di sessione
  2. scambiate (nella parte pubblica)
  3. **usate** per lo scambio di chiave di sessione
  4. **dimenticate**



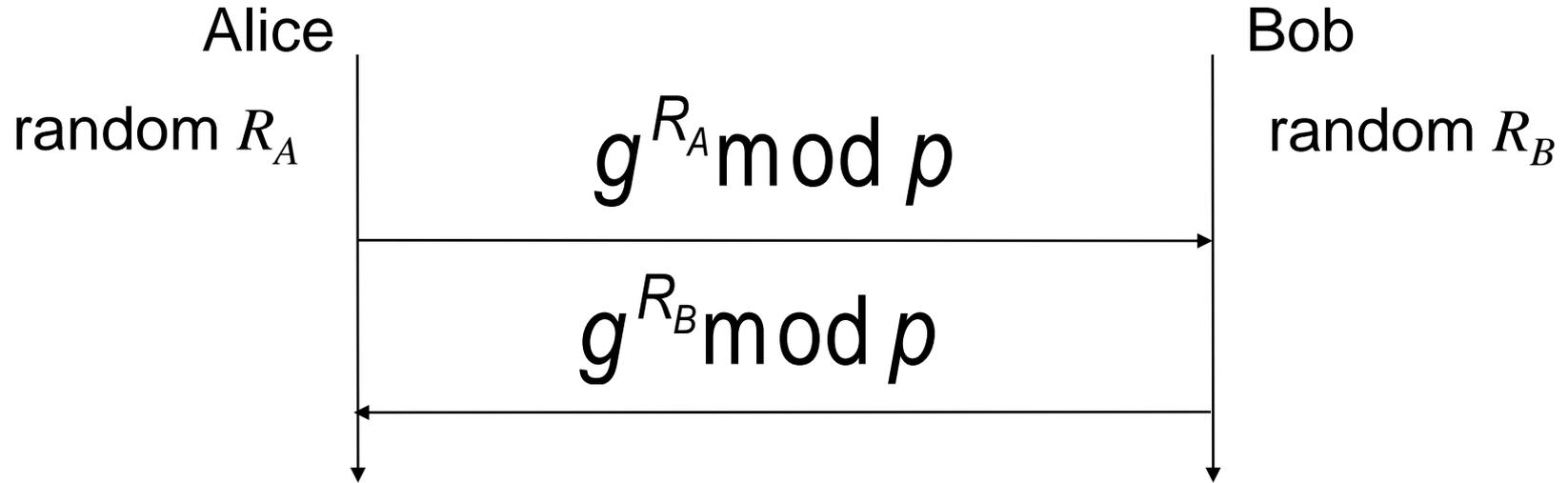
# chiavi effimere

- chiavi (RSA asimmetriche) **effimere**
  1. **generate** prima di ogni scambio di chiave di sessione
  2. scambiate (nella parte pubblica)
  3. **usate** per lo scambio di chiave di sessione
  4. **dimenticate**
- una chiave effimera esiste in memoria solo per una frazione di secondo e poi viene dimenticata
- la generazione di chiavi RSA è inefficiente
- si usa il protocollo di scambio Diffie-Hellman

# diffie-hellman

- è un protocollo di scambio di chiave di sessione che gode di PFS
- basato sul problema del logaritmo discreto
  - “il logaritmo mod  $p$  in base  $g$  è difficile da calcolare”
  - dimostrato essere equivalente alla fattorizzazione in numeri primi (su cui si basa RSA)
- $p$  e  $g$  due numeri pubblicamente noti
  - per garantire la sicurezza  $p$  e  $g$  devono avere delle proprietà particolari

# diffie-hellman



$$\left(g^{R_B}\right)^{R_A} = g^{R_A R_B} \bmod p \qquad \left(g^{R_A}\right)^{R_B} = g^{R_A R_B} \bmod p$$

- A è B prendono come chiave di sessione  $g^{R_A R_B}$
- se i numeri random  $R_A$  e  $R_B$  sono dimenticati il protocollo gode di PFS

# diffie-hellman

- notare che è solo per lo scambio di chiavi
- l'autenticazione necessita di altro
  - es. segreti a lungo termine RSA

esercizio:  
scrivere la versione di diffie-hellman con autenticazione RSA.

# PFS e impossibilità dello sniffing

- per decifrare una sessione bisogna decifrare la chiave di sessione  $S$
- $S$  è stata cifrata con chiavi effimere mai pubblicate
- quindi: **non c'è modo di sniffare una sessione con PFS senza un attacco crittanalitico**
- **... neanche conoscendo i segreti a lungo termine!**
- questo è vero anche per sessioni registrate e segreti a lungo termine rivelati in seguito

# PFS: l'intercettazione legale

- A e B devono avere le chiavi private presso un escrow E (notaio)
- C è una autorità preposta all'intercettazione
- la magistratura impone ad E di fornire a C le chiavi private di A e B
- C fa MitM e impersona A nei confronti di B e B nei confronti di A
  - C fa uso delle chiavi ottenute da E
  - le due sessione A-C e C-B sono del tutto normali
- C fa da passacarte tra le due sessioni
  - C passa i dati ricevuti da A a B e viceversa
  - A e B non si accorgono di nulla
  - C può vedere tutto il traffico