Crittografia a chiave pubblica

Alessandro Armando

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

Sicurezza del computer

Contorno

Motivazione

> La crittografia a chiave pubblica nasce nel maggio 1975, figlia di due problemi: il problema della distribuzione delle chiavi e il problema delle firme. La scoperta consisteva non in una soluzione, ma nel riconoscimento che i due problemi, ciascuno dei quali sembrava irrisolvibile per definizione, potevano essere risolti del tutto e che le soluzioni per entrambi arrivavano in un unico pacchetto.
>
> Whitfield Diffie, *I primi dieci anni della crittografia a chiave pubblica* , 1988

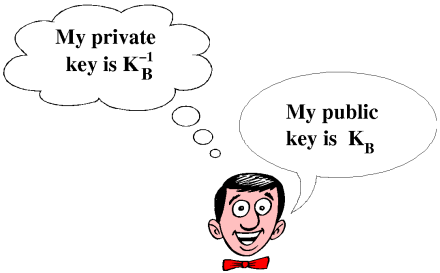
Consideriamo fino a che punto questi problemi sono “risolti”.

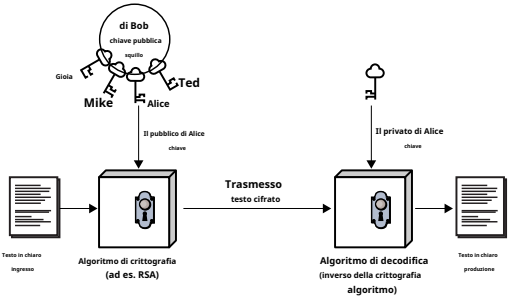
Crittografia a chiave pubblica

- Permettere $\{E_e : e\ K\}$ e $\{D_e : e\ K\}$ formare uno schema di crittografia.
- Considera le coppie di trasformazione (E_e, D_D) dove sapere? E_e è irrealizzabile, dato C Cper trovare un m M dove $E_e(m) = C$. Ciò implica che è impossibile determinare$D$ a partire dal$e$.

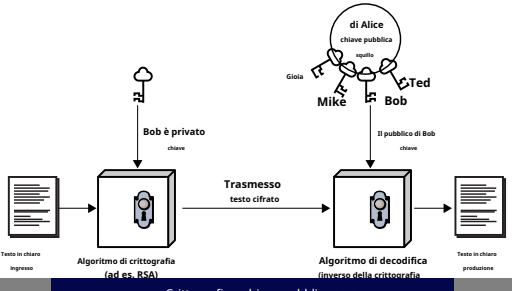
  ? E_e costituisce una botola con funzione unidirezionale con botola D.

- Chiave pubblica come e può essere un'informazione pubblica





(a) Crittografia



Crittografia convenzionale (simmetrica) e a chiave pubblica (asimmetrica)

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Crittografia a chiave pubblica: segretezza

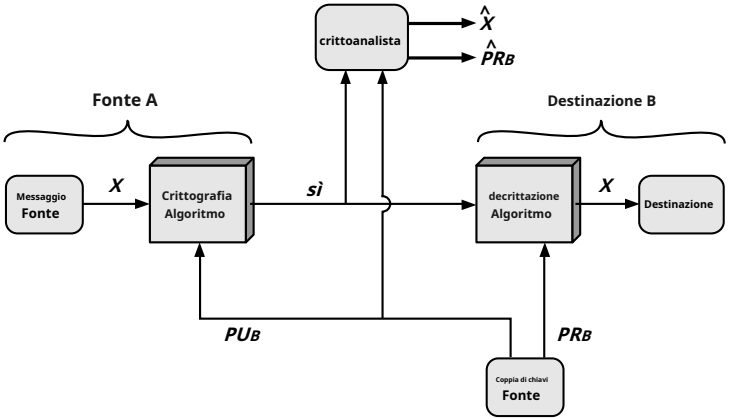


Figura 9.2 Pubblica [illegible] Segretezza

Crittografia a chiave pubblica: autenticazione

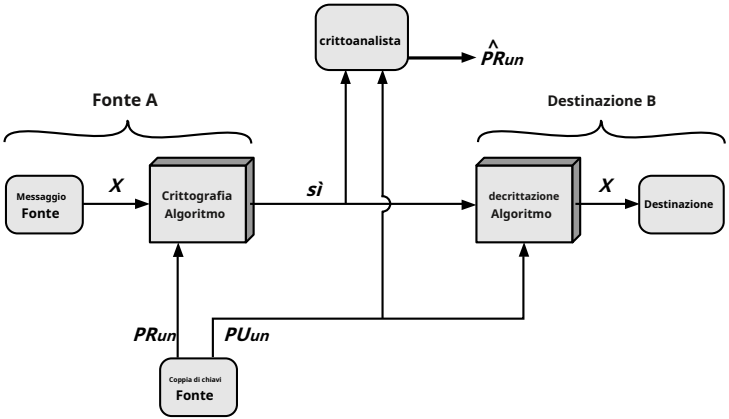


Figura 9.3 Pubblico P tu blu circuito integrato -K e si C R si P a G R un P h di autenticazione

Crittografia a chiave pubblica: segretezza e autenticazione

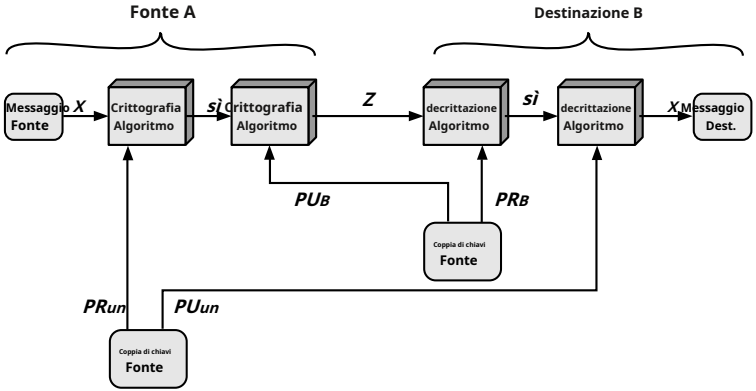


Figura 9.4 Crittografia a chiave pubblica: autenticazione e segretezza

Requisiti per la crittografia a chiave pubblica

1. È computazionalmente facile per B per generare una coppia (chiave pubblica PU_B, chiave privata PR_B).
2. È computazionalmente facile per il mittente UN, sapendo PU_B e m, generare

$$C = E(PU_B, m)$$

3. È computazionalmente facile per il ricevitore B decifrare C usando PR_B riprendersi m:

$$m = D(PR_B, C) = D(PR_B, E(PU_B, m))$$

4. È computazionalmente irrealizzabile per un avversario, sapendo PU_B determinare PR_B.
5. È computazionalmente irrealizzabile per un avversario, sapendo PU_B e $C = E(PU_B, m)$ riprendersi m.
6. (Utile, ma non sempre necessario) Le due chiavi possono essere applicate in qualsiasi ordine:

$$m = D(PU_B, E(PR_B, m)) = D(PR_B, E(PU_B, m))$$

- Questi sono requisiti difficili.
- Di fatto solo pochi algoritmi che godono dei requisiti di cui sopra hanno ricevuto finora un'ampia accettazione:

Algoritmo	Crittografia/ decrittazione	Firma digitale	Scambio di chiavi
RSA	sì	sì	sì
Curva ellittica	sì	sì	sì
Diffie Hellman	No	No	sì
DSS	No	sì	No

Funzioni unidirezionali

- Una funzione $F : X \to sì$ è un funzione unidirezionale, Se F è “facile” da calcolare per tutti $X ? X$, ma $F–1$ è "difficile" da calcolare
- **Esempio:** problema di radici cubiche modulari
  - Seleziona numeri primi $P = 48611$ e $Q = 53993$.
  - Permettere $n = pq = 2624653723$ e $X = \{1, 2, \ldots, n-1\}$.
  - Definire $F : X \to$ n di $F(X) = X3$ modalità n.
  - Esempio: $F(2489991) = 1981394214$. Informatica F è facile. Invertendo F
  - è difficile: dato $sì$ e n, trova X tale che $X3 = sì$ modalità n.
- **Nota:** Da non confondere con le funzioni unidirezionali che prendono un campo dati arbitrariamente come argomento e lo mappano su un output a lunghezza fissa.

Funzioni botola unidirezionali

- UN botola funzione unidirezionale è una funzione unidirezionale $F_K : X \to sì$ tale che

$sì = F_K(X)$	facile, se K e X sono conosciuti
$X = F_K^{-1}(sì)$	facilmente, se K e $sì$ sono conosciuti
$X = F_K^{-1}(sì)$	irrealizzabile, se $sì$ è noto ma K non è noto

  K è il informazioni sulla botola.

- **Esempio:** Il calcolo delle radici cubiche modulari (sopra) è facile quando P e Q sono noti (teoria dei numeri di base) ma difficili se non sono noti.

Crittoanalisi a chiave pubblica

- **Attacchi di forza bruta** Contromisura: usate chiavi grandi!
  - Ma è necessario un compromesso poiché la complessità della crittografia/decrittografia potrebbe non scalare in modo lineare con la lunghezza della chiave.
  - In pratica: la crittografia a chiave pubblica è limitata a *gestione delle chiavi* e *firma digitale*.
- **Calcolo della chiave privata dalla chiave pubblica.** *Nessuna prova che questo attacco sia irrealizzabile!* (Anche per RSA)
- **Attacco di probabile messaggio.** Immagina un breve messaggio m (ad esempio una chiave DES a 56 bit) viene inviato crittografato con PU_{un}, cioè $C = E(PU_{un}, m)$. L'attaccante calcola tutto $s\grave{i}_{io} = E(PU_{un}, X_{io})$ per tutto il testo in chiaro possibile X_{io} per $io = 1, \ldots, 2^{56}$ e si ferma non appena $s\grave{i}_{io} = C$ concludendo che $m = X_{io}$ (il messaggio inviato).

  Soluzione: aggiungi alcuni bit casuali a m.

Contorno

Fattorizzazione primi

- Fattorizzare un numero n è scriverlo come prodotto di altri numeri: $n = un ? B ? C$.
- Moltiplicare i numeri è facile, fattorizzare i numeri sembra difficile.

  Non possiamo fattorizzare la maggior parte dei numeri con più di 1024 bit.
- Il *scomposizione in fattori primi* di un numero un equivale a scriverlo come un prodotto di potenze di numeri primi:

$$un = P?P\, P_{unP} = 2_{un2} ? 3_{un3} ? 5_{un5} ? 7_{un7} ? 11_{un11} \cdots$$

dove P è l'insieme dei numeri primi e $unP ? \mathbb{N}$.

Per esempio

$$\begin{aligned} 91 &= 7 ? 13 \\ 3600 &= 2_4 ? 3_2 ? 5_2 \end{aligned}$$

Fattorizzazione primi

- Fattorizzare un numero n è scriverlo come prodotto di altri numeri: $n = un ? B ? C$.
- Moltiplicare i numeri è facile, fattorizzare i numeri sembra difficile.

  Non possiamo fattorizzare la maggior parte dei numeri con più di 1024 bit.
- Il *scomposizione in fattori primi* di un numero un equivale a scriverlo come un prodotto di potenze di numeri primi:

$$un = P?P\, Pun_P = 2un_2 ? 3un_3 ? 5un_5 ? 7un_7 ? 11un_{11} \cdots$$

  dove P è l'insieme dei numeri primi e $un_P ?$ N.

  Per esempio

$$\begin{aligned} 91 &= 7 ? 13 \\ 3600 &= 2_4 ? 3_2 ? 5_2 \end{aligned}$$

Numeri relativamente primi e gcd

- Due numeri *un*, *B* sono *relativamente primo* se non hanno divisori/fattori comuni oltre a 1, cioè se gcd(*a, b*) = 1.

  Ad esempio, 8 e 15 sono relativamente primi poiché
  - i fattori di 8 sono 1,2,4,8, i fattori
  - di 15 sono 1,3,5,15 e 1 è l'unico
  - fattore comune.

- Viceversa possiamo determinare il *massimo comun divisore* confrontando le loro scomposizioni in fattori primi e utilizzando le potenze minime.

  Ad esempio, $300 = 2^2 \cdot 3^1 \cdot 5^2$, $18 = 2^1 \cdot 3^2$ quindi $\gcd(18, 300) = 2^1 \cdot 3^1 \cdot 5^0 = 6$

Aritmetica modulare

- $?un.?qr.\ (un = Q\ ?\ n + R)$ dove $0\ ?\ r < n$.

  Qui R è il *resto* . Scriviamo il resto come un modalità n .

- $a, b\ ?$ Z sono *congruente modulo n*, Se un modalità $n = B$ modalità n

  . Lo scriviamo come$un =_n B$.**Proprietà:**

-
  - $(un \bullet B) =_n (un$ modalità $n) \bullet (B$ modalità $n)$ per $\bullet \{+, -, *\}$

    cioè, $(un \bullet B)$ modalità $n = [(un$ modalità $n) \bullet (B$ modalità $n)]$ modalità n
  - Se $un\ ?\ B =_n un\ ?\ C$ e un è relativamente primo a n, poi $B =_n C$.
- **Esempi:**
  - $30 \bmod 4 =$

    $(5\ ?\ 6)$ modulo $4 =$

    $((5$ modalità $4)\ ?\ (6 \bmod 4)) \bmod 4 =$

    $(1\ ?\ 2) \bmod 4 = 2$
  - $8\ ?\ 4 =_3 8\ ?\ 1\ ?\ 4 =_3 1$

Funzione Totient di Eulero

- Quando si esegue l'aritmetica modulo nSet
- completo di *residui* è $0, \ldots, n-1$
- *Insieme ridotto di residui* consiste di quei numeri (*residui*) che sono relativamente primi a n

  Ad esempio, per $n = 10$:
  - l'insieme completo dei residui è $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - insieme ridotto di residui è $\{1, 3, 7, 9\}$
- Il *Funzione Totient di Eulero* $\phi(n)$ denota il numero di elementi nell'insieme ridotto di residui.

  Proprietà:

  $\phi(P) = p-1$ se P è primo

  $\phi(pq) = \phi(P)\phi(Q) = (p-1)(q-1)$ se P e Q sono primi.

Funzione Totient di Eulero

- Quando si esegue l'aritmetica modulo nSet
- completo di *residui* è $0, \ldots, n-1$
- *Insieme ridotto di residui* consiste di quei numeri (*residui*) che sono relativamente primi a n

  Ad esempio, per $n = 10$:
  - l'insieme completo dei residui è $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - insieme ridotto di residui è $\{1, 3, 7, 9\}$
- Il *Funzione Totient di Eulero* $\phi(n)$ denota il numero di elementi nell'insieme ridotto di residui.

  Proprietà:

  $\phi(P) = p-1$ se P è primo

  $\phi(pq) = \phi(P)\phi(Q) = (p-1)(q-1)$ se P e Q sono primi.

Teorema di Eulero

Teorema

$un^{\lambda(n)} =_n 1$ *per ogni a, n tale che* $\gcd(un) = 1$.

Esempi:

- Se $un = 3$ e $n = 10$, allora $\lambda(10) = 4$ e $3^4 = 81 =_{10} 1$
- Se $un = 2$ e $n = 11$, poi $\lambda(11) = 10$ e $2^{10} = 1024 =_{11} 1$

Teorema

$un^{\varphi(n)} =_n 1$ *per ogni a, n tale che* $\gcd(un) = 1$.

Esempi:

- Se $un = 3$ e $n = 10$, allora $\varphi(10) = 4$ e $3^4 = 81 =_{10} 1$
- Se $un = 2$ e $n = 11$, poi $\varphi(11) = 10$ e $2^{10} = 1024 =_{11} 1$

Teorema

$un?(n) =_n 1$ *per ogni a, n tale che* $\gcd(un) = 1$.

Esempi:

- Se $un = 3$ e $n = 10$, allora $?(10) = 4$ e $3_4 = 81 =_{10} 1$
- Se $un = 2$ e $n = 11$, poi $?(11) = 10$ e $2_{10} = 1024 =_{11} 1$

Contorno

L'algoritmo RSA

- Prende il nome dagli inventori: Rivest, Shamir, Adleman, 1978.
- Pubblicato dopo la sfida del 1976 da Diffie e Hellman.
- La sicurezza deriva dalla difficoltà di fattorizzare grandi numeri
  Le chiavi sono funzioni di una coppia di grandi, ≥ 100 cifre, numeri primi
- Algoritmo a chiave pubblica più popolare
  Utilizzato in molte applicazioni, ad es. PGP, PEM, SSL, ...

L'algoritmo RSA

- Permettere n essere un numero conosciuto da mittente e destinatario. Testo
- in chiaro diviso in blocchi di B tronco d'albero $_2(n)C$ bit.

  Quindi ogni blocco rappresenta un numero m tale che $M < n$. La
- crittografia e la decrittografia sono definite come segue:

$$C = m^e \text{ modalità } n$$

$$m = C^D \text{ modalità } n = (m^e)^D \text{ modalità } n = m^{ed} \text{ modalità } n$$

per alcuni (opportunamente scelti) valori di e e D. È un

- algoritmo di crittografia a chiave pubblica con
  - chiave pubblica $PU = (e, n)$ e
  - chiave privata $PR = (d, n)$.

L'algoritmo RSA

- Permettere n essere un numero conosciuto da mittente e destinatario. Testo
- in chiaro diviso in blocchi di B tronco d'albero $_2(n)C$ bit.

  Quindi ogni blocco rappresenta un numero m tale che $M < n$. La
- crittografia e la decrittografia sono definite come segue:

$$C = m_e \quad \text{modalità } n$$

$$m = C_D \quad \text{modalità } n = (m_e)_D \text{ modalità } n = m_{ed} \quad \text{modalità } n$$

  per alcuni (opportunamente scelti) valori di e e D. È un
- algoritmo di crittografia a chiave pubblica con
  - chiave pubblica $PU = (e, n)$ e
  - chiave privata $PR = (d, n)$.

L'algoritmo RSA

- Permettere n essere un numero conosciuto da mittente e destinatario. Testo
- in chiaro diviso in blocchi di B tronco d'albero $_2(n)C$ bit.

  Quindi ogni blocco rappresenta un numero m tale che $M < n$. La
- crittografia e la decrittografia sono definite come segue:

$$C = m^e \quad \text{modalità } n$$

$$m = C^D \quad \text{modalità } n = (m^e)^D \text{ modalità } n = m^{ed} \quad \text{modalità } n$$

per alcuni (opportunamente scelti) valori di e e D. È un

- algoritmo di crittografia a chiave pubblica con
  - chiave pubblica $PU = (e, n)$ e
  - chiave privata $PR = (d, n)$.

L'algoritmo RSA

- Permettere n essere un numero conosciuto da mittente e destinatario. Testo
- in chiaro diviso in blocchi di B tronco d'albero $_2(n)C$ bit.

  Quindi ogni blocco rappresenta un numero m tale che $M < n$. La
- crittografia e la decrittografia sono definite come segue:

$$C = m^e \quad \text{modalità } n$$
$$m = C^D \quad \text{modalità } n = (m^e)^D \text{ modalità } n = m^{ed} \quad \text{modalità } n$$

  per alcuni (opportunamente scelti) valori di e e D. È un
- algoritmo di crittografia a chiave pubblica con
  - chiave pubblica $PU = (e, n)$ e
  - chiave privata $PR = (d, n)$.

Affinché l'algoritmo RSA funzioni, devono essere soddisfatti i seguenti requisiti:

1. È possibile trovare valori di e, D, e n tale che m_{ed} modalità $n = m$ per tutti $M < n$.
2. È relativamente facile da calcolare m_e modalità n e C_D modalità n per tutti i valori di $M < n$.
3. Non è fattibile determinare D dato e e n.

Correttezza di RSA

Definizione (inversi moltiplicativi)

x e *y* sono *inverse moltiplicative* modalità *R* se *xy* modalità *R* = 1.

Teorema (correttezza di RSA)

Se d ed e sono inversi moltiplicativi modalità $\varphi(n)$, *ovvero se ed mod* $\varphi(n) = 1$, *quindi* M^{ed} modalità $n = M$ *per tutti* $M < n$.

Lemma

Siano p e q numeri primi e $n = pq$. *Se d ed e sono inversi moltiplicativi* modalità $\varphi(n)$, *quindi* $M^{ed} \equiv_P M$ *e* $M^{ed} \equiv_Q M$, *cioè* $(m^{ed} - M)$ *è multiplo di p e q.*

Prova di correttezza di RSA.

Permettere *D* e *e* sono moltiplicativi inversi modalità $\varphi(n)$. Per Lemma, $(m^{ed} - M)$ è multiplo di *P* e *Q*. Da quando *P* e *Q* sono primi allora $(m^{ed} - M)$ è anche multiplo di *pq* e quindi di *n*. □

Correttezza della RSA – continua

Dimostrazione di Lemma.

Permettere D e e sono moltiplicativi inversi modalità $?(n)$. Allora esiste un intero K tale che $ed = k\varphi(n) + 1$.
Dobbiamo dimostrare che $m^{ed} = m^{k\varphi(n)+1}$ $?_P$ m. Due casi:
Caso 1: m e P sono relativamente primi.

$$
\begin{aligned}
m^{k\varphi(n)+1} \text{ modalità } P &= m \cdot m^{K(p-1)(q-1)} \text{ modalità } P \\
&= m \cdot (m^{(p-1)})^{K(q-1)} \text{ modalità } P \\
&= m \cdot (m^{?(P)})^{K(q-1)} \text{ modalità } P \text{ da } P \text{ è primo} \\
&= m \cdot (1)^{K(q-1)} \text{ modalità } P \text{ dal teorema di Eulero} = m \\
&\text{modalità } P
\end{aligned}
$$

Caso 2: m e P non sono relativamente primi. Quindi m è un multiplo di P, cioè m modalità $P = 0$ e quindi $m^{k\varphi(n)+1}$ modalità $P = m$ modalità P. □

Algoritmi RSA

- Genera una coppia di chiavi pubblica/privata:
  1. Genera due (grandi) numeri primi distinti P e Q
  2. . Calcolare$n = pq$ e $? = (p-1)(q-1)$. Seleziona un
  3. e, $1 < e <$, relativamente primo a $?$. Calcolare$D$
  4. tale che ed modalità $? = 1$.
  5. Pubblicare (e, n), mantenere (d, n) privato, scartare P e Q.
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D_{io}}$ modalità n.

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. Genera due (grandi) numeri primi distinti P e Q
  2. . Calcolare$n = pq$ e $? = (p-1)(q-1)$. Seleziona un
  3. e, $1 < e <$, relativamente primo a $?$. Calcolare$D$
  4. tale che ed modalità $? = 1$.
  5. Pubblicare (e, n), mantenere (d, n) privato, scartare P e Q.
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D_{io}}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq$ e $? = (p-1)(q-1)$. Seleziona un
  3. e, $1 < e <$, relativamente primo a $?$. Calcolare D
  4. tale che ed modalità $? = 1$.
  5. Pubblicare (e, n), mantenere (d, n) privato, scartare P e Q.
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e\,io}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $?= (p-1)(q-1) = 46\ ?\ 70 = 3220$Seleziona
  3. un e, $1 < e <$, relativamente primo a $?$. Calcolare$D$ tale che ed
  4. modalità $?= 1$.
  5. Pubblicare (e, n), mantenere (d, n) privato, scartare P e Q.
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?\ 70 = 3220$Scegliere
  3. $e = 79$ (casualmente nell'intervallo $[1..3220]$) Calcola D tale che ed
  4. modalità $? = 1$.
  5. Pubblicare (e, n), mantenere (d, n) privato, scartare P e Q.
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e\,io}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo $[1..3220]$) Calcola D tale che $79 \cdot D$
  4. modalità $3220 = 1$: $D = 1019$Pubblicare (e, n), mantenere (d, n) privato,
  5. scartare P e Q.
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D_{io}}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo [1..3220]) Calcola D tale che $79 \cdot D$
  4. modalità $3220 = 1$: $D = 1019$Chiave pubblica $(e, n) = (79, 3337)$, chiave
  5. privata $(d, n) = (1019, 3337)$
- Crittografia con chiave (e, n)
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo [1..3220]) Calcola D tale che $79 \cdot D$
  4. modalità $3220 = 1$: $D = 1019$Chiave pubblica $(e, n) = (79, 3337)$, chiave
  5. privata $(d, n) = (1019, 3337)$
- Crittografia con chiave $(e, n) = (79, 3337)$
  1. Interrompi messaggio m in blocchi $m_1 m_2 \cdots$ insieme a $m_{io} < n$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D_{io}}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo [1..3220]) Calcola D tale che $79 \cdot D$
  4. modalità $3220 = 1$: $D = 1019$Chiave pubblica $(e, n) = (79, 3337)$, chiave
  5. privata $(d, n) = (1019, 3337)$
- Crittografia con chiave $(e, n) = (79, 3337)$
  1. Interrompi messaggio m in blocchi, ad es 688 232 687 966 668 $\cdots$
  2. Calcolare $C_{io} = m_{e_{io}}$ modalità n.
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo [1..3220]) Calcola D tale che $79 \cdot D$
  4. modalità 3220 = 1: $D = 1019$Chiave pubblica $(e, n) = (79, 3337)$, chiave
  5. privata $(d, n) =$ (1019, 3337)
- Crittografia con chiave (e, n)= (79, 3337)
  1. Interrompi messaggio m in blocchi, ad es 688 232 687 966 668 ···
  2. Calcolare $C_1 = 688^{79}$ modalità 3337 = 1570, $C_2 = ...$
- decrittazione con chiave (d, n):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo [1..3220]) Calcola D tale che $79 \cdot D$
  4. modalità 3220 = 1: $D = 1019$Chiave pubblica $(e, n) = (79, 3337)$, chiave
  5. privata $(d, n) =$ (1019, 3337)
- Crittografia con chiave (e, n)= (79, 3337)
  1. Interrompi messaggio m in blocchi, ad es 688 232 687 966 668 $\cdots$
  2. Calcolare $C_1 = 688^{79}$ modalità 3337 = 1570, $C_2 = \ldots$
- decrittazione con chiave (d, n)= (1019, 3337):
  1. Calcolare $m_{io} = C_{D\ io}$ modalità n

Algoritmi RSA: Esempio

- Genera una coppia di chiavi pubblica/privata:
  1. creare $P = 47$, $Q = 71$
  2. Calcolare $n = pq = 3337$ e $? = (p-1)(q-1) = 46\ ?70 = 3220$Scegliere $e =$
  3. 79 (casualmente nell'intervallo [1..3220]) Calcola D tale che $79 \cdot D$
  4. modalità $3220 = 1$: $D = 1019$Chiave pubblica $(e, n) = (79, 3337)$, chiave
  5. privata $(d, n) = (1019, 3337)$
- Crittografia con chiave $(e, n) = (79, 3337)$
  1. Interrompi messaggio m in blocchi, ad es 688 232 687 966 668 $\cdots$
  2. Calcolare $C_1 = 688^{79}$ modalità $3337 = 1570$, $C_2 = \ldots$
- decrittazione con chiave $(d, n) = (1019, 3337)$:
  1. Calcolare $m_1 = 1570^{1019}$ modalità $3337 = 688$, $m_2 = \ldots$

Calcolo dell'inverso moltiplicativo

Adattando l'algoritmo euclideo esteso:

```
1  funzione inverso (a, n)
2      t := 0; nt := 1; r := n;
3      nr := a;mentre nr != 0
4
5          q := r div numero;
6          (t, nt) := (nt, tq*nt); (r, nr) := (nr,
7          rq*nr);
8      Se r > 1 poi restituire "a is non invertibile";Se t < 0 poi t :=
9      t + n; restituire t;
10
```

Sicurezza RSA

- Calcolo del segreto D dato (e, n)
  - Difficile come fattorizzare. Se possiamo fattorizzare $n = pq$ allora possiamo calcolare $? = (p-1)(q-1)$ e quindi D.
  - Nessun algoritmo di tempo polinomiale noto.
    Ma visti i progressi nel factoring, n dovrebbe avere almeno 1024 bit.
- Calcolo di m_{io}, dato C_{io}, e (e, n)
  - Non è chiaro (= nessuna prova) se è necessario calcolare D, cioè fattorizzare n.

? I progressi nella teoria dei numeri potrebbero rendere insicura RSA.

Malleabilità di RSA

- Ricordiamo che una funzione crittografica $E(K, M)$ è *malleabile* se esistono due funzioni $F(X)$ e $G(X)$ tale che

$$F(E(K, M)) = E(KG(m)) \text{ per tutte le chiavi } K \text{ e messaggi } m$$

- Crittografia RSA, vale a dire $E((e, n), m)) = m^e$ modalità n è chiaramente malleabile.
  Permettere $F(X) = X ? (m_1^e$ modalità $n)$ per qualsiasi dato m_1.

$$F(E((e, n), m)) = E((e, n), m) ? (m^e \text{ modalità } n) =$$
$$= (m^e \text{ modalità } n) ? (m^e \text{ modalità } n) = (m ? m_1)^e \text{ modalità } n =$$
$$= E((e, n), m ? m_1) = E((e, n), G(m))$$

- Per questo motivo, RSA è comunemente usato insieme a metodi di riempimento come OAEP o PKCS1.

Contorno

Algoritmi asimmetrici per la distribuzione di chiavi segrete

- Utilizza algoritmi di crittografia a chiave pubblica per supportare la crittografia simmetrica (più veloce).
- Vedremo due approcci:
  - Distribuzione di chiavi segrete con
  - scambio di chiavi RSA Diffie-Hellman

Contorno

Distribuzione della chiave segreta con RSA

- Crittografia di m (con chiave pubblica (e, n))
  - scegliere K a caso
  - c = (K_e modalità n, $E_K(m)$)
- Decrittografia (con chiave privata (d, n))
  - Diviso C in (C_1, C_2)
  - $K = C_D$ modalità n $\qquad m = D_K(C_2)$
- **Esempio:** SSL
  Alice sceglie un segreto, lo crittografa con il PK di Bob e il resto della sessione è protetto in base a quel segreto.
- **Problema:** se la chiave privata (d, n) viene compromesso, allora K può essere recuperato da un intruso dal traffico osservato in precedenza.

Contorno

Background sui logaritmi discreti

- UN radice primitiva S di un numero primo P è un numero le cui potenze generano 1 $, \ldots, p-1$.
  Così S modalità P, S_2 modalità P, . . ., S_{p-1} modalità P sono distinti, cioè una permutazione da 1 a $p-1$.
  Quindi:

$$?B\ ?\ \mathrm{Z}.\exists io\ \{0, \ldots, p-1\}.\ B = S_{io} \text{ modalità } P$$

- Dato $B\ ?\ \mathrm{Z}$, esponente io sopra è il logaritmo discreto di B per base S, modalità P. Il
- calcolo di log discreti sembra impossibile.

Scambio di chiavi Diffie-Hellman

1. I principali condividono il primo Q e radice primitiva α di Q. Entrambi possono
2. essere pubblici. UN e B generare numeri casuali X_{UN} e X_B (risp.) entrambi inferiori
3. a Q. UN calcola $si_{UN} = \alpha^{X_{UN}}$ modalità Q, B calcola $si_B = \alpha^{X_B}$ modalità Q. UN e B
4. scambiare i risultati.
5. UN calcola $K_{UN} = si^{X_{UN}}_B$ modalità Q, B calcola $K_B = si^{X_B}_{UN}$ modalità Q.
   Le chiavi sono uguali, cioè $K_{UN} = K_B$:

$$
\begin{aligned}
K_{UN} &= si_B^{X_{UN}} \text{ modalità } Q \\
&= (\alpha^{X_B})^{X_{UN}} \text{ modalità } Q \\
&= (\alpha^{X_{UN}})^{X_B} \text{ modalità } Q \\
&= si_{UN}^{X_B} \text{ modalità } Q = K_B
\end{aligned}
$$

La sicurezza dipende dalla difficoltà di elaborazione di log discreti.

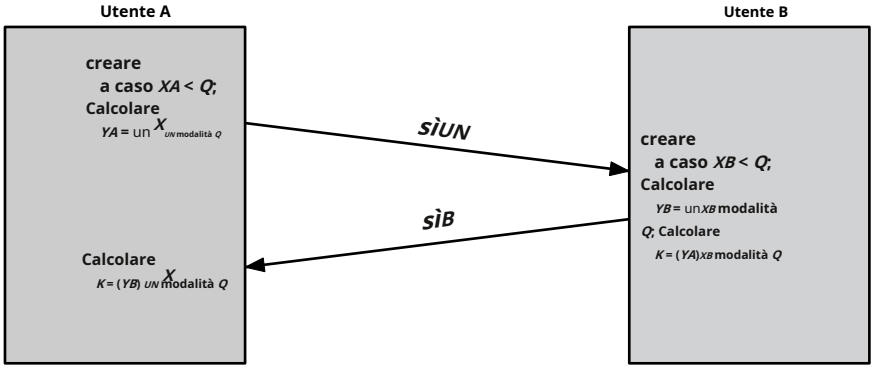


Figura 10.8 Scambio di chiavi Diffie-Hellman

Diffie-Hellman: punti di forza

- Il segreto condiviso viene creato dal nulla!
- Il segreto condiviso non viene mai trasmesso (nemmeno in forma crittografata).

? **Perfetta segretezza in avanti** (PFS), cioè se qualcuno registra l'intera conversazione e poi scopre le chiavi private di Alice e/o Bob, non sarà in grado di decifrare nulla!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle* :

1. A trasmette si_{UN} essere
2. io intercetta si_{UN} e trasmette si_D a B. Calcolo anche io $K_2 = (si_{UN})^{x_{D2}}$ modalità Q.
3. B riceve si_{D1} e calcola $K_B = (si_{D})^{x_B}$ modalità Q
4. B trasmette si_B ad A
5. io intercetta si_B e trasmette si_{D_2} ad A. io calcolo $K_1 = (si_B)^{x_{D1}}$ modalità Q.
6. A riceve si_{D2} e calcola $K_{UN} = (si_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $sì_{UN}$ essere
2. io intercetta $sì_{UN}$ e trasmette $sì_D$ a B. Calcolo anche io $K_2 = (sì_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $sì_{D1}$ e calcola $K_B = (sì$ $_{D})^{x_B}$ modalità Q
4. B trasmette $sì_B$ ad A
5. io intercetta $sì_B$ e trasmette $sì_D$ $_2$ ad A. io calcolo $K_1 = (sì_B)^{x_{D1}}$ modalità Q.
6. A riceve $sì_{D2}$ e calcola $K_{UN} = (sì_D)^{x_{UN}}$ modalità Q $_2$

Ora UN e B pensano che condividano una chiave segreta, ma invece UN condivide la chiave segreta K_2 insieme a io e B condivide la chiave segreta K_1 insieme a io.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $si\grave{}_{UN}$ essere
2. io intercetta $si\grave{}_{UN}$ e trasmette $si\grave{}_D$ a B. Calcolo anche io $K_2 = (si\grave{}_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $si\grave{}_{D1}$ e calcola $K_B = (si\grave{}_{D})^{x_B}$ modalità Q
4. B trasmette $si\grave{}_B$ ad A
5. io intercetta $si\grave{}_B$ e trasmette $si\grave{}_{D2}$ ad A. io calcolo $K_1 = (si\grave{}_B)^{x_{D1}}$ modalità Q.
6. A riceve $si\grave{}_{D2}$ e calcola $K_{UN} = (si\grave{}_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $si\grave{a}_{UN}$ essere
2. io intercetta $si\grave{a}_{UN}$ e trasmette $si\grave{a}_D$ a B. Calcolo anche io $K_2 = (si\grave{a}_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $si\grave{a}_{D1}$ e calcola $K_B = (si\grave{a}_D)^{x_B}$ modalità Q
4. B trasmette $si\grave{a}_B$ ad A
5. io intercetta $si\grave{a}_B$ e trasmette $si\grave{a}_{D2}$ ad A. io calcolo $K_1 = (si\grave{a}_B)^{x_{D1}}$ modalità Q.
6. A riceve $si\grave{a}_{D2}$ e calcola $K_{UN} = (si\grave{a}_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $si\grave{a}_{UN}$ essere
2. io intercetta $si\grave{a}_{UN}$ e trasmette $si\grave{a}_D$ a B. Calcolo anche io $K_2 = (si\grave{a}_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $si\grave{a}_{D1}$ e calcola $K_B = (si\grave{a}_{D1})^{x_B}$ modalità Q
4. B trasmette $si\grave{a}_B$ ad A
5. io intercetta $si\grave{a}_B$ e trasmette $si\grave{a}_{D2}$ ad A. io calcolo $K_1 = (si\grave{a}_B)^{x_{D1}}$ modalità Q.
6. A riceve $si\grave{a}_{D2}$ e calcola $K_{UN} = (si\grave{a}_{D2})^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $sì_{UN}$ essere
2. io intercetta $sì_{UN}$ e trasmette $sì_D$ a B. Calcolo anche io $K_2 = (sì_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $sì_{D1}$ e calcola $K_B = (sì_{D})^{x_B}$ modalità Q
4. B trasmette $sì_B$ ad A
5. io intercetta $sì_B$ e trasmette $sì_{D2}$ ad A. io calcolo $K_1 = (sì_B)^{x_{D1}}$ modalità Q.
6. A riceve $sì_{D2}$ e calcola $K_{UN} = (sì_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $sì_{UN}$ essere
2. io intercetta $sì_{UN}$ e trasmette $sì_D$ a B. Calcolo anche io $K_2 = (sì_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $sì_{D1}$ e calcola $K_B = (sì_{D})^{x_B}$ modalità Q
4. B trasmette $sì_B$ ad A
5. io intercetta $sì_B$ e trasmette $sì_{D2}$ ad A. io calcolo $K_1 = (sì_B)^{x_{D1}}$ modalità Q.
6. A riceve $sì_{D2}$ e calcola $K_{UN} = (sì_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $sì_{UN}$ essere
2. io intercetta $sì_{UN}$ e trasmette $sì_D$ a B. Calcolo anche io $K_2 = (sì_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $sì_{D1}$ e calcola $K_B = (sì_{D})^{x_B}$ modalità Q
4. B trasmette $sì_B$ ad A
5. io intercetta $sì_B$ e trasmette $sì_{D2}$ ad A. io calcolo $K_1 = (sì_B)^{x_{D1}}$ modalità Q.
6. A riceve $sì_{D2}$ e calcola $K_{UN} = (sì_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!

Diffie-Hellman: debolezze

Le chiavi sono non autenticato e quindi è vulnerabile a quanto segue *Attacco man-in-the-middle*:

1. A trasmette $sì_{UN}$ essere
2. io intercetta $sì_{UN}$ e trasmette $sì_D$ a B. Calcolo anche io $K_2 = (sì_{UN})^{x_{D2}}$ modalità Q.
3. B riceve $sì_{D1}$ e calcola $K_B = (sì_{D})^{x_B}$ modalità Q
4. B trasmette $sì_B$ ad A
5. io intercetta $sì_B$ e trasmette $sì_{D_2}$ ad A. io calcolo $K_1 = (sì_B)^{x_{D1}}$ modalità Q.
6. A riceve $sì_{D2}$ e calcola $K_{UN} = (sì_D)^{x_{UN}}$ modalità Q

Ora *UN* e *B* pensano che condividano una chiave segreta, ma invece *UN* condivide la chiave segreta K_2 insieme a *io* e *B* condivide la chiave segreta K_1 insieme a *io*.

Soluzione: firmare gli esponenti. Ma questo richiede chiavi condivise!