Come generare efficientemente primi casuali

Algoritmo 8.31

Input : lunghezza n, parametro t

Output : un primo di n bit uniforme

for $i = 1$ to t

$p' \leftarrow \{0,1\}^{n-1}$

$p \leftarrow 1 \| p'$

se p è primo, restituisci p (return p)

return fallito

" sceglie ripetutamente interi casuali di n bit... "

Dato un modo per determinare se p è primo o no, l'algoritmo dà in output un primo uniforme di n bit (se non fallisce)

↓

la probabilità dipende da t

↓

Vogliamo scegliere t in modo tale da sia trascurabile

Pertanto, dobbiamo capire due cose:

① la probabilità che un intero uniforme di n bit è primo

② come verificare efficientemente se un intero è primo

① Il teorema dei numeri primi $\longrightarrow$ offre limitazioni precise alla frazione di interi di una data lunghezza che risultano primi.

Una versione più debole è sufficiente per i nostri scopi.

Teorema 8.32 (Postulato di Bertrand)

Per ogni $n > 1$, la frazione di interi di n bit che sono primi è almeno $1/3n$. ∎

Come scegliere t?

Se $t = 3n^2$ $\longrightarrow$ la probabilità che NON scegliamo un primo in tutte e t le iterazioni è:

$$\left(1 - \frac{1}{3n}\right)^t = \left(\left(1 - \frac{1}{3n}\right)^{3n}\right)^n \leq \left(e^{-1}\right)^n = e^{-n}$$

$\Longleftrightarrow$ usando un numero polinomiale di iterazioni, la probabilità diventa trascurabile.

② Verifica della primalità

Sin dagli anni '70 algoritmi probabilistici efficienti sono noti

Nel 2002 un algoritmo deterministico di tempo polinomiale è stato trovato (non usato nella pratica, meno efficiente dei probabilistici)

Garanzie degli algoritmi probabilistici

input p primo, output "primo"

input p composto, output
- quasi sempre "composto"
- con prob. trascurabile, "primo".

(∃ anche algoritmi con garanzie invertite)

Test di Miller e Rabin

Teorema 8.33. Se p è primo, allora il test di Miller e Rabin dà in output sempre "primo". Se p è composto, l'algoritmo dà in output "composto" eccetto con probabilità al più 2^{-t}.

⇒ Mettendo tutto insieme, possiamo raffinare l'algoritmo precedente per includere la verifica della primalità.

Algoritmo 8.34

Input : lunghezza n

Output : un primo uniforme di n bit

for $i = 1$ to $3 \cdot n^2$

$p' \leftarrow \{0,1\}^{n-1}$

$p \leftarrow 1 \| p'$

Esegui il test di Miller-Rabin su p e 1^n

se l'output è "primo", return p

return "fallito"

Osservazioni:

- Facile da modificare per ottenere primi di forma specifica. Due forme importanti sono:

$$p = 2q + 1 \qquad \text{(strong primes)}$$

(p primo, q primo)

p tale che $p = 3 \bmod 4$

- Funzionano bene in pratica. Difficile fornire prove rigorose che queste versioni modificate eseguono in tempo polinomiale e falliscono con prob. trascurabile.

Test di Miller e Rabin

Idea chiave: trovare una proprietà che distingue i primi dai composti.

Se N è primo, allora $|\mathbb{Z}_N^*| = N-1$

$\Rightarrow$ per ogni $a \in \{1, \dots, N-1\}$, $a^{N-1} = 1 \bmod N$

Test: scegli un elemento a uniforme e verifica se $a^{N-1} = 1 \bmod N$

NO $\swarrow$ $\searrow$ SI

N non può essere primo — Potrebbe esserlo...

Ripeti il test molte volte...

Algoritmo 8.35

Input: N e 1^t

Output: Decisione su N (primo o no)

for $i := 1$ to t

$a \leftarrow \{1, \dots, N-1\}$

if $a^{N-1} \neq 1 \bmod N$ return "composto"

return "primo".

Ogni a tale che $a^{N-1} \neq 1 \bmod N$ è un "testimone" che N è composto.

Intuizione

Noi speriamo che quando N è composto ci sono molti testimoni

$\Downarrow$

l'algoritmo trova un testimone con alta probabilità.

L'intuizione è corretta posto che ci sia almeno un testimone.

Nota: se $a \notin \mathbb{Z}_N^*$, allora $\gcd(a, N) \neq 1$.

Discende che $\gcd(a^{N-1}, N) \neq 1$ e, quindi,

$a^{N-1} \neq 1 \bmod N$.

Nell'analisi seguente restringiamo l'attenzione al caso in cui $a \in \mathbb{Z}_N^*$.

Teorema 8.38. Fissato N, diciamo che esiste un testimone che N è composto. Allora, almeno la metà degli elementi di $\mathbb{Z}_N^*$ sono testimoni che N è composto.

Dal teorema discende che:

se N è composto ed esiste un testimone che N è composto

$\Downarrow$

ci sono almeno $|\mathbb{Z}_N^*|/2$ testimoni

$\Downarrow$

la probabilità che in una iterazione troviamo un testimone o che $a \notin \mathbb{Z}_N^*$ risulta $\geq 1/2$

$\Downarrow$

La probabilità che l'algoritmo non trova un testimone in nessuna delle t iterazioni è $\leq \frac{1}{2^t}$.

Sfortunatamente non è una soluzione completa

$\Downarrow$

$\exists$ infinitamente tanti numeri composti N che non hanno alcun testimone della loro compositezza.

Tali N sono noti come numeri di Carmichael.

Cosa fare allora?

Un raffinamento del test funziona per tutti gli N.

Sia $N-1 = 2^s \cdot u$, u dispari ed $s \geq 1$

(u: possono essere calcolati facilmente)

L'algoritmo precedente controlla solo $a^{N-1} = a^{2^s \cdot u} = 1 \bmod N$?

Il test più raffinato controlla la sequenza di $s+1$ valori

$$a^u, a^{2u}, \ldots, a^{2^s \cdot u} \quad (\text{modulo } N)$$

↑ ciascun valore è il quadrato del valore precedente

Nota che se qualche valore è uguale a ± 1, allora tutti i valori seguenti nella sequenza sono uguali a 1.

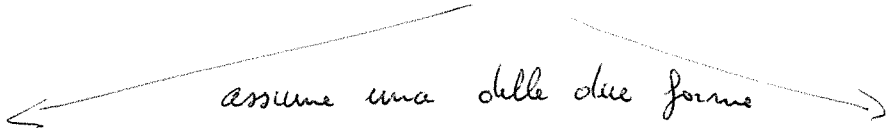
Diremo che $a \in \mathbb{Z}_N^*$ è un <u>testimone forte</u> che N è composto se:

1) $a^u \neq \pm 1 \bmod N$

2) $a^{2^i \cdot u} \neq -1 \bmod N$, per tutti gli $i \in \{1, \ldots, s-1\}$

Osservazione: quando un elemento a NON è un testimone forte vale la seguente proprietà.

la sequenza $a^u, a^{2u}, a^{2^2 u}, \dots, a^{2^2 \cdot u}$



$(\pm 1, 1, \dots, 1)$ $(*, *, \dots *, -1, 1, \dots, 1)$

1) 2)

In entrambi i casi, $a^{2^{2-1} \cdot u} = \pm 1 \bmod N$ e, pertanto

$$a^{N-1} = a^{2^2 \cdot u} = \left(a^{2^{2-1} \cdot u}\right)^2 = 1 \bmod N$$

$\Rightarrow$ a non è un testimone di N è composto.

Quindi, se a non è un testimone forte, a non è un testimone.

Equivalentemente: a testimone $\Rightarrow$ a è anche un testimone forte.

Pertanto, ci sono più testimoni forti che testimoni.

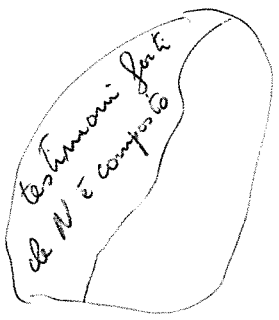
Si può dimostrare che:

> Se N è primo $\Rightarrow$ non esiste un testimone forte che N è composto

Prova data nel lemma 8.39.

Osservazione. Un intero composto N è <u>una potenza di un primo</u> se $N = p^2$, p primo, $2 \geq 1$ intero.

Teorema 8.50. Sia N un numero dispari che non è una potenza di un primo. Allora almeno la metà degli elementi di $\mathbb{Z}_N^*$ sono testimoni forti che N è composto.



N non è una potenza di un primo

"... più della metà di $\mathbb{Z}_N^*$"

Mettendo tutto insieme:

1) Se N è un primo dispari, NON esistono testimoni forti che N è composto

   $\Rightarrow$ il test di Miller e Rabin restituisce "primo"

2) Se N è pari o è la <u>potenza di un primo</u>, si può dimostrare che esiste un modo efficiente per

accorgersene (vedere se $N = \hat{N}^e$, ovvero è una potenza perfetta)

$\Rightarrow$ il test di Miller e Rabin restituisce "composto".

3) Se N è un intero dispari composto, allora

se $a \notin \mathbb{Z}_N^*$ si può far vedere che

$$a^u \neq \pm 1 \bmod N \quad \text{e} \quad a^{2^i \cdot u} \neq -1 \bmod N, \quad i = 1, \dots, e-1$$

se $a \in \mathbb{Z}_N^*$, per il Teorema 8.40, a è un testimone forte con probabilità $\geq 1/2$

Pertanto, iterando il test t volte, la probabilità che in nessuna delle t iterazioni si abbia in output "composto" è al più 2^{-t}.

$\Rightarrow$ abbiamo dimostrato il teorema 8.33.

L'algoritmo finale è riportato di seguito.

Algoritmo 8.44

Input: Intero $N > 2$ e parametro 1^t

Output: Decisione: "primo" o "composto"

if N è pari, return "composto"

if N è una potenza perfetta, return "composto"

calcola $r \geq 1$ ed u dispari tali che

$$N-1 = 2^r \cdot u$$

for $j = 1$ to t

$a \leftarrow \{1, \dots, N-1\}$

if $a^u \neq \pm 1 \bmod N$ e $a^{2^i u} \neq -1 \bmod N$, $i = 1, \dots, r-1$

return "composto"

return "primo".

Distribuzione e gestione delle chiavi

"Come possono le parti condividere una chiave segreta prima di iniziare a comunicare?"

Abbiamo visto che la crittografia a chiave privata può essere usata per realizzare comunicazioni sicure su un canale insicuro

Pertanto, sembra che risolva completamente il problema principale della crittografia.

Ma non è così!

La condizione iniziale - condivisione della chiave - deve essere soddisfatta.

Può essere effettuata attraverso un "canale sicuro": un servizio di corrieri fidati.

Governi, corpi militari e l'Intelligence possono. Persone comuni forse no.

Le parti possono incontrarsi fisicamente in un luogo, generare la chiave e portarne una copia con sé.

Soluzione parziale in un ambiente di lavoro

↓

Uso di un "controllore" fidato

Compagnia: ogni dipendente dispone di
- una chiave (per comunicare in modo sicuro con il controllore) K_c^u
- una chiave (per comunicare con il dipendente i-esimo) K_i^u

per $i = 1, \dots, \ell - 1$

Nuovo dipendente: riceve
($(\ell+1)$esimo)
- una chiave (per comunicare in modo sicuro con il controllore) K_c^v
- una chiave (per comunicare con il dipendente i-esimo) K_i^v

per $i = 1, \dots, \ell$.

Dipendenti già presenti : il dipendente j-esimo, $j = 1, \dots, l$ riceve

- K_j^* cifrata con la chiave K_c^j che condivide con il controllore

Tuttavia :

- è un approccio complicato
- le chiavi non sono completamente segrete
- un controller disonesto può decifrare tutte le comunicazioni tra i dipendenti

$\longrightarrow$

- l dipendenti $\Rightarrow \Theta(l^2)$ chiavi segrete nel sistema
- ogni dipendente deve memorizzare l chiavi
- possono aver bisogno di altre chiavi per accedere a risorse remote

Quando l'azienda è grande, un tale approccio crea diversi problemi, a vari livelli.

Quando un numero ridotto di chiavi deve essere memorizzato la soluzione è percorribile.

Le chiavi generate da una parte fidata vengono memorizzate in una smart card, un dispositivo hardware protetto in modo robusto.

Quindi, in sistemi "chiusi", le chiavi possono essere distribuite "fisicamente".

Sfortunatamente, in sistemi "aperti" in cui le parti non hanno modo di ottenere fisicamente le chiavi, la crittografia a chiave privata da sola è insufficiente a garantire la sicurezza di operazioni tipo:

- acquisto sicuro su Internet
- invio di una email confidenziale ad un collega in un'altra nazione.

Soluzione parziale: Key Distribution Center (KDC)

Tutti i dipendenti di un'azienda devono fidarsi per esempio del manager ICT. Il manager installa un singolo server, il KDC, che svolge la funzione di intermediario tra i dipendenti che desiderano comunicare.

Un KDC lavora come segue:

- ogni dipendente condivide una chiave con il KDC, generata e consegnata al dipendente il primo giorno di lavoro
- quando Alice vuole comunicare con Bob in modo sicuro, invia una richiesta al KDC, autenticata con la chiave condivisa
- il KDC sceglie a caso una chiave segreta, detta chiave di sessione, e

- la invia ad Alice, cifrata con la chiave segreta condivisa con Alice
- la invia a Bob, cifrata con la chiave condivisa con Bob

Alice e Bob, ottenuta la chiave di sessione, possono comunicare in modo sicuro

Al termine della sessione, Alice e Bob cancellano la chiave.

Vantaggi:

- Ogni dipendente memorizza soltanto una chiave segreta
  Il KDC ne memorizza molte ma può essere installato in un luogo sicuro
- Un nuovo dipendente comporta la generazione di una nuova chiave soltanto.
  Nessun aggiornamento per gli altri è richiesto.
- Similmente in caso di licenziamento.

Svantaggi

- Un attacco riuscito contro il KDC compromette la sicurezza dell'intero sistema e delle sue parti.
- Il KDC è un target per attacchi
- Un Adv interno, il manager ICT, con accesso al KDC, può decifrare tutte le comunicazioni
- Il KDC rappresenta un singolo punto critico: se il server non funziona, le comunicazioni sicure sono temporaneamente impossibili
- Il KDC può essere sovraccaricato, rallentando le comunicazioni e incrementando la probabilità di fallimenti.

Soluzioni possibili:

- replica del KDC nel sistema
  - funziona ma introduce più punti d'attacco
  - nascono problemi di coordinamento e sincronizzazione quando utenti debbono essere aggiunti/rimossi

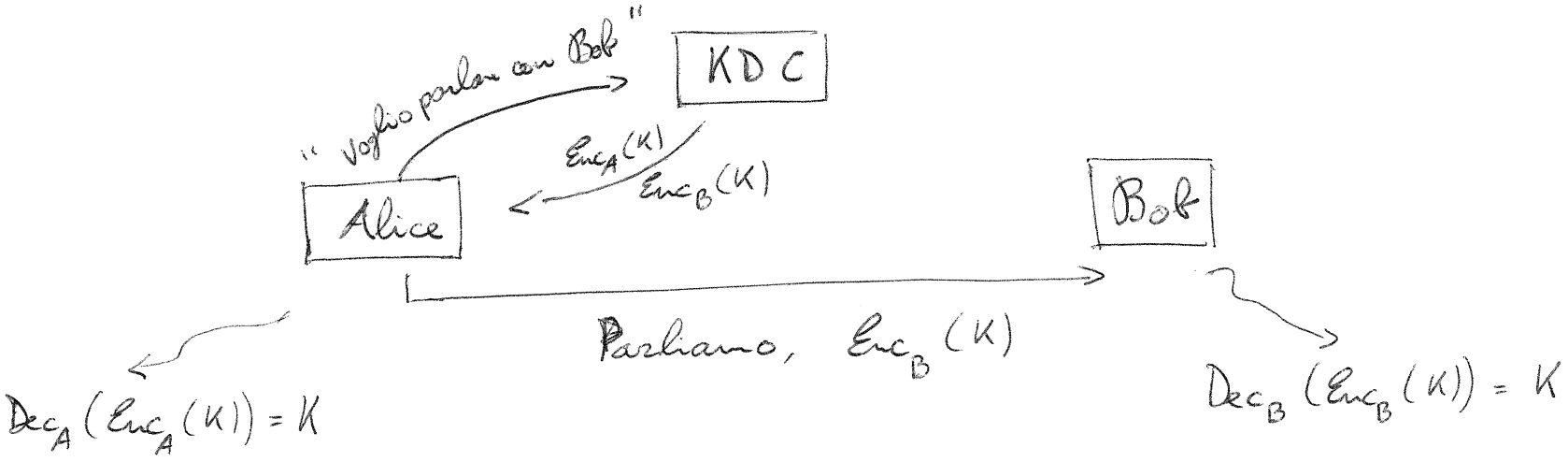
- realizzazione di un KDC "distribuito", detto DKDC
  - nessun server da solo conosce le chiavi
  - si riduce la probabilità di disservizi in caso di rottura
  - il carico delle richieste è distribuito tra i server
  - un utente deve contattare un sottoinsieme dei server per ottenere una chiave di sessione

Esistono diversi protocolli per la distribuzione di chiavi usando un KDC

Protocollo di Needham - Schroeder

Rispetto al modello descritto, tale protocollo si discosta come segue:

- Quando Alice contatta il KDC e chiede di comunicare con Bob, il KDC non invia la chiave di sessione cifrata ad entrambi.
- Il KDC invia la chiave di sessione cifrata DUE volte, con la chiave condivisa con Alice e con quella condivisa con Bob (due messaggi diversi) ed è Alice stessa ad inviare a Bob la chiave di sessione cifrata con la chiave che Bob condivide con KDC



Il protocollo fu progettato in questo modo perché Bob potrebbe essere off-line.
Il KDC in tal caso sarebbe costretto ad attendere, mantenendo "aperta la sessione".

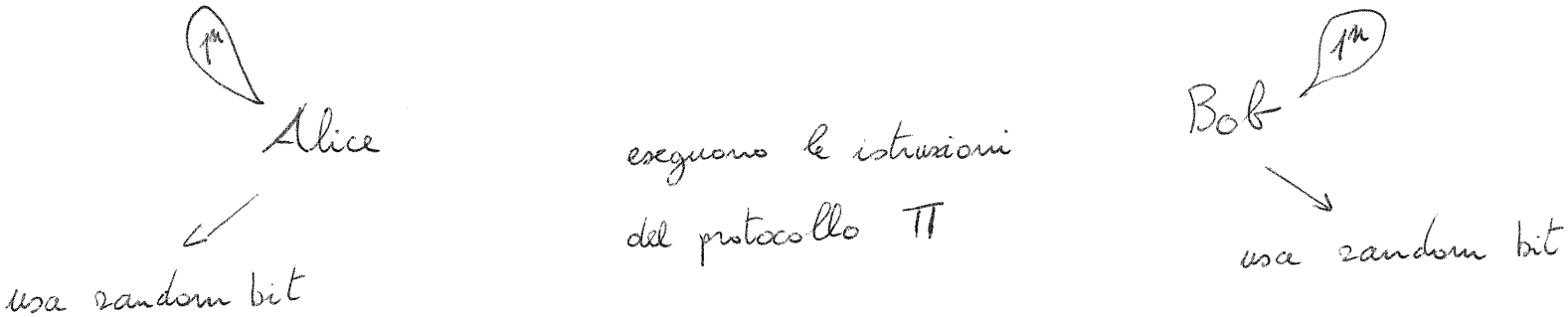
La chiave di sessione per Bob ricevuta da Alice viene detta "ticket".

È una sorta di credenziale che permette ad Alice di comunicare con Bob

Un sistema reale che utilizza un KDC è KERBEROS (sviluppato al MIT)

Usato da Windows per proteggere una rete aziendale.

Scambio di chiavi Diffie-Hellman



Al termine dell'esecuzione di Π, Alice e Bob calcolano, rispettivamente, K_A e $K_B \in \{0,1\}^m$

Requisito di base di Π (correttezza): $K_A = K_B = K$

Intuitivamente, un protocollo di scambio di chiavi è sicuro se la chiave che Alice e Bob danno in output è totalmente impredicibile da un Adv che ascolta la comunicazione.

Può essere formalizzata richiedendo che Adv, dopo aver ascoltato una esecuzione, non è in grado di distinguere la chiave K, generata da Π, da una chiave scelta uniformemente a caso di lunghezza n.

Si noti che è una nozione più forte della richiesta di "imprevedibilità", i.e., calcolare K esattamente

$KE^{eav}_{A,\Pi}(n)$: /* esperimento */ Π protocollo, A Adv, n par. sicurezza

1. Due parti, su input 1^n, eseguono Π.
   Sia trans il transcript della comunicazione totale e sia K la chiave che danno in output
2. Viene scelto $b \leftarrow \{0,1\}$. Se $b = 0$, pone $\hat{K} := K$; altrimenti $\hat{K} \leftarrow \{0,1\}^n$ (uniforme)
3. A riceve trans e $\hat{K}$, e dà in output b'
4. L'output dell'esperimento è 1 se $b' = b$; 0, altrimenti

↓

A vince

Definizione 8.1 Un protocollo di scambio di chiavi Π è sicuro in presenza di un ascoltatore se, $\forall A$ PPT, $\exists$ una funzione trascurabile negl, tale che:

$$P_2\left[KE^{eav}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

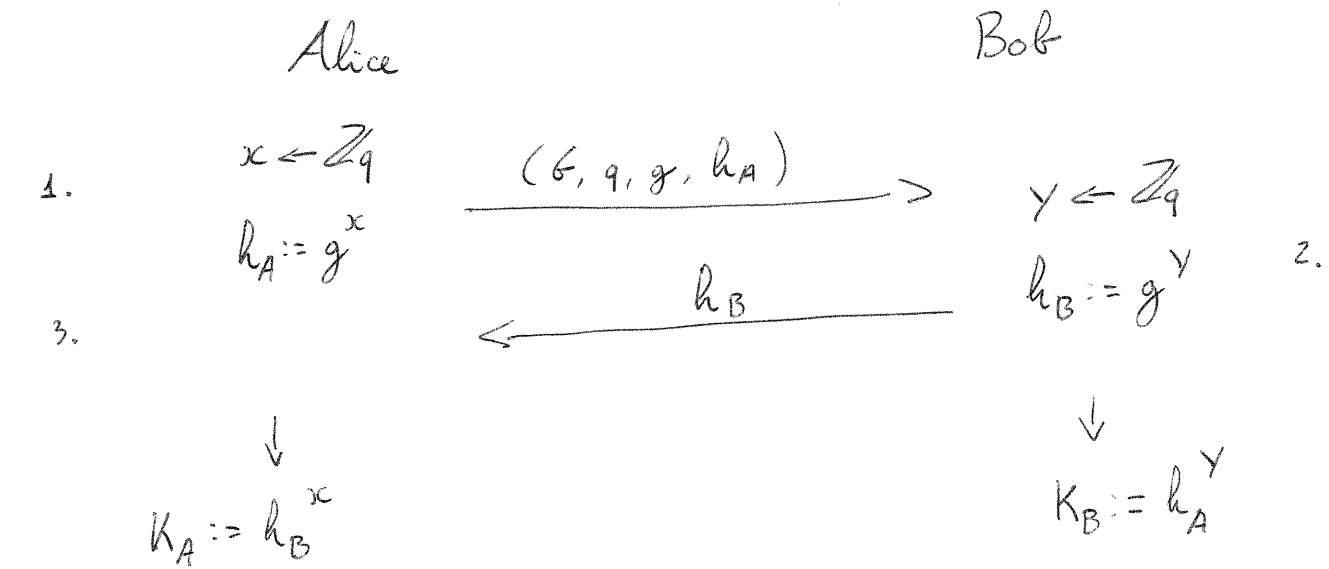
Protocollo di Diffie - Hellman

$G(1^n)$, PPT, dà in output un gruppo ciclico G, di ordine q (n bit) e un generatore $g \in G$

1. Alice esegue $G(1^n)$ ed ottiene (G, q, g).
   Sceglie $x \leftarrow \mathbb{Z}_q$ e calcola $h_A := g^x$
   Invia (G, q, g, h_A) a Bob

2. Bob riceve (G, q, g, h_A).
   Calcola $y \leftarrow \mathbb{Z}_q$ e $h_B := g^y$. Invia h_B ad Alice e dà in output $K_B := h_A^y$

3. Alice riceve h_B e dà in output $K_A := h_B^x$



È facile vedere che il protocollo è corretto.

$$K_B = h_A^y = (g^x)^y = g^{xy}$$
$$K_A = h_B^x = (g^y)^x = g^{yx}$$

→ elemento del gruppo G (uniforme)

(elemento di G)

↓

ha "chiave" $K = g^{xy}$ deve essere trasformata usando una appropriata funzione di derivazione H

↓

ma stringa di bit uniforme

Circa la sicurezza, possiamo convenire che:

- DL deve essere difficile relativamente a G
- DH computazionale deve essere difficile relativamente a G, ma non è sufficiente per provare che il protocollo raggiunge il requisito della definizione.

Vale il seguente

Teorema 10.3 Se il problema DDH è difficile relativamente a G, allora lo scambio di chiavi Diffie-Hellman Π è sicuro in presenza di un Adv di ascolta.

Nota: Sia $\widehat{KE}^{eav}_{A,\Pi}(n)$ l'esperimento $KE^{eav}_{A,\Pi}(n)$ in cui A distingue tra K e un elemento di G scelto uniformemente a caso.

Dim. Sia A un Adv PPT.

Poiché $P_2[b=0] = P_2[b=1] = 1/2$, risulta

$$P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n)=1\right] = \frac{1}{2}\cdot P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n)=1 \mid b=0\right] + \frac{1}{2}\cdot P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n)=1 \mid b=1\right]$$

Nell'esperimento $\widehat{KE}^{eav}_{A,\Pi}$, A riceve (G, q, g, h_A, h_B) e $\hat{K}$ (chiave K / elemento u uniforme di G)

Nota che distinguere tra i due casi (h_A, h_B, K) e (h_A, h_B, u)

è equivalente a risolvere DDH!

Pertanto la $P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n)=1\right]$ risulta uguale a

$$\frac{1}{2}\cdot P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n)=1 \mid b=0\right] + \frac{1}{2}\cdot P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n)=1 \mid b=1\right]$$

$$= \frac{1}{2}\cdot P_2\left[A(G, q, g, g^x, g^y, g^{xy})=0\right] + \frac{1}{2}\cdot P_2\left[A(G, q, g, g^x, g^y, g^z)=1\right]$$

($z \leftarrow \mathbb{Z}_q$)

$$= \frac{1}{2} \cdot \left(1 - P_2\left[A(G, q, g, g^x, g^y, g^{xy}) = 1\right]\right) + \frac{1}{2} \cdot P_2\left[A(G, q, g, g^x, g^y, g^z) = 1\right]$$

$$= \frac{1}{2} + \frac{1}{2} \cdot \left(P_2\left[A(G, q, g, g^x, g^y, g^z) = 1\right] - P_2\left[A(G, q, g, g^x, g^y, g^{xy}) = 1\right]\right)$$

$$\leq \frac{1}{2} + \frac{1}{2} \cdot \left| P_2\left[A(G, q, g, g^x, g^y, g^z) = 1\right] - P_2\left[A(G, q, g, g^x, g^y, g^{xy}) = 1\right] \right|$$

Se DDH è difficile relativamente a $\mathcal{G}$, allora $\exists$ negl tale che:

$$\left| P_2\left[A(G, q, g, g^x, g^y, g^z) = 1\right] - P_2\left[A(G, q, g, g^x, g^y, g^{xy}) = 1\right] \right| \leq \text{negl}(n)$$

Discende, quindi, che:

$$P_2\left[\widehat{KE}^{eav}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + \frac{1}{2} \cdot \text{negl}(n)$$

■

Cosa possiamo dire rispetto ad avversari "attivi"?

Adv attivo: può inviare messaggi ad una o entrambe le parti

- attacchi di impersonificazione: una parte onesta esegue Π e l'Adv si sostituisce all'altra parte

- attacchi di tipo "Man-in-the-middle": le due parti eseguono Π e l'Adv intercetta e modifica i messaggi nelle due direzioni

Diffie-Hellman è totalmente insicuro rispetto ad attacchi "Man-in-the-middle". Un Adv può far sì che, al termine di Π, Alice e Bob condividano chiavi con lui, credendo di condividere una chiave tra loro.

La rivoluzione a chiave pubblica

Nel 1976, Whitfield Diffie e Martin Hellman pubblicarono un articolo dal titolo apparentemente innocente:

"New directions in Cryptography"

L'influenza fu enorme.

- Introdusse un nuovo modo di guardare alla crittografia
- Servì per portare la crittografia fuori dal dominio privato e all'interno del pubblico

Possiamo immaginare un crittosistema con DUE chiavi invece di una

- una chiave di cifratura, usata dal mittente per cifrare i messaggi
- una chiave di decifratura, usata dal ricevente per decifrare i messaggi

la segretezza dei messaggi cifrati è preservata anche rispetto ad un Adv che conosce la chiave di cifratura (ma non quella di decifratura, ovviamente)

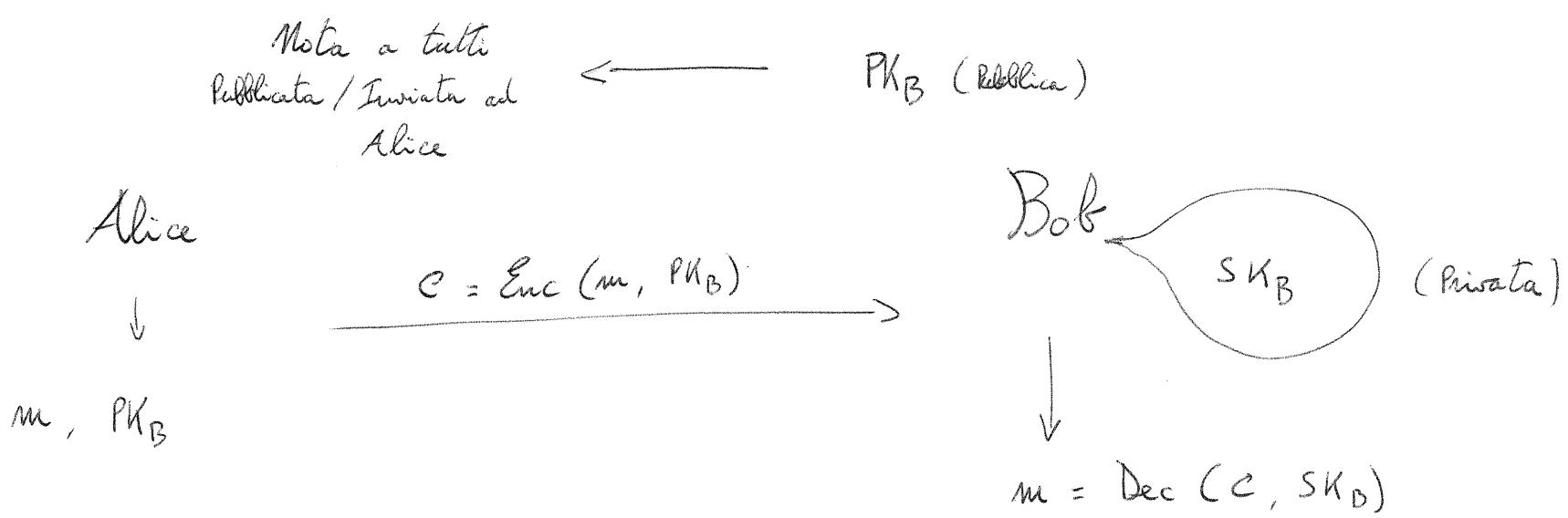
Sembra che uno schema del genere non possa esistere, vero?

Invece esistono e si usano tutti i giorni.

Sono detti "asimmetrici" o "a chiave pubblica", in contrapposizione agli schemi simmetrici o a chiave privata che abbiamo trattato sino ad ora.

La chiave di cifratura viene detta "chiave pubblica", perché viene pubblicata dal ricevente, in modo tale che chiunque possa inviargli messaggi.

La chiave di decifratura viene detta "chiave privata", poiché viene tenuta segreta con molta cura dal ricevente, in modo da essere l'unico a decifrare i messaggi.



Osservazioni:

- la crittografia a chiave pubblica rende possibile la distribuzione di chiavi, da usare con sistemi simmetrici, su canali pubblici
- semplifica lo sviluppo iniziale di un sistema e però complifica il mantenimento del sistema quando parti entrano o escono dal sistema

- riduce notevolmente la necessità di memorizzare molte chiavi segrete
  - anche se tutte le coppie di utenti di un sistema vogliono comunicare in modo sicuro, ciascuna parte deve memorizzare soltanto la propria chiave privata
  - le chiavi pubbliche degli altri possono essere ottenute o quando richiesto oppure memorizzate in una modalità non protetta, i.e., pubblicamente, accessibili a tutti.

- maggiormente idonea ad ambienti aperti, in cui le parti possono non aver mai interagito tra di loro in precedenza.