

# Reti di Elaboratori

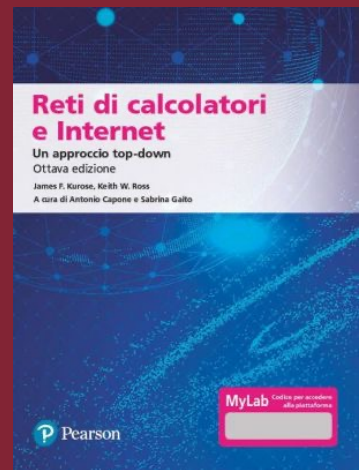
Sicurezza nelle reti – comunicazione tra due entità



SAPIENZA  
UNIVERSITÀ DI ROMA

Alessandro Checco

[alessandro.checco@uniroma1.it](mailto:alessandro.checco@uniroma1.it)



Capitolo 8

# Appelli

- 5 Giugno 2024 14pm-17pm
- 3 Luglio 2024 9am-12am

# Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- Integrità dei messaggi, autenticazione

# Che cos'è la sicurezza della rete?

**riservatezza:** solo il mittente, il destinatario previsto dovrebbe poter "comprendere" il contenuto del messaggio

- mittente cifra il messaggio
- il destinatario decifra il messaggio

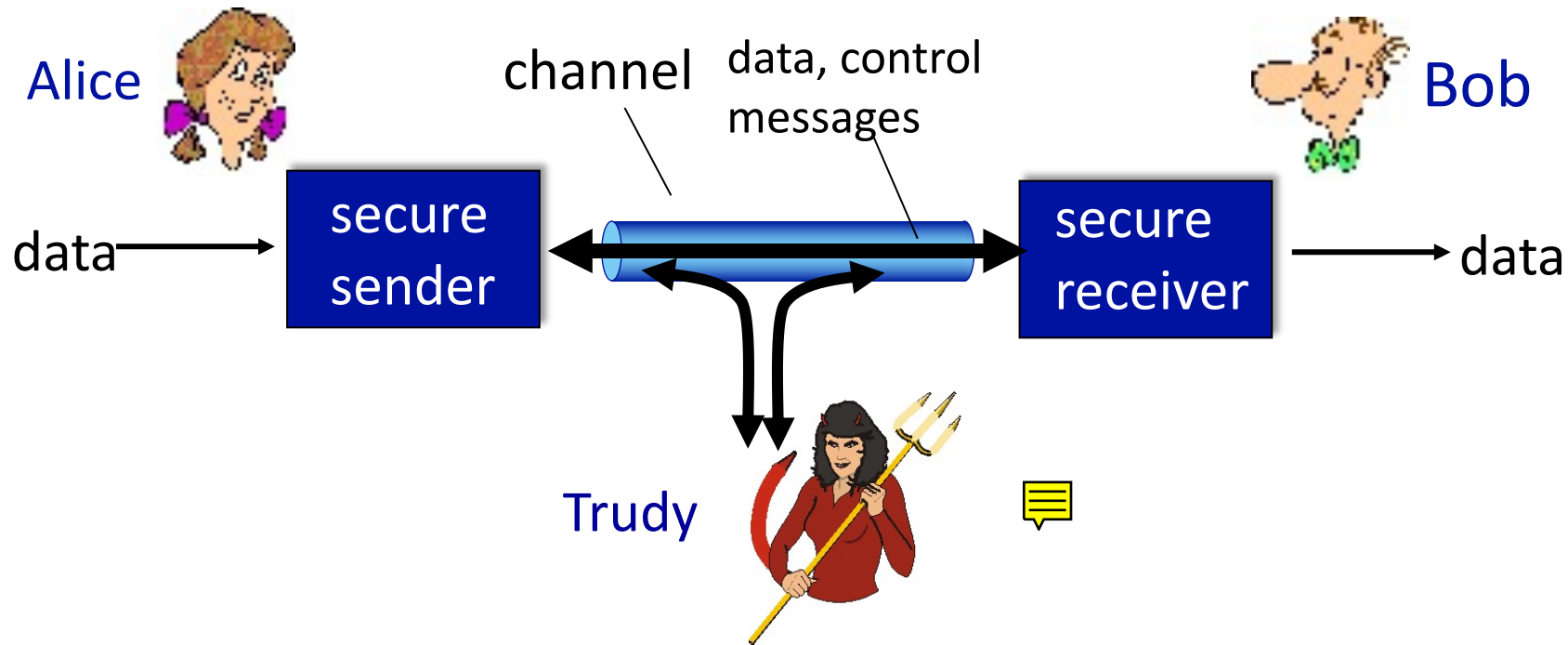
**autenticazione:** il mittente e il destinatario vogliono confermare l'identità l'uno dell'altro

**integrità del messaggio:** il mittente e il destinatario vogliono garantire che il messaggio non venga alterato (in transito o successivamente) senza poter rilevare tale manomissione 

**accesso e disponibilità:** i servizi devono essere accessibili e disponibili per gli utenti

# Amici e nemici: Alice, Bob, Trudy

- ben noti nel mondo della sicurezza di rete
- Bob, Alice vogliono comunicare “in sicurezza”
- Trudy (intruso) può intercettare, cancellare, aggiungere o modificare messaggi



# Amici e nemici: Alice, Bob, Trudy

Chi potrebbero essere Bob e Alice?

- ... esseri umani chiamati Bob e Alice!
- Browser/server Web per transazioni elettroniche (ad es. acquisti online)
- cliente/server bancario online
- Server DNS
- Router BGP che scambiano aggiornamenti della tabella di routing
- altri esempi?

# Intrusi e malintenzionati

D: Cosa può fare un malintenzionato?

R: Molto!

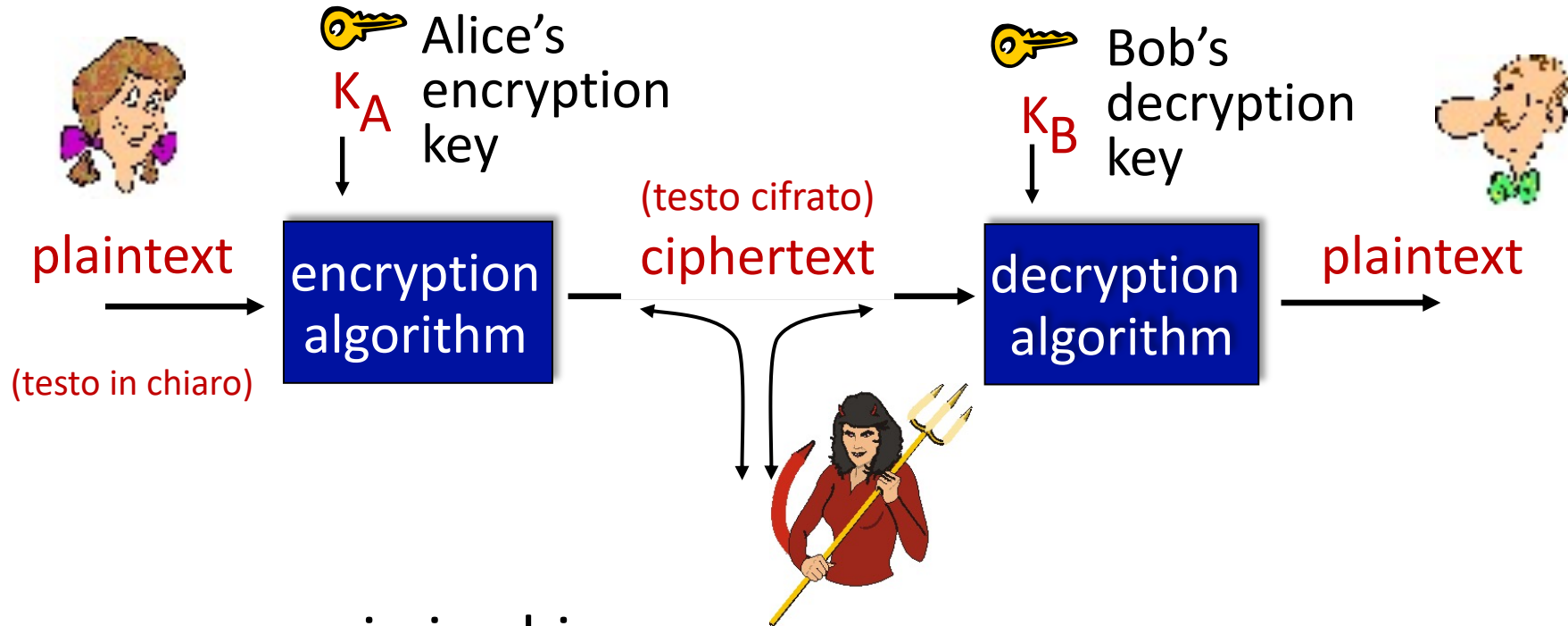
- **origliare**: intercettare i messaggi
- **inserire** attivamente messaggi nella connessione
- **impersonificazione**: falsificare (spoof) l'indirizzo di origine nel pacchetto (o qualsiasi campo nel pacchetto)
- **hijacking**: “prendere il controllo” della connessione in corso rimuovendo il mittente o il destinatario, inserendosi al loro posto
- **denial of service**: impedire che il servizio venga utilizzato da altri (ad esempio, sovraccaricando le risorse)

# Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- **Principi di crittografia**
- Integrità dei messaggi, autenticazione



# Il linguaggio della crittografia



$m$ : messaggio in chiaro

$K_A(m)$ : messaggio cifrato, usando la chiave  $K_A$

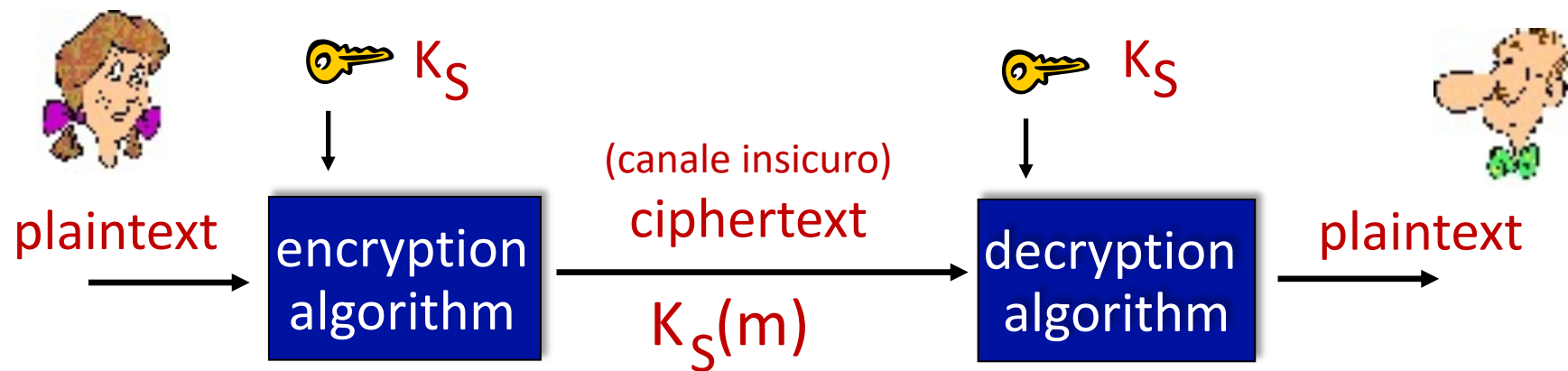
$m = K_B(K_A(m))$



# Compromettere uno schema di crittografia

- **Attacco solo con testo cifrato:** Trudy ha un testo cifrato che può analizzare
- **due approcci:**
  - forza bruta: cerca tra tutte le chiavi (costoso)
  - analisi statistica
- **Attacco con testo in chiaro noto:** Trudy conosce del testo in chiaro corrispondente al testo cifrato
  - *ad esempio*, nel cifrario monoalfabetico, Trudy determina gli accoppiamenti per a,l,i,c,e,b,o,
- **attacco con testo in chiaro selezionato:** Trudy può ottenere testo cifrato per un particolare testo in chiaro di sua scelta
  - ad esempio, Trudy può usare il servizio e osservare il risultato
  - header di un protocollo sempre uguali (o saluti all'inizio di un messaggio)

# Crittografia a chiave simmetrica

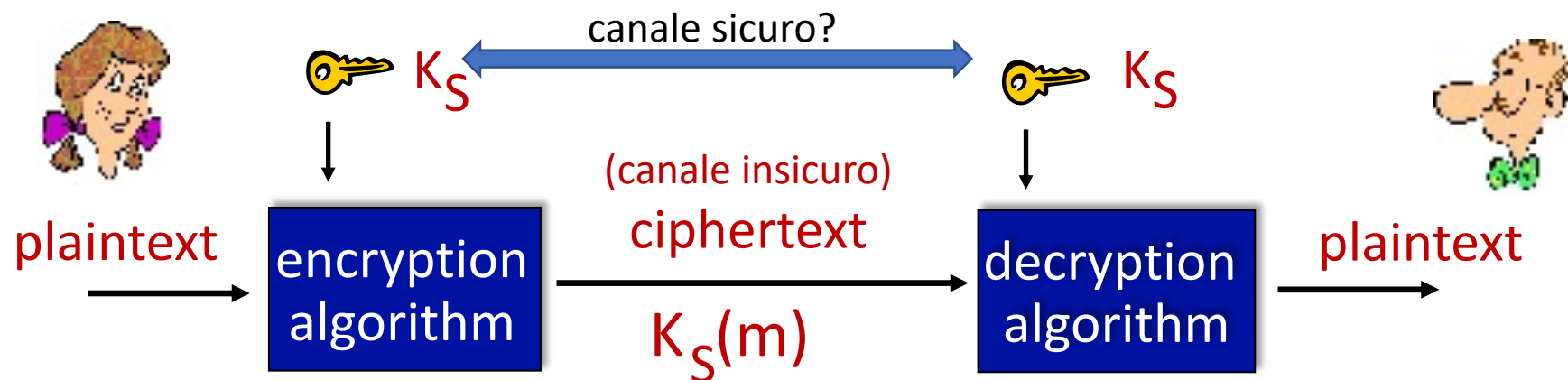


**crittografia a chiave simmetrica:** Bob e Alice condividono la stessa chiave (simmetrica):  $K$

- *ad esempio*, la chiave è conoscere il modello di sostituzione nel cifrario di sostituzione monoalfabetico

D: in che modo Bob e Alice concordano sul valore della chiave?

# Crittografia a chiave simmetrica



**crittografia a chiave simmetrica:** Bob e Alice condividono la stessa chiave (simmetrica):  $K$

- *ad esempio*, la chiave è conoscere il modello di sostituzione nel cifrario di sostituzione monoalfabetico

D: in che modo Bob e Alice concordano sul valore della chiave?

# Schema di crittografia semplice

*cifrario per sostituzione*: sostituire parti del messaggio con altre

- Cifrario monoalfabetico: sostituzione di una lettera con un'altra

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

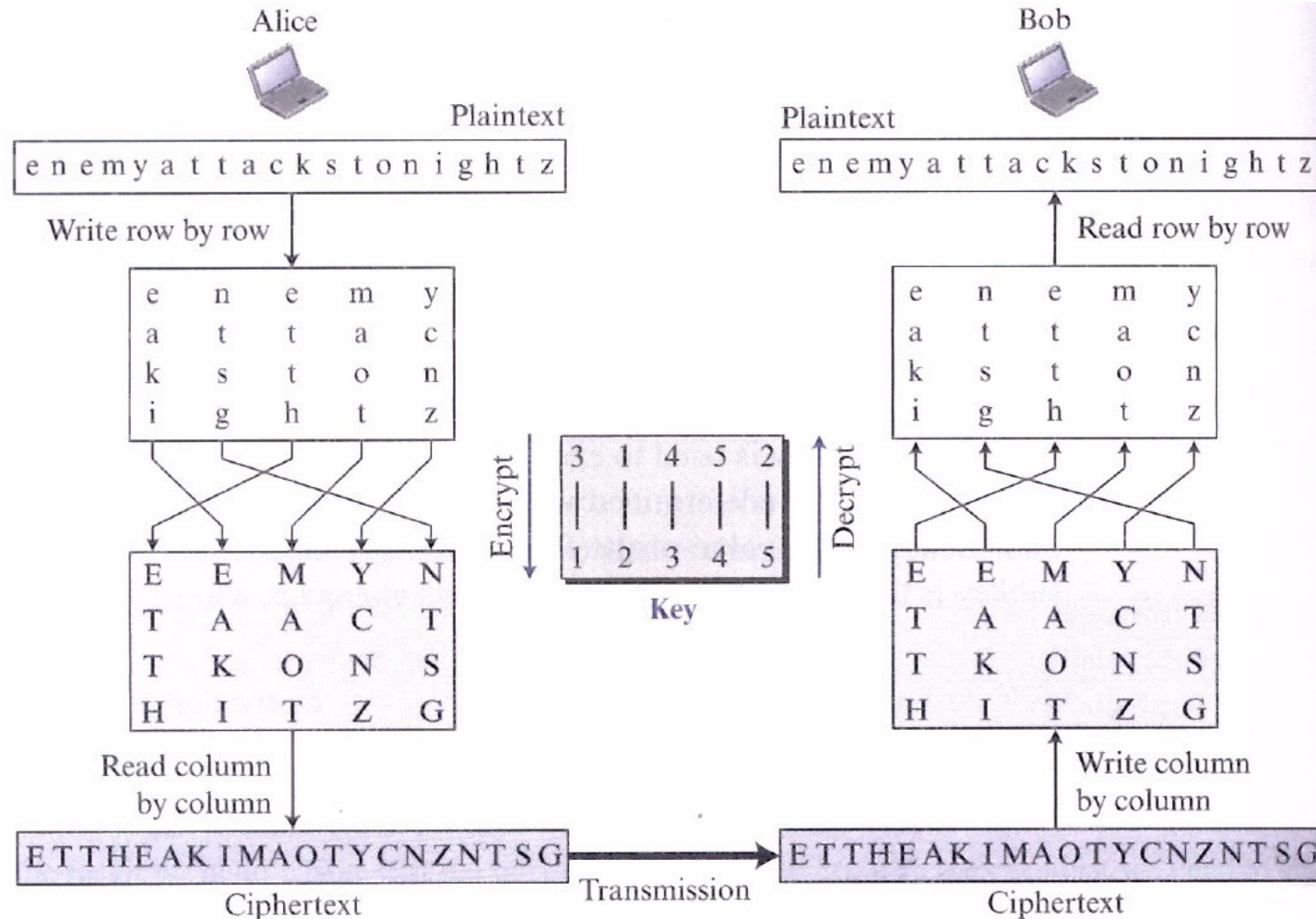
e.g.: Plaintext: bob. i love you. alice

ciphertext: nkn. s gktc wky. mgsbc

🔑 *Chiave*: mapping tra lettere dell'alfabeto (biie ttiva)

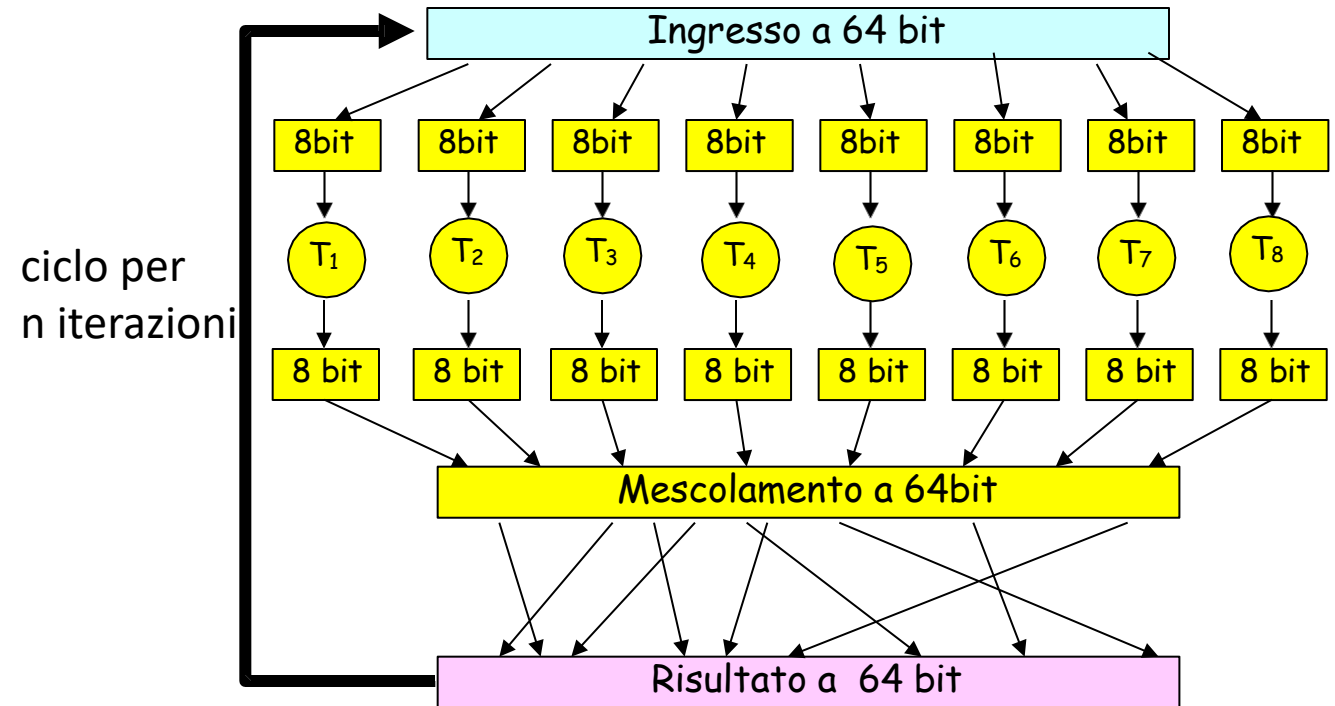
# Schema di crittografia più avanzato

## *cifrario per trasposizione*



# Cifrario a blocchi

- $T_i$ : tabelle di corrispondenza tra blocco in chiaro e blocco cifrato



- DES (Data Encryption Standard) è un esempio di cifrario a blocchi moderno usato per crittografia simmetrica

# Crittografia a chiave simmetrica: DES

## DES: standard di crittografia dei dati

- Standard di crittografia statunitense [NIST 1993]

Chiave simmetrica a 56 bit, input di testo in chiaro a 64 bit

- cifratura a blocchi con concatenazione a blocchi di cifratura
- quanto è sicuro DES?
  - Sfida DES: frase crittografata con chiave a 56 bit decrittografata (forza bruta) in meno di un giorno
  - nessun buon attacco di tipo analitico conosciuto
- rendere DES più sicuro:
  - 3DES: cifra 3 volte con 3 chiavi diverse



# AES: Advanced Encryption Standard

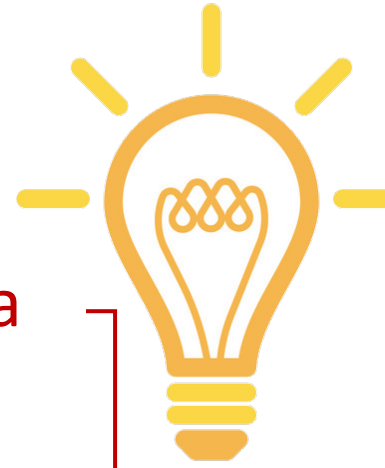
- standard NIST a chiave simmetrica, sostituisce DES (novembre 2001)
- elabora i dati in blocchi di 128 bit
- Chiavi a 128, 192 o 256 bit
- se un attacco a forza bruta impiegherebbe 1 secondo per DES, impiega 149 trilioni di anni per AES

# Enigma machine

- Lista di chiavi simmetriche per chi possiede la macchina
- Come è stato attaccato durante la seconda guerra mondiale:  
[https://www.youtube.com/watch?v=d2NWPG2gB\\_A](https://www.youtube.com/watch?v=d2NWPG2gB_A)
- Come lo attaccheremmo oggi
  - <https://github.com/mikepound/enigma>
  - <https://web.archive.org/web/20060720040135/http://members.fortunecity.com/jpeschel/gillog1.htm>
  - <https://www.cryptool.org/en/ct2/>



# Crittografia a chiave pubblica



## crittografia a chiave simmetrica

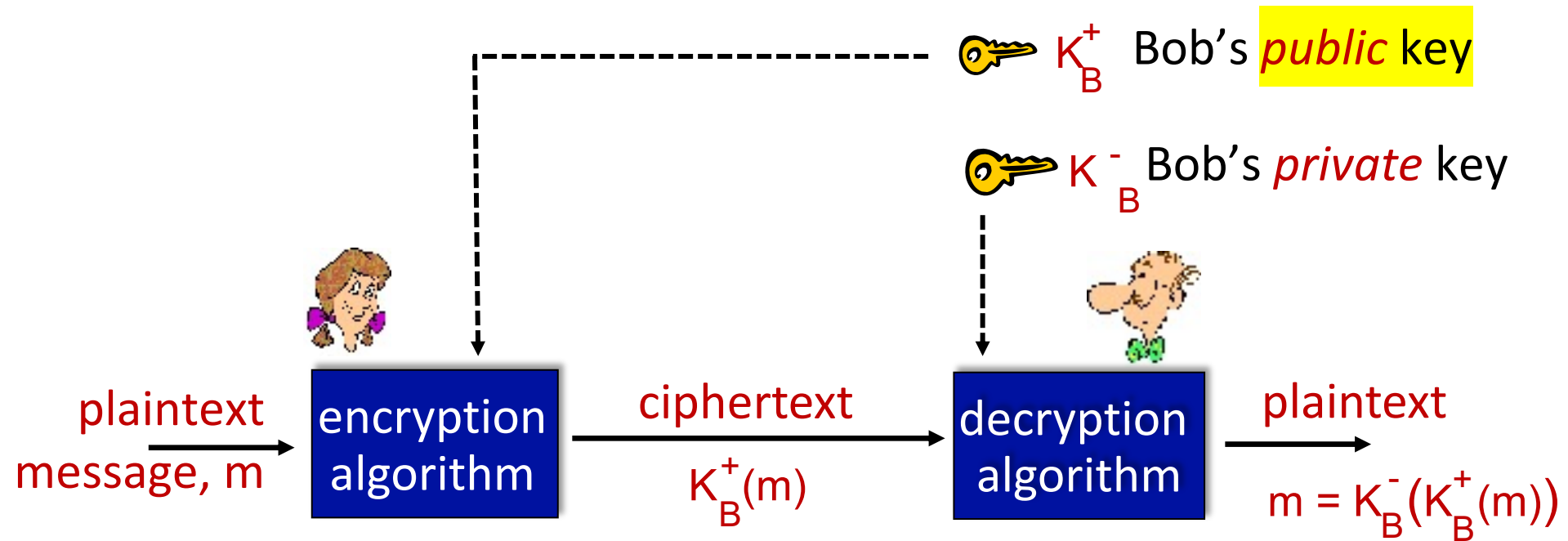
- richiede che il mittente e il destinatario conoscano la chiave segreta condivisa
- D: come accordarsi sulla chiave in primo luogo (in particolare se non possono “incontrarsi”)?



## crittografia a chiave pubblica

- *approccio radicalmente different* [Diffie-Hellman76, RSA78]
- mittente e destinatario *non* condividono una chiave segreta
- chiave di cifratura *pubblica* nota a *tutti*
- chiave di decrittazione nota soltanto al destinatario

# Crittografia a chiave pubblica



Una coppia di chiavi può essere usata per comunicare solo in una direzione


- Analogia chiave privata lucchetto pubblico
- Tutti i mittenti che vogliono inviare un messaggio a Bob possono usare **la stessa** chiave pubblica


# Algoritmi di cifratura a chiave pubblica

- La crittografia asimmetrica è solitamente impiegata per cifrare/decifrare quantità limitate di informazioni (es. la chiave di un cifrario simmetrico)
- Il testo in chiaro e il testo cifrato sono considerati **numeri interi**
- La cifratura e la decifratura sono funzioni matematiche
- Il testo cifrato può essere inteso come  $c = f(K_{\text{pubb}}, m)$
- Il testo in chiaro può essere inteso come  $m = g(K_{\text{priv}}, c)$
- RSA (Rivest, Shamir, Adleman): Algoritmo a chiave asimmetrica largamente diffuso

# RSA: scelta delle chiavi

1. Scegliere due numeri primi di valore elevato:  $p, q$ .  
(es.: 1024 bit ciascuno)
2. Calcolare  $n = pq$ ,  $z = (p-1)(q-1)$
3. Scegliere  $e$  (con  $e < n$ ) tale che non abbia fattori in comune con  $z$ . ( $e, z$  sono “primi fra loro”)
4. Scegliere  $d$  tale che  $e \cdot d - 1$  sia esattamente divisibile per  $z$ . (in altre parole:  $e \cdot d \bmod z = 1$ )
5. La chiave *pubblica* è  $(n, e)$ , quella *privata* è  $(n, d)$   

  
 $K_{\text{pubb}}$

  
 $K_{\text{priv}}$

# RSA: cifratura e decifratura

Dati  $(n,e)$  e  $(n,d)$  calcolati come abbiamo appena visto,

- Per la codifica di  $m$  si calcola:

$$c = m^e \bmod n$$

- Per decifrare il messaggio ricevuto,  $c$ , si calcola:

$$m = c^d \bmod n$$

$$\text{Funziona! } m = \underbrace{(m^e \bmod n)}_c^d \bmod n$$

# Un esempio di RSA

Bob (destinatario) sceglie  $p=5$ ,  $q=7$ . Poi  $n=35$ ,  $z=24$ .

$e=5$  (così  $e$ ,  $z$  sono primi fra loro)

$d=29$  (così  $e \cdot d - 1$  è esattam. divisibile per  $z$ )

	<u>lettera</u>	<u>m</u>	<u>m<sup>e</sup></u>	<u>c = m<sup>e</sup> mod n</u>
cifratura:	I	12	1524832	17

	<u>c</u>	<u>c<sup>d</sup></u>	<u>m = c<sup>d</sup> mod n</u>	<u>lettera</u>
decifratura:	17	481968572106750915091411825223071697	12	I

per craccare la chiave privata (conoscendo solo  $(e,n)$  serve:

- fattorizzare  $n$  (**difficile per  $n$  grande**):  $n = 7 \times 5$  quindi  $p=5$ ,  $q=7$
- calcolare  $z$  togliendo 1 ad entrambi i fattori:  $z=6 \times 4 = 24$
- trovare il numero  $d$  tale che  $e \cdot d - 1$  è divisibile per  $z$



# Why does RSA work?

## Prerequisite: modular arithmetic

- $x \bmod n$  = remainder of  $x$  when divide by  $n$

- facts:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$$

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$$

- thus

$$(a \bmod n)^d \bmod n = a^d \bmod n$$

- example:  $x=14$ ,  $n=10$ ,  $d=2$ :

$$(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$$

$$x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$$

# Why does RSA work?

- must show that  $c^d \bmod n = m$ , where  $c = m^e \bmod n$
- fact: for any  $x$  and  $y$ :  $x^y \bmod n = x^{(y \bmod z)} \bmod n$ 
  - where  $n = pq$  and  $z = (p-1)(q-1)$
- thus,
$$\begin{aligned}c^d \bmod n &= (m^e \bmod n)^d \bmod n \\&= m^{ed} \bmod n \\&= m^{(ed \bmod z)} \bmod n \\&= m^1 \bmod n \\&= m\end{aligned}$$

# RSA: un'altra proprietà importante

Questa proprietà sarà **molto** utile dopo per la firma digitale

$$\underbrace{K_B^-(K_B^+(m))}_{\text{usa prima la chiave pubblica, seguita dalla chiave privata}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{usa prima la chiave privata, seguita dalla chiave pubblica}}$$

usa prima la  
chiave pubblica,  
seguita dalla  
chiave privata

usa prima la  
chiave privata,  
seguita dalla  
chiave pubblica

*il risultato è lo stesso!*

Why  $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$  ?

follows directly from modular arithmetic:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n\end{aligned}$$

# Perché RSA è sicuro?

- supponiamo di conoscere la chiave pubblica di Bob  $(n,e)$ .  
Quanto è difficile determinare  $d$ ?
- essenzialmente bisogna trovare i fattori di  $n$  senza conoscere i due fattori  $p$  e  $q$ 
  - Fattorizzare un numero di grandi dimensioni è difficile!

# RSA in pratica: chiavi di sessione

- l'esponenziazione in RSA è computazionalmente intensiva
- DES è almeno 100 volte più veloce di RSA
- si usa la crittografia a chiave pubblica per stabilire una connessione sicura, quindi comunicare in maniera sicura una seconda chiave (la chiave di sessione **simmetrica**) per crittografare i dati

## chiave di sessione, $K_s$

- Bob e Alice usano RSA per scambiare una chiave di sessione simmetrica  $K_s$  (cioè la chiave  $K_s$  viene crittografata con la chiave pubblica del destinatario)
- una volta che entrambi hanno  $K_s$ , usano la crittografia a chiave simmetrica per il resto della comunicazione

# Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- Integrità dei messaggi, **autenticazione**

# Autenticazione

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

**Protocol ap1.0:** Alice dice “I am Alice”



*quando non funziona?*

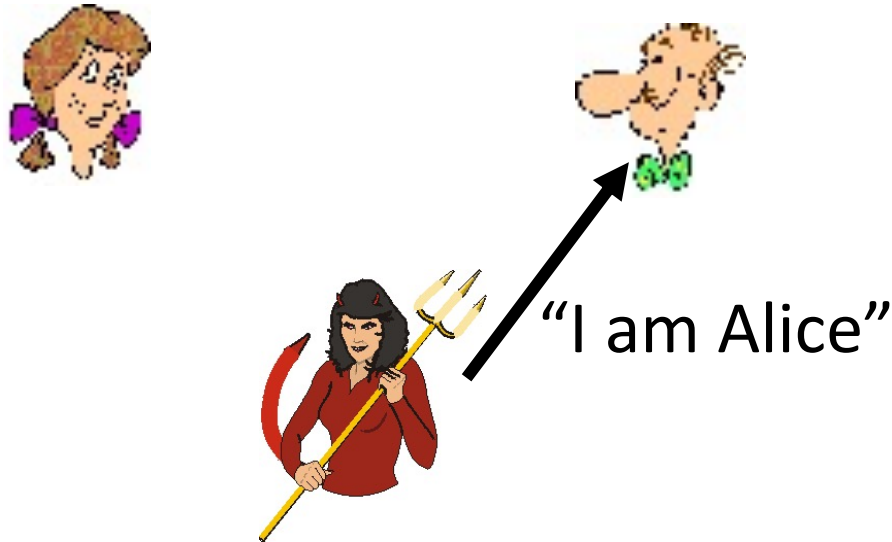




# Autenticazione

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

**Protocol ap1.0:** Alice dice “I am Alice”



*in una rete, Bob  
non può "vedere"  
Alice, quindi  
Trudy può  
semplicemente  
dichiarare di  
essere Alice*

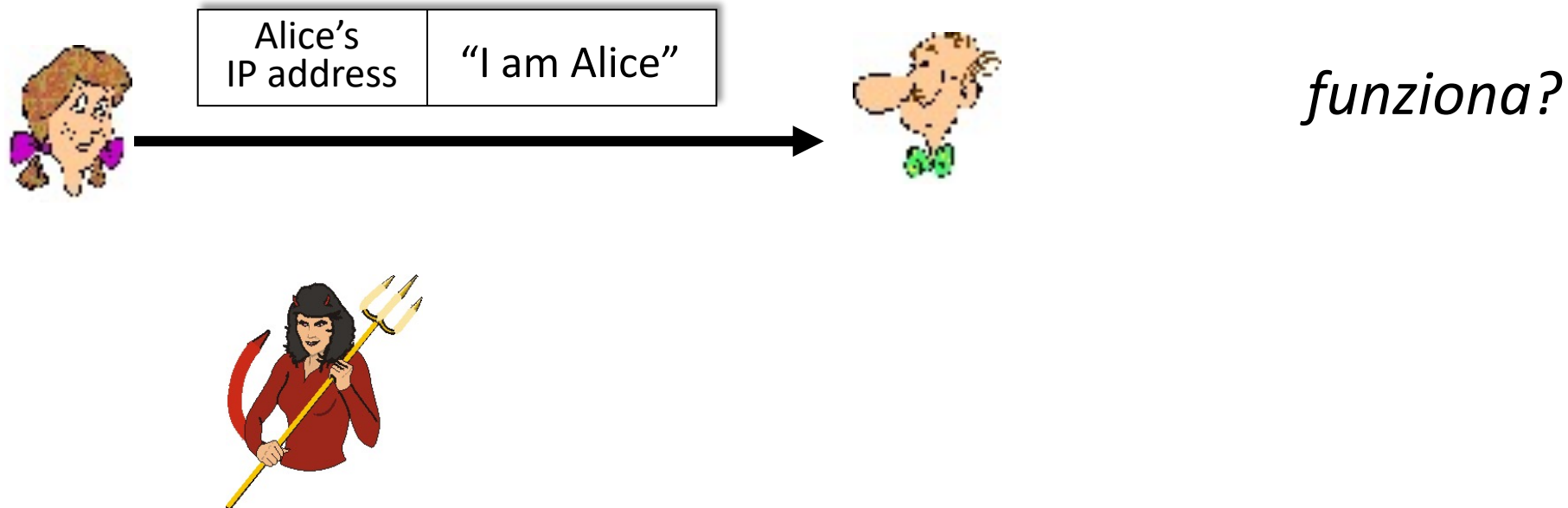


*"On the Internet, nobody knows you're a dog."*

# Autenticazione: secondo tentativo

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

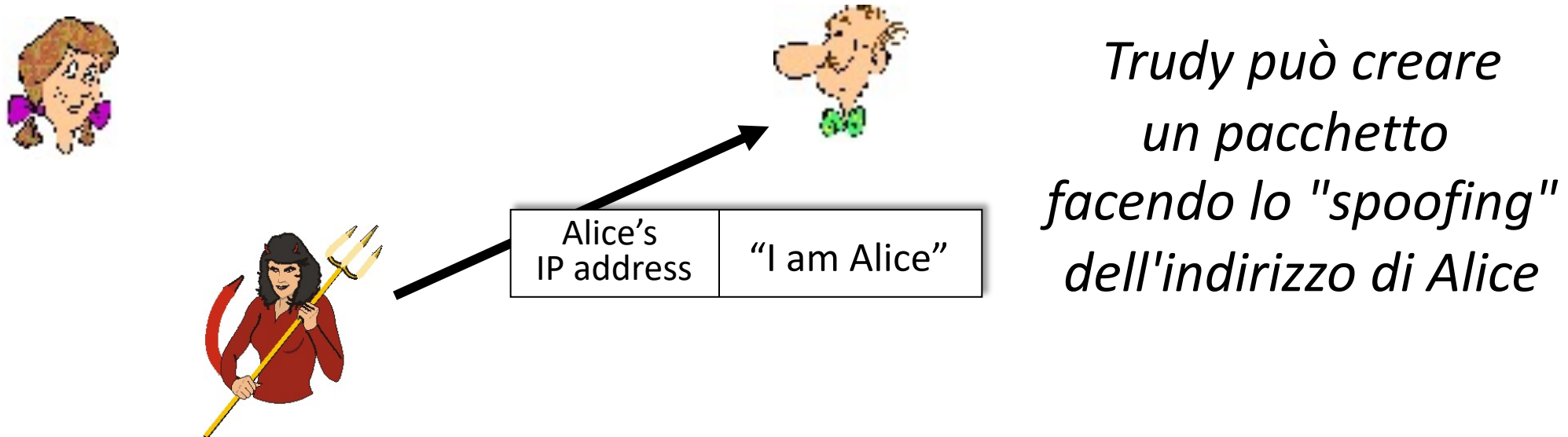
**Protocol ap2.0:** Alice dice “I am Alice” in un pacchetto IP contenente il suo indirizzo IP di origine



# Autenticazione: secondo tentativo

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

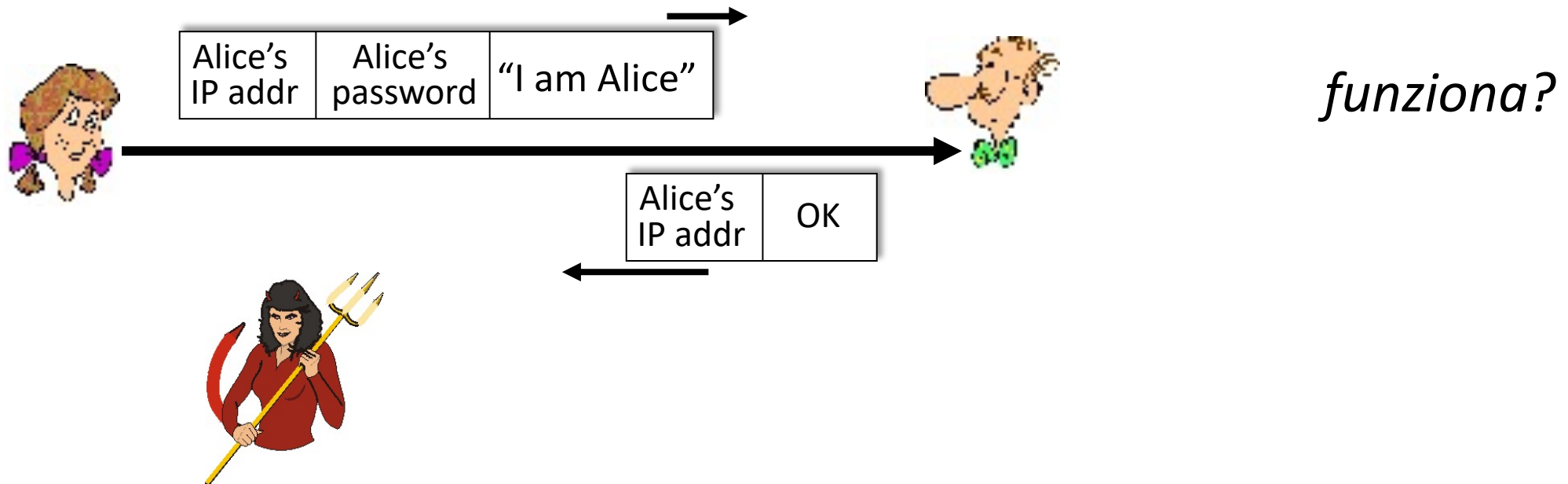
**Protocol ap2.0:** Alice dice “I am Alice” in un pacchetto IP contenente il suo indirizzo IP di origine



# Autenticazione: un terzo tentativo

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

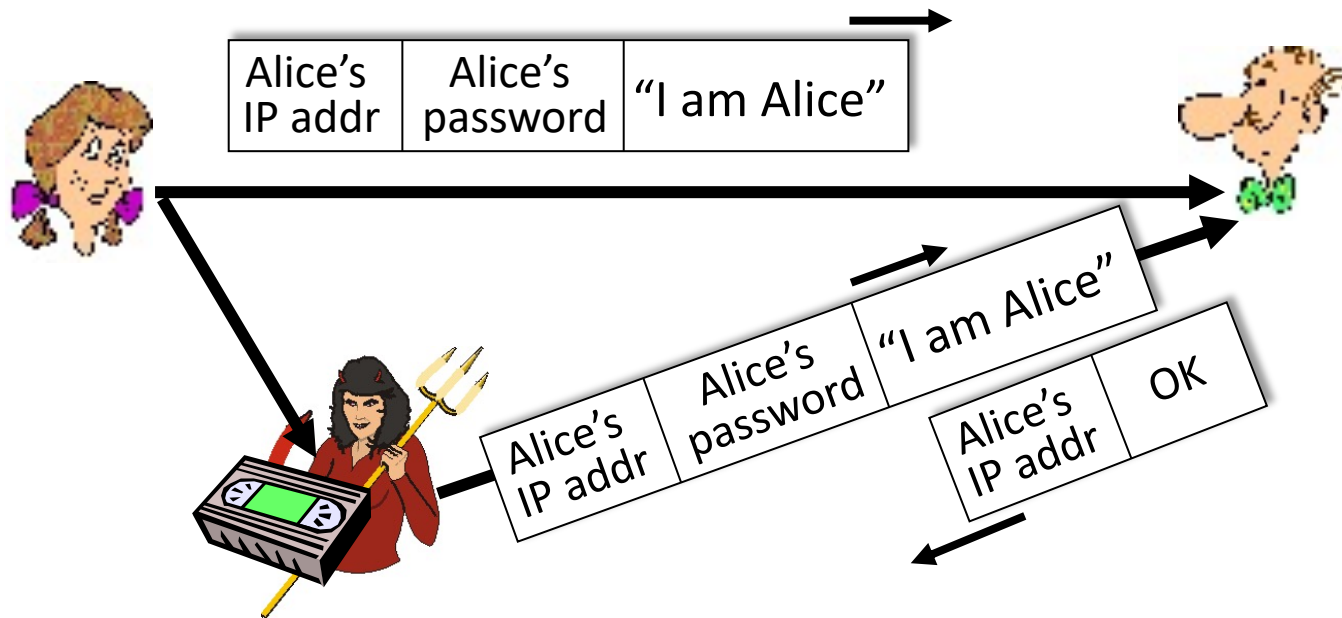
**Protocol ap3.0:** Alice dice "Io sono Alice" e invia una password segreta per "provarlo"



# Autenticazione: un terzo tentativo

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

**Protocol ap3.0:** Alice dice "Io sono Alice" e invia una password segreta per "provarlo"

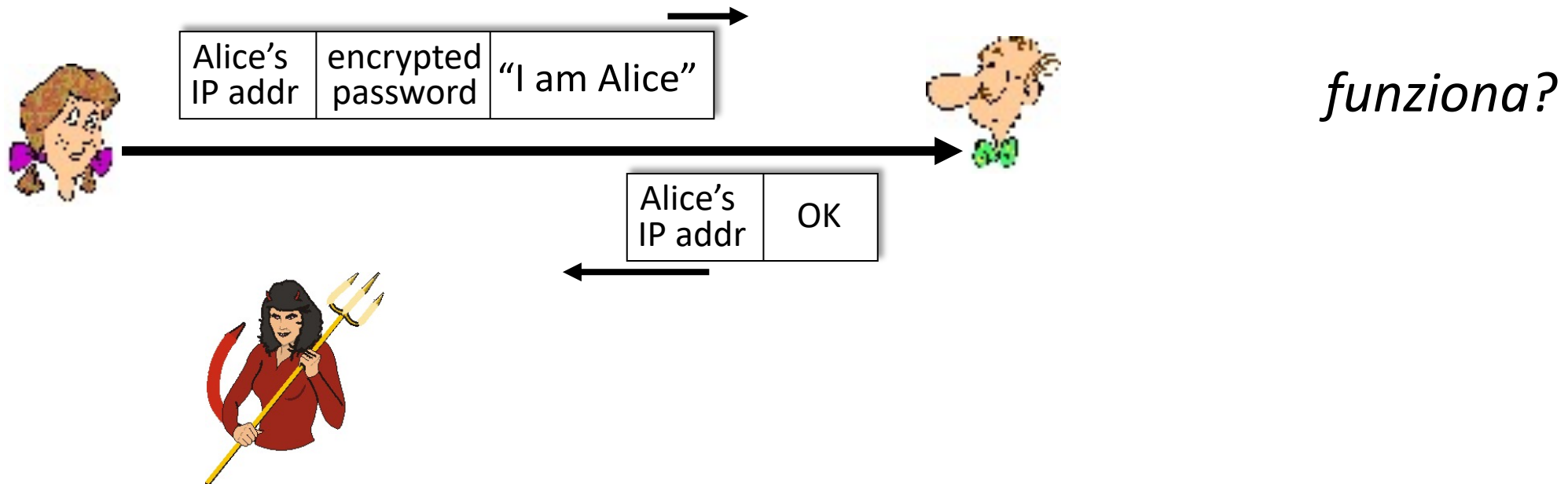


**playback attack:**  
*Trudy registra il pacchetto di Alice e più tardi lo riproduce a Bob*

# Autenticazione: un terzo tentativo modificato

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

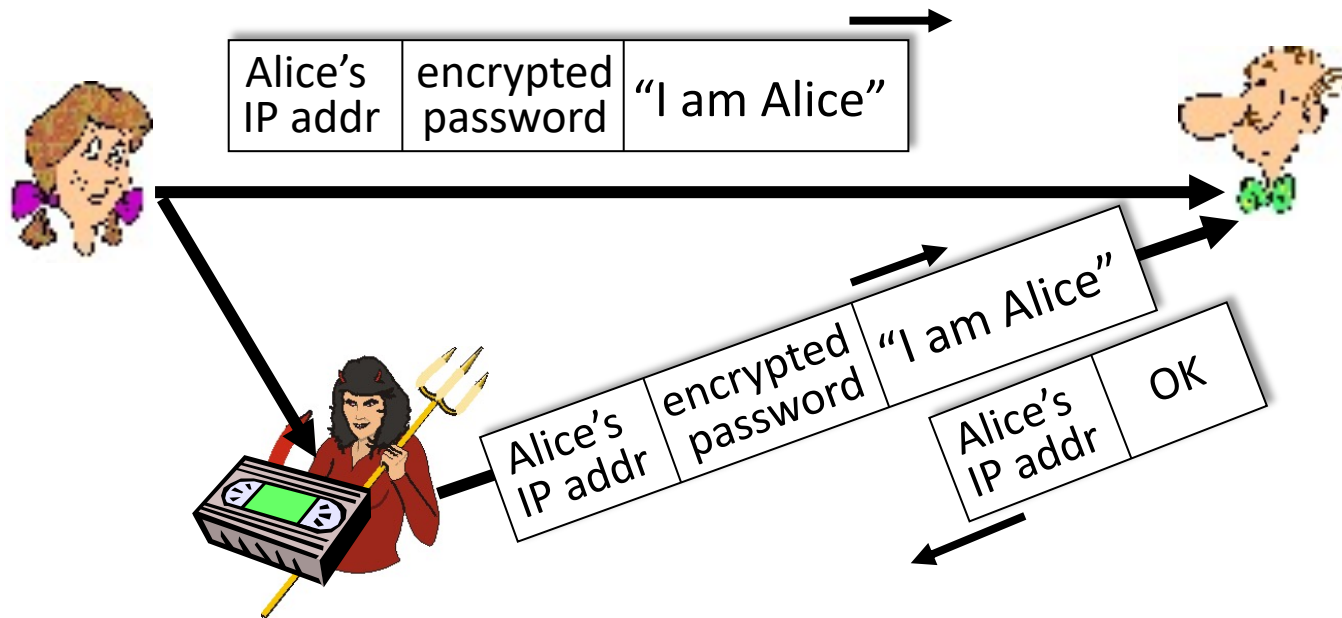
**Protocol ap3.0:** Alice dice "Io sono Alice" e invia la sua password segreta **crittografata** per "provarlo".



# Autenticazione: un terzo tentativo modificato

**Goal:** Bob vuole che Alice gli “dimostri” la sua identità

**Protocol ap3.0:** Alice dice "Io sono Alice" e invia la sua password segreta **crittografata** per "provarlo".



*il playback attack è ancora efficace!*

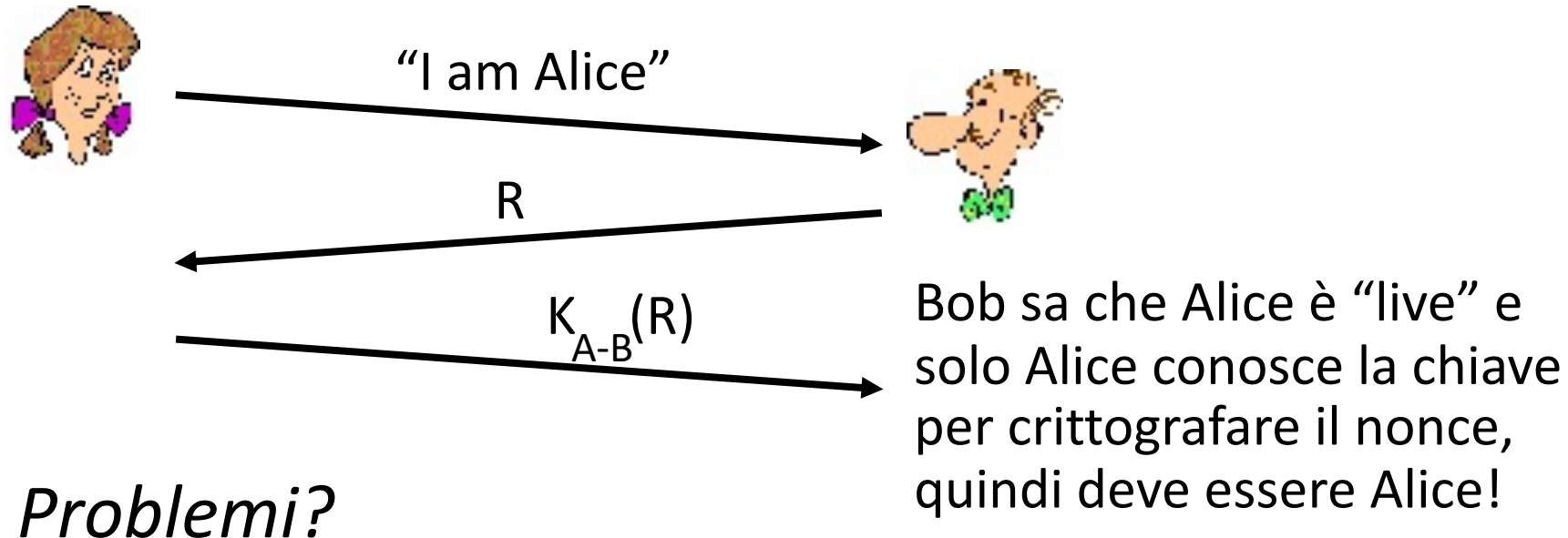
# Autenticazione: un quarto tentativo

**Goal:** evitare il playback attack

**nonce:** numero (R) usato **once-in-a-lifetime**

**protocol ap4.0:** per verificare che Alice è “live”, Bob manda ad Alice il nonce, R

- Alice deve restituire R, cifrato con chiave segreta condivisa





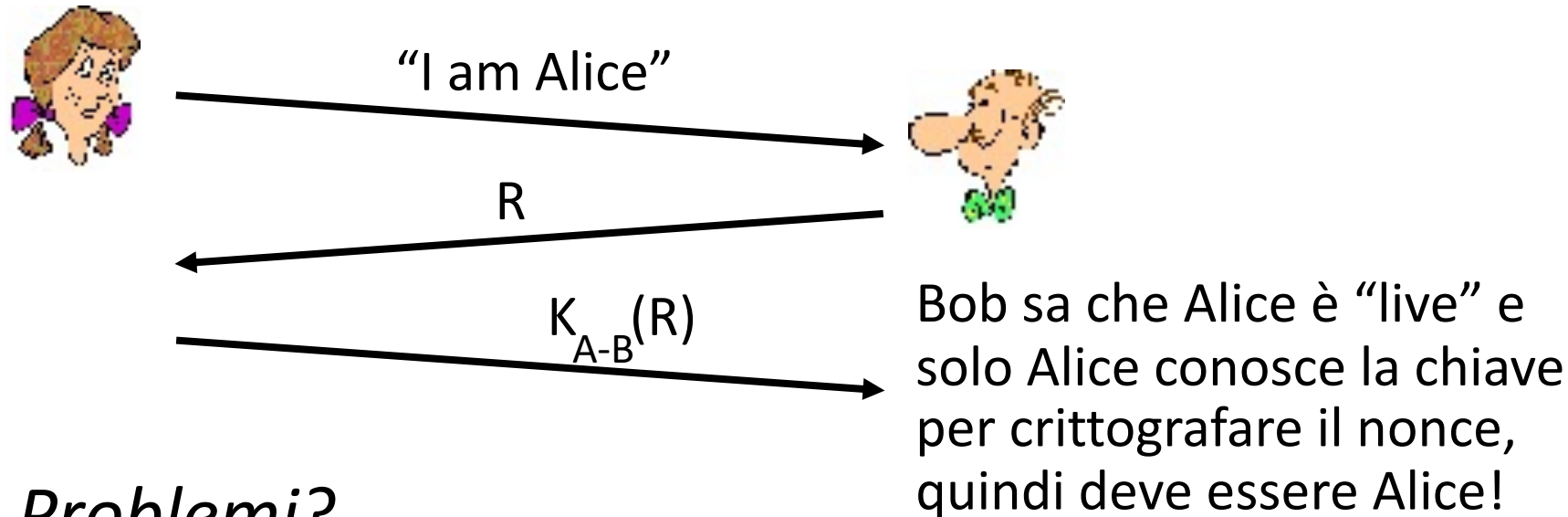
# Autenticazione: un quarto tentativo

**Goal:** evitare il playback attack

**nonce:** numero (R) usato **once-in-a-lifetime**

**protocol ap4.0:** per verificare che Alice è “live”, Bob manda ad Alice il nonce, R

- Alice deve restituire R, cifrato con chiave segreta condivisa



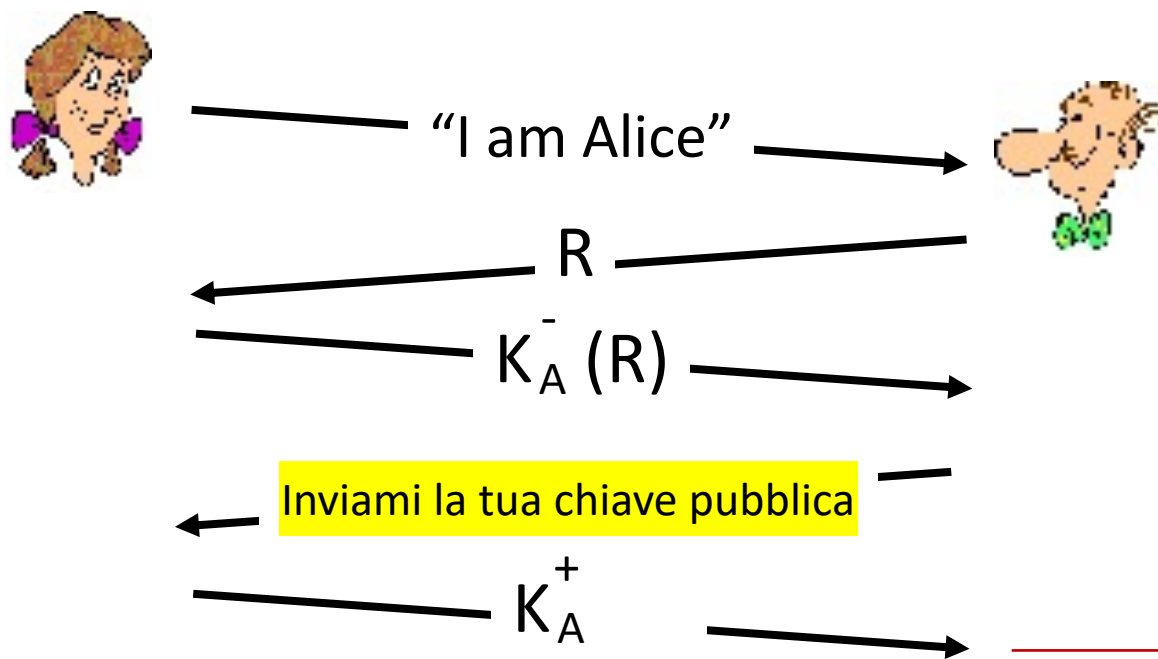
*Problemi?*

*bisogna usare un canale sicuro per scambiarsi la chiave in anticipo*

# Autenticazione: ap5.0

ap4.0 richiede una chiave simmetrica condivisa: possiamo autenticarci utilizzando tecniche a chiave pubblica?

**ap5.0:** usa nonce + crittografia a chiave pubblica



Bob calcola

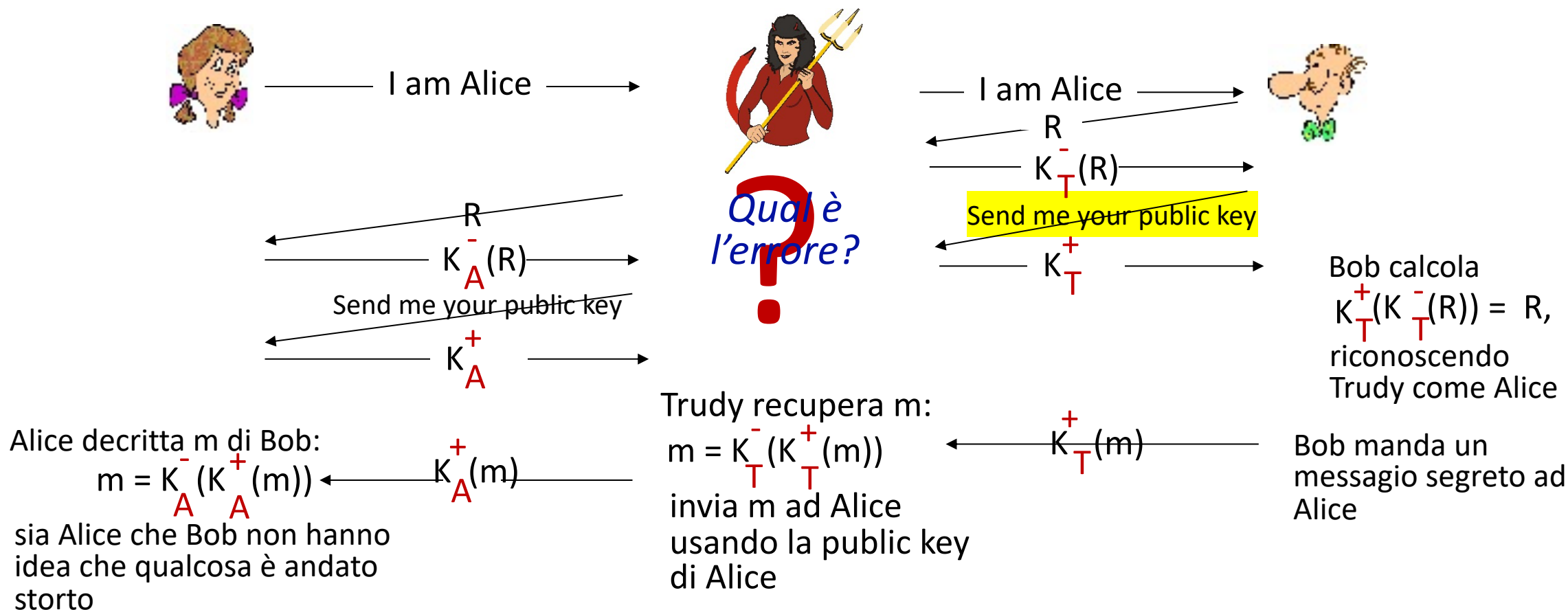
$$K_A^+ (K_A^-(R)) = R$$

e sa che solo chi conosce la chiave privata di Alice può creare un messaggio tale che:

$$K_A^+ (K_A^-(R)) = R$$

# Autenticazione: ap5.0 – c'è ancora un difetto!

**man in the middle attack:** Trudy impersona Alice quando comunica con Bob e impersona Bob quando comunica con Alice



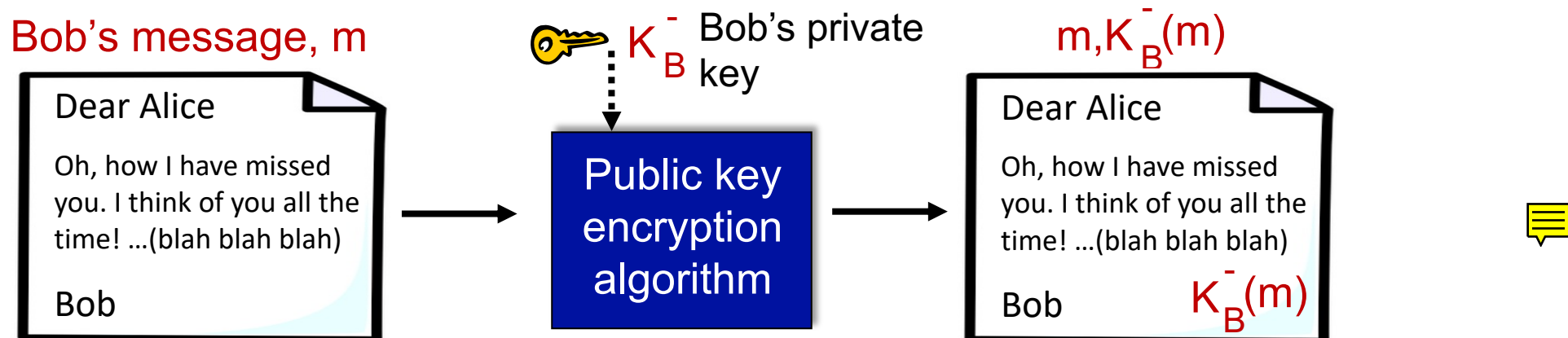
# Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- Integrità dei messaggi, autenticazione

# Firme digitali

## tecnica crittografica analoga alle firme autografe:

- il mittente (Bob) firma digitalmente il documento: è il proprietario/creatore del documento.
- *verificabile, non falsificabile*: il destinatario (Alice) può verificare che Bob, e nessun altro (nemmeno Alice), ha firmato il documento
- **firma digitale semplice per messaggio  $m$** :
  - Bob cifra  $m$  con la sua chiave privata  $K_B$ , creando un messaggio "firmato" con  $K_B^-(m)$



# Firme digitali - verifica

- supponiamo che Alice riceva msg  $m$ , con firma:  $m, K_B^-(m)$
- Alice verifica che  $m$  sia firmato da Bob applicando la chiave pubblica di Bob  $K_B^+$  a  $K_B^-(m)$  quindi controlla  $K_B^+(K_B^-(m)) = m$ .
- Se  $K_B^+(K_B^-(m)) = m$ , chiunque abbia firmato  $m$  deve aver usato la chiave privata di Bob

## Alice verifica così che:

- Bob ha firmato  $m$
- nessun altro ha firmato  $m$
- il messaggio  $m$  è integro, non è stato modificato lungo il percorso

## non ripudio:

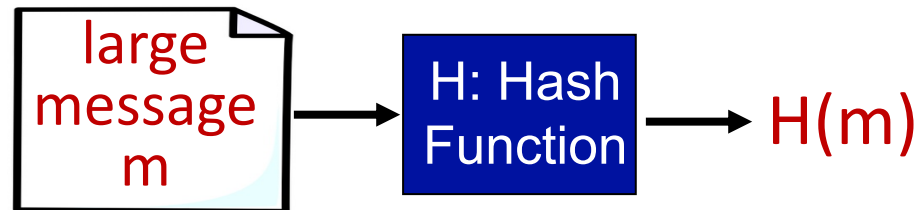
- ✓ Alice può portare  $m$ , e la firma  $K_B^-(m)$  in tribunale e dimostrare che Bob ha firmato  $m$  (valore legale, Bob non può negare di aver firmato)

# Message digests

computazionalmente costoso crittografare messaggi lunghi con chiave pubblica

**goal:** “impronta digitale” di lunghezza fissa e facile da calcolare

- applicando la funzione hash  $H$  a  $m$ , si ottiene un digest del messaggio di dimensioni fisse,  $H(m)$



## Proprietà della funzione hash:

- many-to-1 (messaggi diversi possono risultare nello stesso digest)
- produce digest msg di dimensioni fisse (impronta digitale)
- dato un digest  $x$ , è computazionalmente difficile ricostruire il messaggio  $m$  tale che  $x = H(m)$
- difficilissimo alterare il messaggio senza causare una modifica del digest

# Checksum Internet: una mediocre funzione hash crittografica

Il checksum Internet ha alcune proprietà delle funzioni hash:

- produce un digest di lunghezza fissa (somma a 16 bit) del messaggio
- è molti a uno

ma dato un messaggio con un determinato valore hash, è facile trovare un altro messaggio con lo stesso valore hash:

<u>message</u>	<u>ASCII format</u>		<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31		I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39		0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42		9 B O B	39 42 D2 42
<hr/>			<hr/>	
B2 C1 D2 AC		<i>messaggi diversi</i> <i>ma checksum identici!</i>	B2 C1 D2 AC	

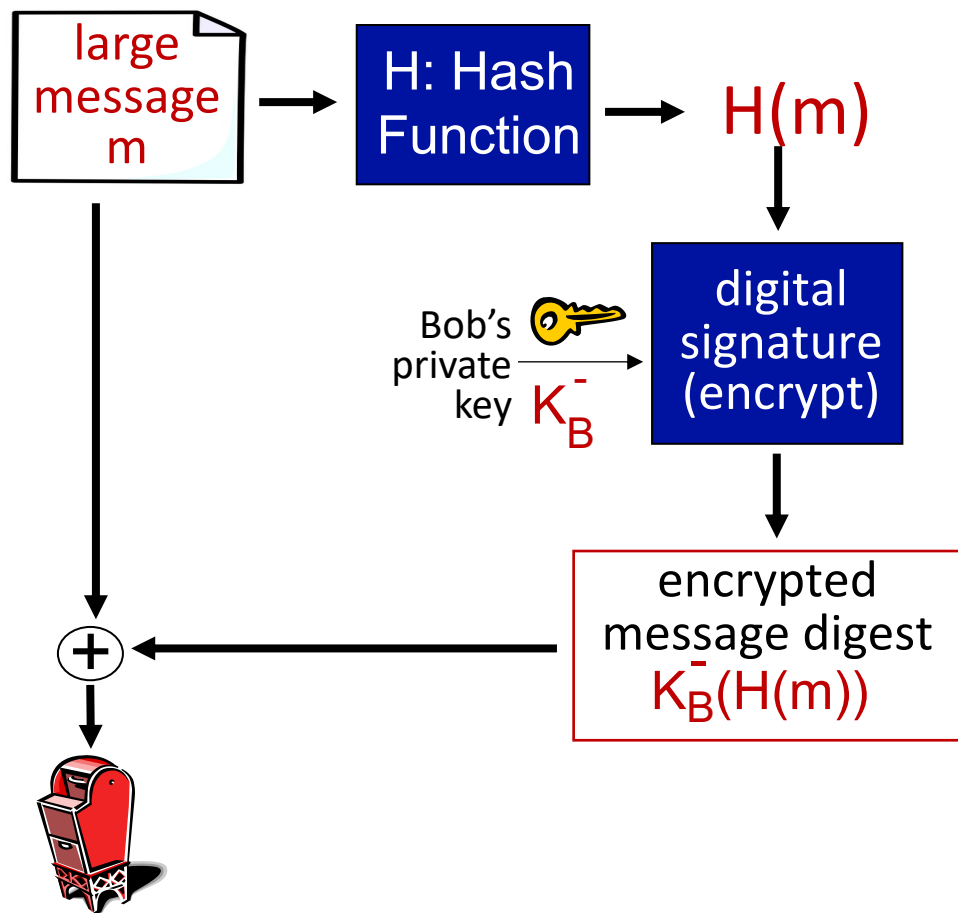


# Algoritmi per funzioni hash

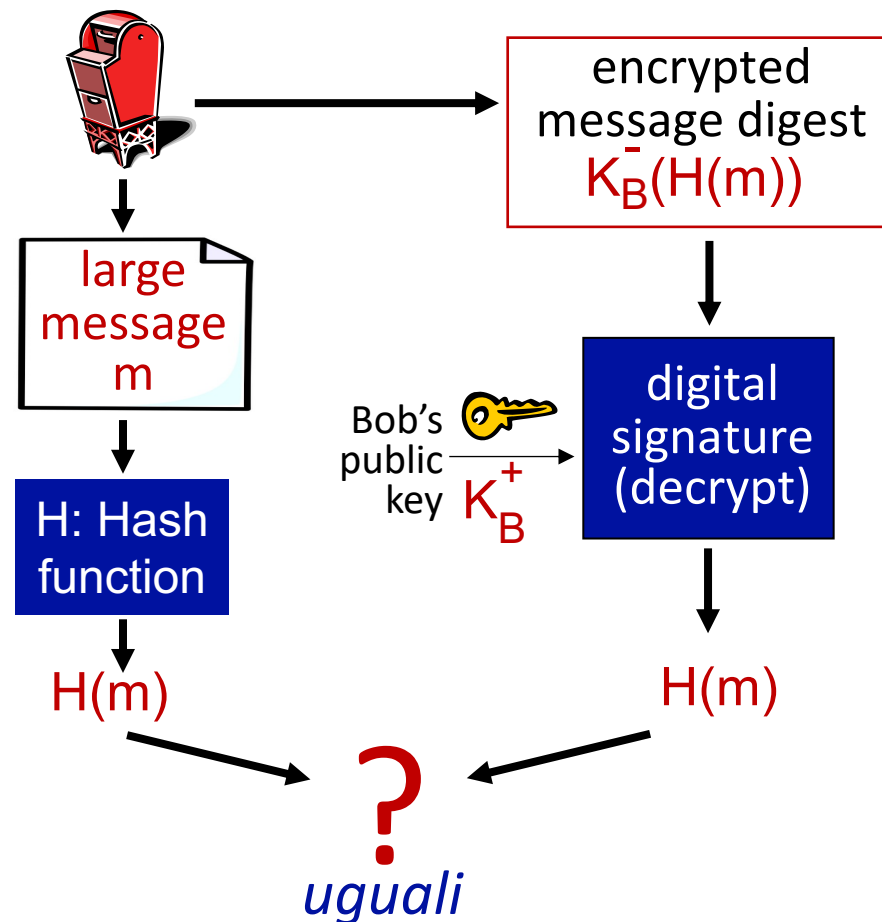
- **Funzione hash MD5 ampiamente utilizzata (RFC 1321)**
  - calcola il digest del messaggio a 128 bit con un processo in 4 fasi.
  - con una stringa  $x$  di 128 bit arbitraria, è difficile costruire un msg  $m$  il cui hash MD5 sia uguale a  $x$
- **Viene utilizzato anche SHA-1**
  - Standard USA [NIST, FIPS PUB 180-1]
  - Digest del messaggio a 160 bit

# Firma digitale = digest del messaggio firmato

Bob invia un messaggio con firma digitale

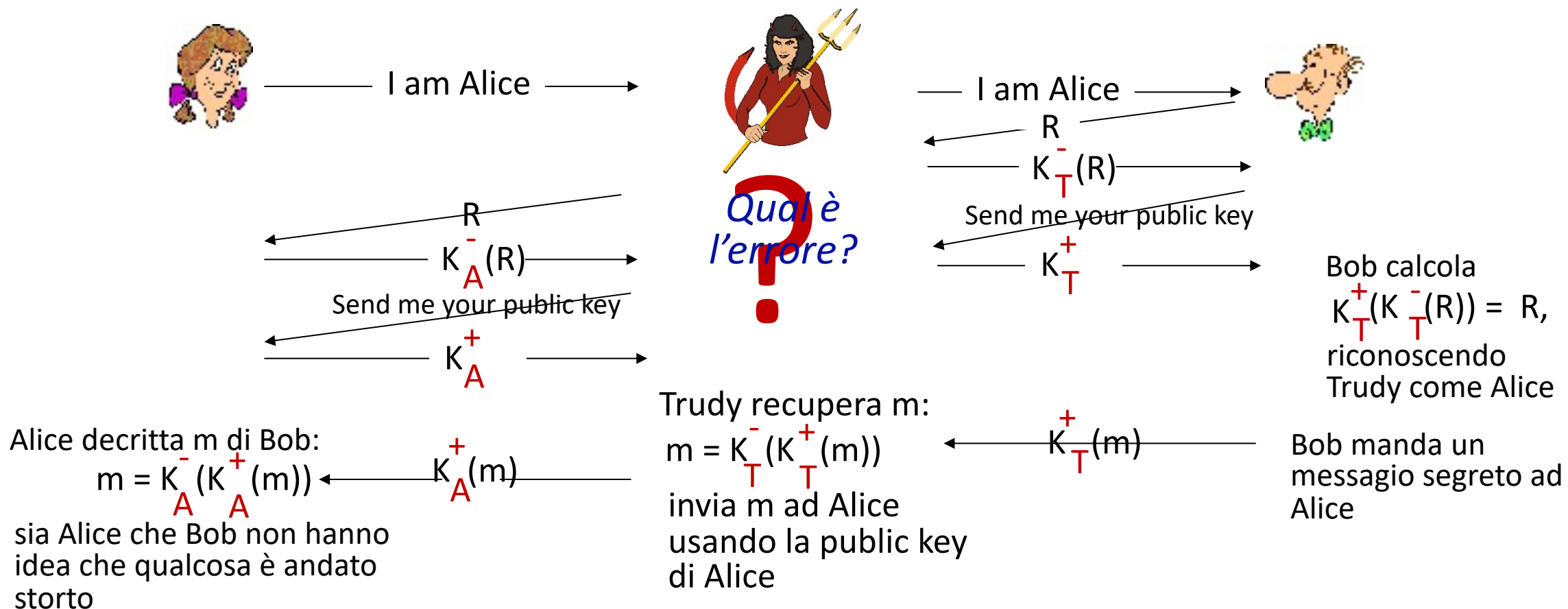


Alice verifica la firma e l'integrità del messaggio



# Autenticazione: ap5.0 – risolviamo il problema!

**Ricordiamo man-in-the-middle:** Trudy impersona Alice (con Bob) e Bob (con Alice)



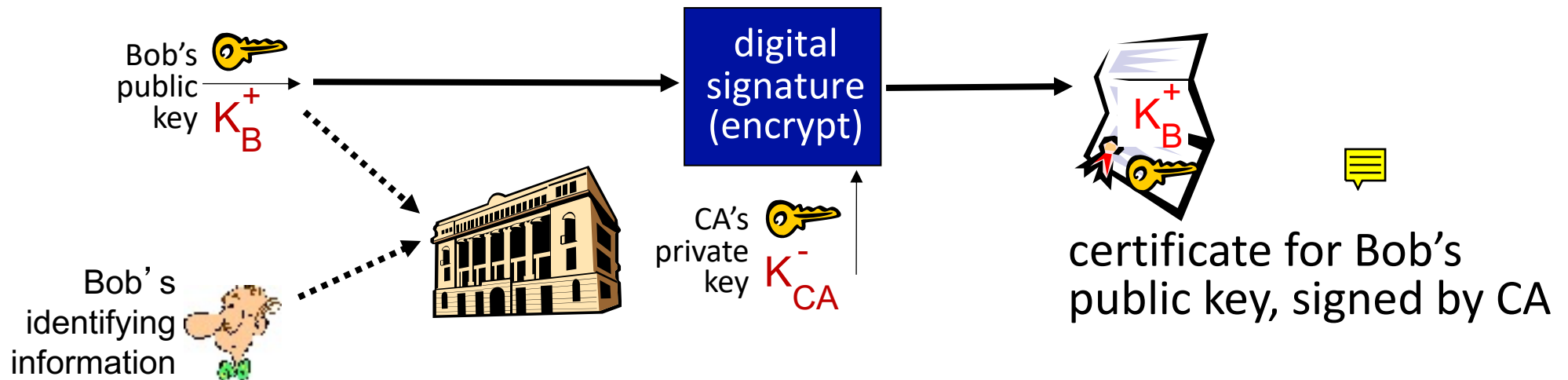
# Necessità di chiavi pubbliche certificate

- un altro esempio: Trudy fa uno scherzo con la pizza a Bob
  - Trudy crea un ordine via e-mail:  
*Caro Pizza Store, per favore consegnami quattro pizze. Grazie, Bob*
  - Trudy firma l'ordine con la sua chiave privata
  - Trudy invia l'ordine a Pizza Store
  - Trudy invia a Pizza Store la sua chiave pubblica, **ma dice che è la chiave pubblica di Bob**
  - Pizza Store verifica la firma; poi consegna quattro pizze a Bob
  - A Bob nemmeno piace la pizza!



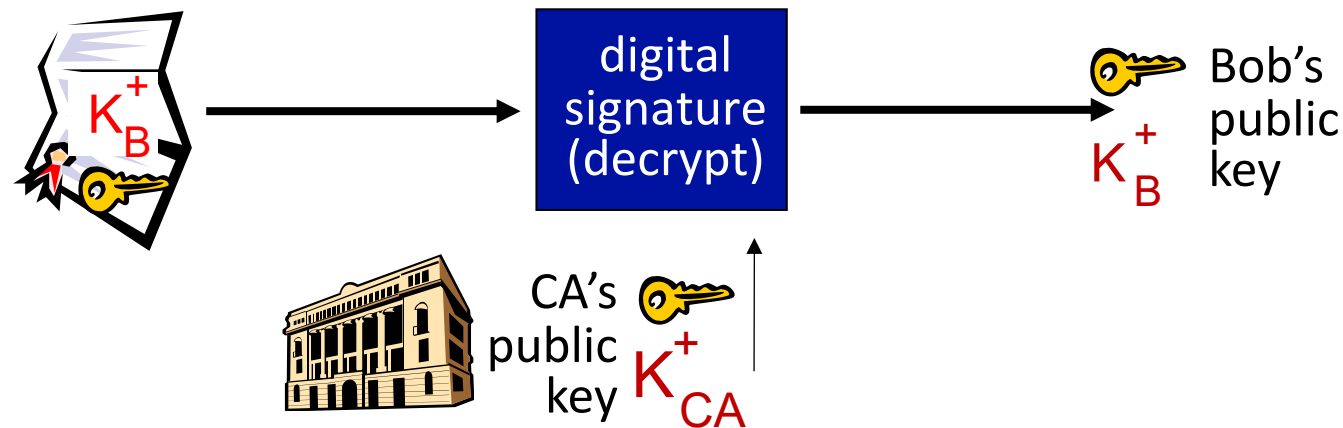
# Public key Certification Authorities (CA)

- **autorità di certificazione (CA):** associa la chiave pubblica a una particolare entità, E
- l'entità (persona, sito web, router) registra la propria chiave pubblica e inoltre fornisce "prova di identità" alla CA
  - La CA crea il certificato che associa la chiave pubblica di E alla sua identità
  - il certificato contenente la chiave pubblica di E firmata digitalmente da CA: CA dice "questa è la chiave pubblica di E"



# Public key Certification Authorities (CA)

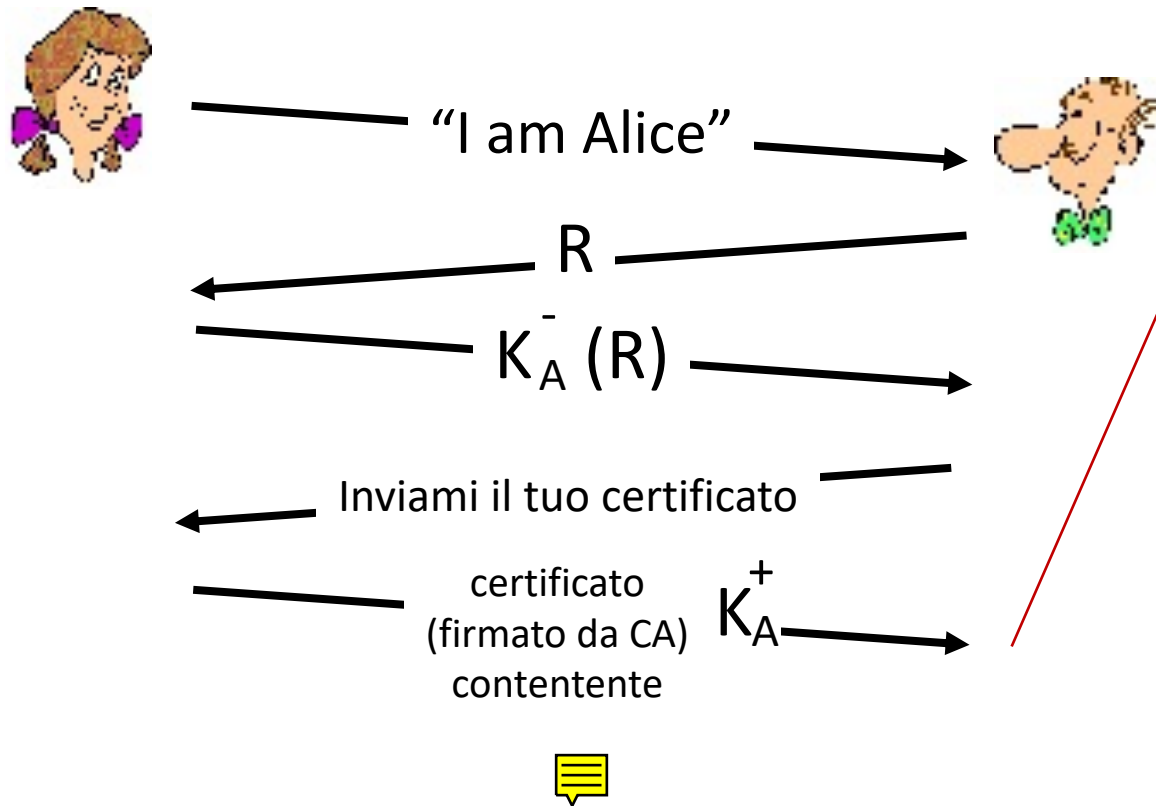
- quando Alice vuole la chiave pubblica di Bob:
  - ottiene il certificato di Bob (da Bob o altrove)
  - applica la chiave pubblica della CA alla firma del certificato di Bob per verificare la chiave pubblica di Bob
- Attacco man-in-the-middle non è più possibile
  - a meno che la CA non sia compromessa...



# Struttura del certificato

- Certificato
  - Versione
  - Numero seriale
  - ID dell'algoritmo
  - Ente emettitore
  - Validità
    - Non prima
    - Non dopo
  - Soggetto
  - **Informazioni sulla chiave pubblica del soggetto**
    - Algoritmo per l'utilizzo della chiave pubblica
    - **Chiave pubblica**
  - Codice identificativo univoco dell'emittente (facoltativo)
  - Codice identificativo univoco del soggetto (facoltativo)
  - Estensioni (facoltativo)
    - ...
- Algoritmo di firma del certificato
- **Firma del certificato**

# Autenticazione: completa



Bob usa la  $K_{CA}^+$  sul certificato fornito da Alice per validare  $K_A^+$

Bob calcola

$$K_A^+ (K_A^-(R)) = R$$

e sa che solo chi conosce la chiave privata di Alice può creare un messaggio tale che:

$$K_A^+ (K_A^-(R)) = R$$