Algoritmi Asimmetrici

(aka Chiavi Asimmetriche / chiave pubblica-privata)

Indice

Cifratura a Chiave Pubblica

- Diffie ed Hellman – 1976
- Il primo **rivoluzionario** progresso nella crittografia
- Basati su funzioni matematiche, non su manipolazioni dei bit
- **Asimmetrico**, ovvero richiede due chiavi separate
- Impatta su **Confidenzialità, distribuzione delle chiavi, autenticazione**

Fondamenti degli algoritmi asimmetrici

- Testo in chiaro (Plaintext): messaggio in input all'algoritmo
- Algoritmo di cifratura: trasformazione del testo in chiaro in testo crittato
- **Chiave pubblica e chiave privata**: la prima usata per ricevere comunicazioni cifrata, la seconda per decifrarle
- Testo cifrato (Ciphertext): messaggio crittato
- Algoritmo di decifratura: produce il plaintext

Firma digitale

La firma digitale fornisce

- autenticazione
- Integrità
- Non-Ripudio

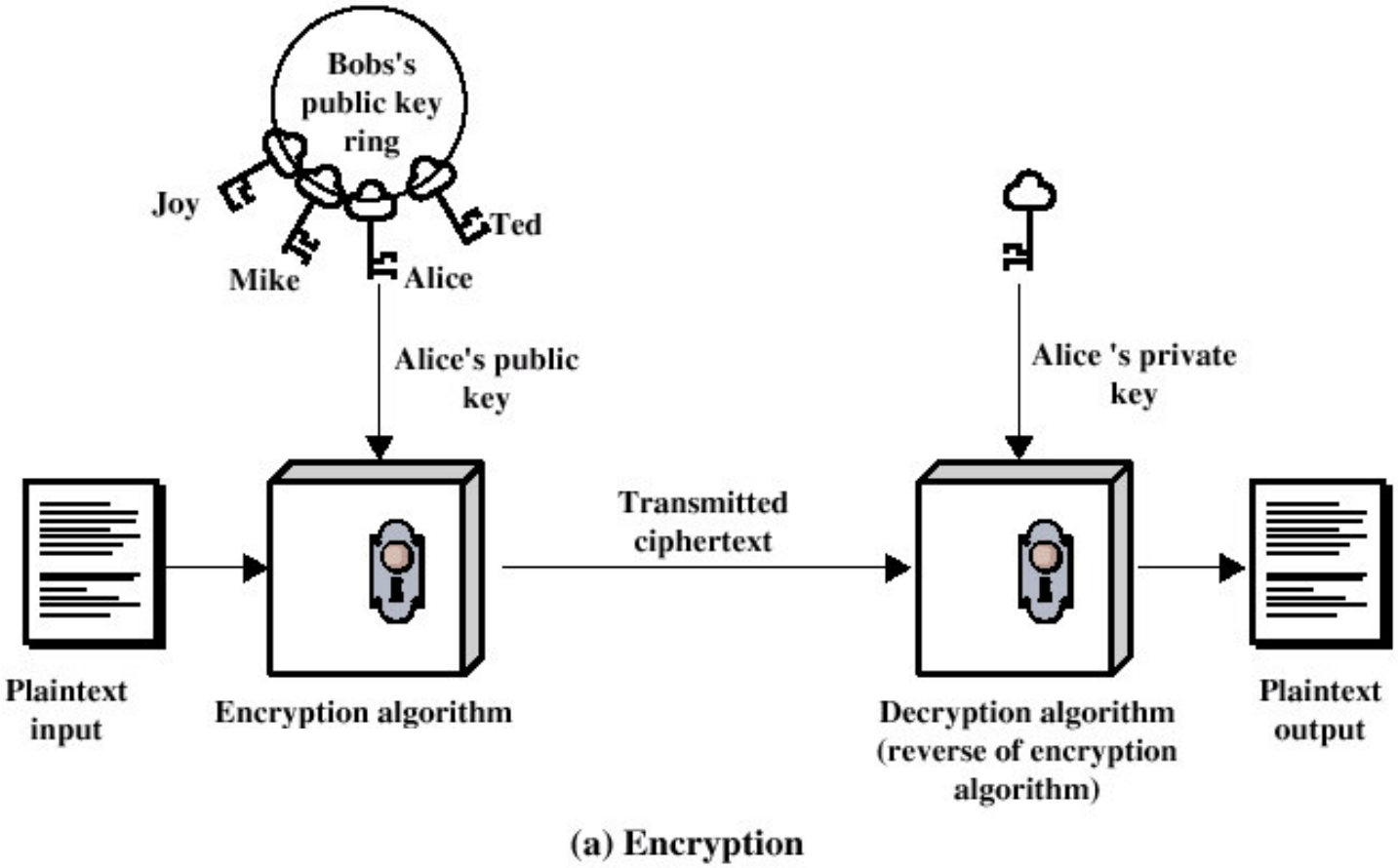
Esempio di applicazione

- Certificazione di chiavi pubbliche in ambienti distribuiti

Firma digitale

- Definizioni
  - Firma digitale – una stringa che associa un messaggio ad un originatore
  - Algoritmo di generazione di firma digitale– un meotdo per produrre una firma digitale
  - Schema di firma digitale – consiste di un algoritmo di generazione e di un algoritmo di verifica della firma digitale

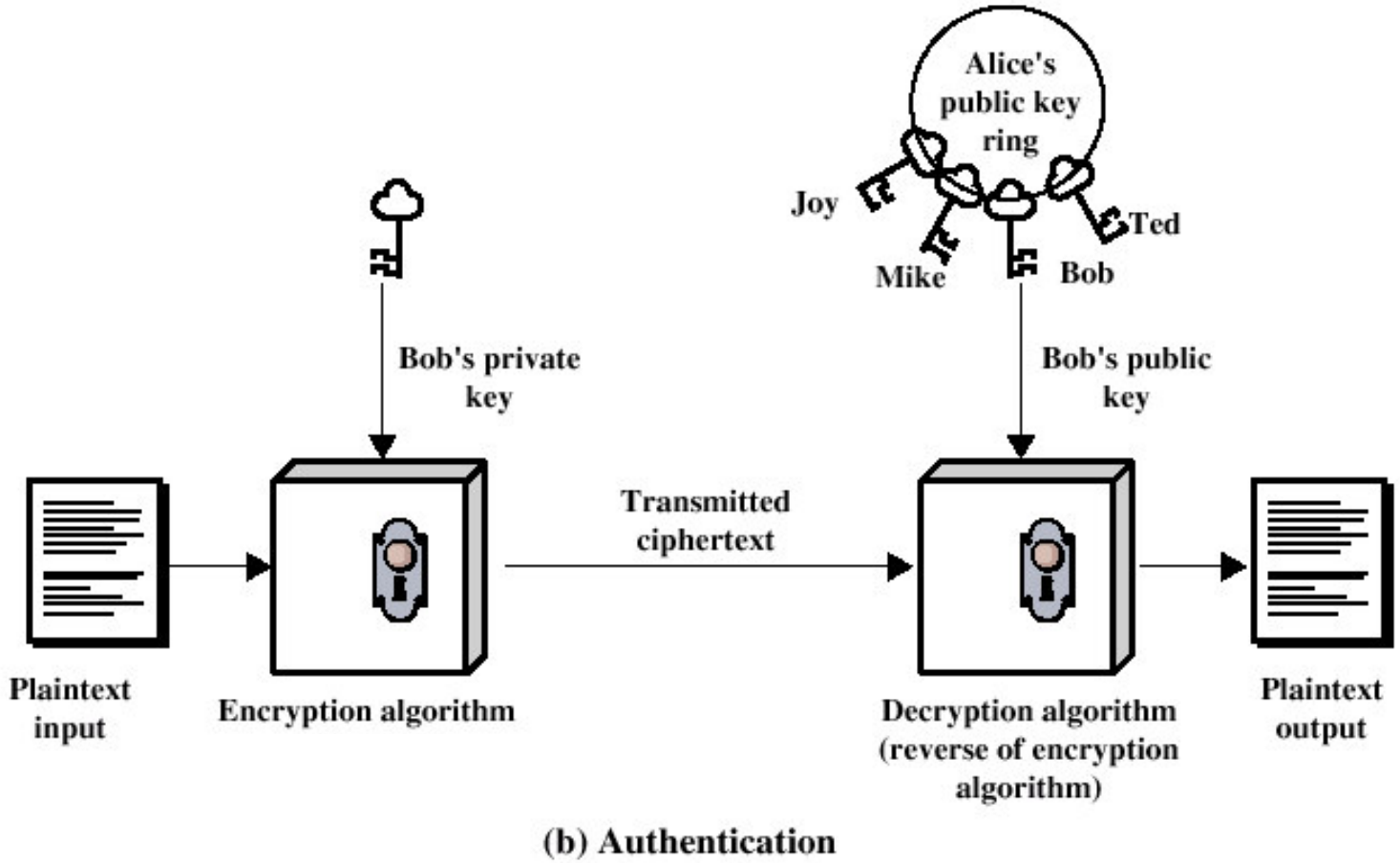
Modello di funzionamento algoritmi asimmetrici



I Fondamentali

- Ciascun utente genera una coppia di chiavi
- La chiave pubblica viene pubblicizzata
- La chiave segreta va custodita gelosamente
- Se Bob desidera inviare un messaggio ad Alice, Bob *cifra* il messaggio utilizzando la *chiave pubblica* di Alice
- Quando Alice riceve il messaggio, lo decifra utilizzando la sua *chiave privata*

Autenticazione con algoritmi Asimmetrici



(b) Authentication

Rotture

- Corruzione (rottura) di uno schema di firma
  - *Rottura totale*: la chiave privata è compromessa
  - *Falsificazione selettiva*: un'avversario può creare una firma valida su un messaggio selezionato
  - *Falsificazione esistenziale*: l'avversario può creare una firma valida, su un messaggio casuale (che non controlla)

Diffie-Hellman

- Diffie-Hellman è uno schema per la distribuzione di chiavi asimmetriche
- Primo schema di tipo pubblico, proposto nel 1976.
  - W Diffie, M E Hellman, "New directions in Cryptography", IEEE Trans. Information Theory, IT-22, pp644-654, Nov 1976

Diffie-Hellman

- Non è utile per scambiarsi messaggi
- Scambia solo chiavi, il cui valore dipende dai partecipanti
- L'algoritmo è basato sull'esponenziazione in GF
  - L'esponenziazione richiede $O((\log n)^3)$ operazioni

Diffie-Hellman

- La sicurezza si basa sulla difficoltà di calcolo del del logaritmo in GF
  - Risolvere il problema del logaritmo discreto richiede $O(e^{\log n \log \log n})$ operazioni
- Non c'è autenticazione degli attori

Diffie-Hellman

- L'algoritmo:
  - Alice e Bob desiderano scambiarsi una chiave su un canale di comunicazione insicuro.
  - Selezionano un primo grande p (~512 bit), tale che (p-1)/2 è anch'esso primo
  - Selezionano g, una radice primitiva mod p
    - g è radice primitiva se per ogni *n* da 0 a p-1, esiste un *a* tale che $g^a = n \bmod p$.
- *p* e *g* sono pubbliche

Diffie-Hellman (DH)

- Protocollo

Alice	Bob
Sceglie S_a random	sceglie S_b random
calcola $T_A=g^{Sa} \bmod p$	compute $T_B=g^{Sb} \bmod p$
invia T_A a Bob	
	riceve T_A da Alice
Riceve T_B da Bob	invia T_B a Alice
Calcola $T_B^{Sa} \bmod p$	
	calcola $T_A^{Sb} \bmod p$

Alice e Bob calcolano lo stesso valore $g^{SaSb} \bmod p$, che viene usato come chiave.

Valutazione del DH

- Fattori di sicurezza
  - $T = g^s \bmod p$
  - Congettura: dati T, g, p, è estremamente difficile calcolare il valore s (ovvero il *logaritmo discreto*)
  - La chiave segreta non è mai trasmessa.
- Svantaggi:
  - Esponenziazioni sono onerese
    - DoS possibile.
  - Lo schema non serve per la cifratura
  - Non c'è autenticazione tra i partecipanti...

Man In The Middle

Alice	Trudy	Bob
g^{Sa}=123	g^{Sx} =654	g^{Sb} =255
123 -->	654 -->	
	<--654	<--255
654^{Sa}=123^{Sx}	255^{Sx}=654^{Sb}	

- Trudy finge di essere Bob nei confronti di Alice ed Alice nei confronti di Bob

DH in modalità elenco telefonico (phone-book mode)

- DH soggetto al man-in-the-middle perchè le chiavi pubbliche sono intercettate e sostituite
- La modalità elenco telefonico consente ai partecipanti di generare le chiavi pubbliche e di rendere note via canali affidabili, e.g. $< T_B >$ per Bob
- Tutti i partecipanti concordano su $<g, p>$

Cifratura con Diffie-Hellman

- Ogni attore calcola e pubblica $<p, g, T>$
  - $T=g^S \bmod p$
- Alice vuole comunicare con Bob:
  - Alice
    - Sceglie un segreto random S_a
    - Usa $K_{ab} = T_b^{Sa} \bmod p_b$ per cifrare
    - Calcola $g_b^{Sa} \bmod p_b$
    - Invia il messaggio cifrato assieme ad $g_b^{Sa} \bmod p_b$
  - Bob
    - $(g_b^{Sa})^{Sb} \bmod p_b = (g_b^{Sb})^{Sa} \bmod p_b = T_b^{Sa} \bmod p_b = K_{ab}$
    - Usa K_{ab} per decifrare

El Gamal

- E' una variante dello schema Diffie-Hellman, che consente anche lo scambio sicuro di messaggi
- Pubblicato nel 1985 da ElGamal in
  - T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Trans. Information Theory, vol IT-31(4), pp469-472, July 1985.
- Come per Diffie-Hellman la sicurezza si basa sul problema del logaritmo discreto

El Gamal

- Generazione delle chiavi
- Selezioniamo un grande p (~512 digit), e g una radice primitiva mod p
  - Bob ha il segreto x_B
  - Bob calcola y_B, e rende tale valore pubblico
    - $y_B = g^{x_B} \bmod p$
- Per cifrare il messaggio **M**
  - Seleziona un numero casuale **k**, $0 <= k <= p-1$
  - Calcola la chiave **K**
    - $K = (y_B)^k \bmod p$

El Gamal

- Calcola la coppia –cyphertext-: *C* = {c1,c2}
  - $c_1 = g^k \bmod p, \quad c_2 = K*M \bmod p$
- Per decifrare il messaggio
  - Estrai la chiave **K**
    - $K = c_1^{x_B} \bmod p = g^{k*x_B} \bmod p$
  - Estrai il messaggio **M** risolvendo (per **M**) l'equazione:
    - $c_2 = K*M \bmod p$

El Gamal

- El Gamal può anche essere usato (con una piccola variante) per firmare messaggi
- Come nello schema di cifratura, la chiave pubblica sarà $y = g^x \bmod p$, assieme a g e p
- Per firmare un messaggio:
  - Bob seleziona un numero casuale k tale che $gcd(k,p-1)=1$
  - Bob calcola $a = g^k \bmod p$
  - Bob risolve l'equazione $M = x*a + k*b \pmod{p-1}$ in b.
  - La firma è la coppia a and b

El Gamal

- Per verificare la firma:
  - Controlla che $y^a a^b \bmod p = g^M \bmod p$
- Esempio (1/2):
  - p = 11, g = 2. Bob sceglie x = 8.
  - $y = 2^8 \bmod 11 = 3$
  - Chiave pubblica: y = 3, g = 2, p = 11
  - Bob vuole firmare M = 5
  - Sceglie k = 9 (nota che gcd(9, 10) = 1)
    - $a = 2^9 \bmod 11 = 6$

El Gamal

- Esempio (2/2)
  - Risolve: 5 = 8*6 + 9 * b mod 10
  - b = 3
  - La firma è *a = 6, b = 3*
  - Per verificare la firma, testare che:
    - $3^{6}6^{3} \bmod 11 = 2^{5} \bmod 11$

RSA

- Dal nome dei suoi inventori: Ron Rivest, Adi Shamir and L. Adleman
  - R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
- Uno schema a chiave pubblica che può essere utilizzato crittare i messaggi, scambiarsi chiavi e creare firme digitali

RSA

- Basato sull'esponenziazione in campi finiti (Galois) su interi modulo un primo
  - L'esponenziazione richiede $O((\log n)^3)$ operazioni
- La sicurezza risiede nella difficoltà di fattorizzazione di numeri molto grandi
  - La fattorizzazione richiede $O(e^{\log n \log \log n})$ operazioni
  - (lo stesso bound che per il logaritmo discreto)
- Brevetto scaduto nel 2000

RSA

- Il più diffuso e conosciuto
- Utilizzabile con chiavi di diversa lunghezza (oggigiorno 1024 bits).
- Il plaintext può avere lunghezza del blocco variabile.
  - Plaintext deve essere più corto della chiava.
  - Il Ciphertext ha la stessa dimensione della chiave.

Come scelgo i parametri RSA?

- Per generare una coppia di chiavi:
  - Scegliamo due primi *p,q* grandi (>= 512 bits each)
  - Calcoliamo $n = p*q$
  - Come chiave pubblica, scegliamo un *e* che è un primo relativo a $\phi(n) = (p-1)(q-1)$, poniamo

    pub = $\langle e,n \rangle$
  - Chiave privata: troviamo un *d* che è l'inverso di *e* mod $\phi(n)$, i.e., $e*d = 1 \bmod \phi(n)$, e poniamo priv = $\langle d,n \rangle$

Primitive per RSA

- Dati pub = $\langle e, n \rangle$ e priv = $\langle d, n \rangle$
  - cifratura: $c = m^e \bmod n$, $m < n$
  - decifratura: $m = c^d \bmod n$
  - firma: $s = m^d \bmod n$, $m < n$
  - verifica: $m = s^e \bmod n$

Perchè RSA funziona?

- Dati pub = $\langle e, n \rangle$ e priv = $\langle d, n \rangle$
  - $n = p*q$, $\phi(n) = (p-1)(q-1)$
  - $e*d = 1 \bmod \phi(n)$
  - $x^{e*d} = x \bmod n$
  - encryption: $c = m^e \bmod n$
  - decryption: $m = c^d \bmod n = m^{e*d} \bmod n = m \bmod n = m$ (since $m < n$)
  - digital signature (similare)

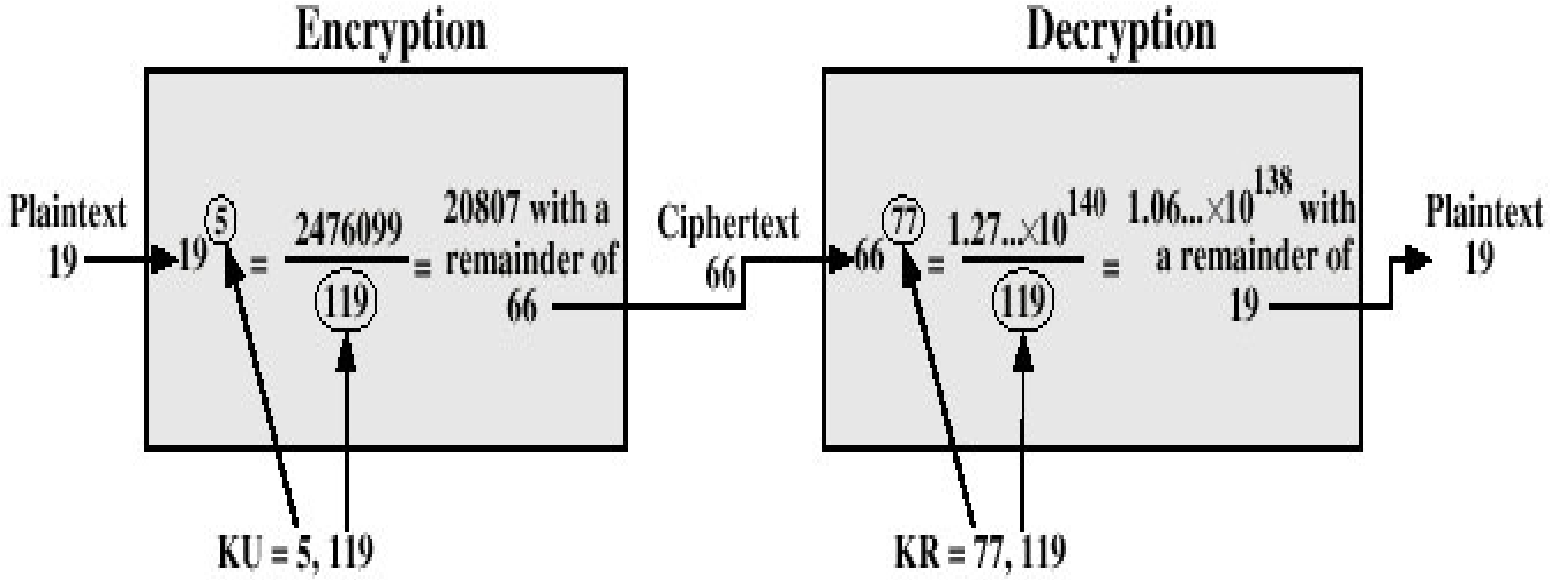
Sicurezza RSA

- Fattorizzare 1024-bit è molto difficile!
- Ma, se vi si riuscisse, allora, data la chiave pubblica <*e,n*>, sarebbe possibile trovare *d*, e quindi la chiave privata via:
  - Da *n* potremmo ricavare *p*, q, tali che
    $n = p*q$
  - quindi $\phi(n) = (p-1)(q-1)$
  - infine *d* tale che $e*d = 1 \bmod \phi(n)$

Esempio di uso RSA (1/2)

- Selezioniamo due primi , $p=7$ and $q=17$
- Calcoliamo $n = pq = 7 \times 17 = 119$
- Calcoliamo $\phi(n) = (p-1)(q-1) = 96$
- Selezioniamo e t.c. e sia un primo relativo di $\phi(n) = 96$ e minore di $\phi(n)$; in questo caso, $e= 5$
- Determiniamo d tale che $d*e = 1 \bmod 96$ e $d<96$. Il valore corretto è $d = 77$, poiché $77 \times 5 = 385 = 4 \times 96 + 1$

Esempio di uso RSA (2/2)



Forza di RSA

- Attacco a forza bruta: proviamo tutte le possibili chiavi – più grandi *e, d* più sicuro il sistema
- Più grande la chiave, più lento il sistema
- Per grandi *n* con grandi fattori primi, la fattorizzazione e' difficile
- Nel 1994 recuperata una chiave a 428 bit key; $100
- Oggigiorno una chiave a 1024 bit è considerata sicura (???!!!!)

RSA

- Il miglior algoritmo di fattorizzazione (Brent-Pollard) richiede:

$$O\left(\frac{e^{\sqrt{2 \ln p \ln \ln p}}}{\ln p}\right)$$

Dove *p* è il fattore più grande tra i due.

- Number Field Sieve ha complessità asintotica di:

$$O(e^{(\log n)^{1/3}(\log \log n)^{2/3}})$$

RSA: Valori parametri

- Dimensioni del modulo >= 1024 bits
- Selezioniamo i primi
  - *p* e *q* debbono avere all'incirca la stessa dimensione (in bit)
  - *p* – *q* dovrebbe essere ragionevolmente grande (sqrt() attack)
  - Primi "forti"
    - p – 1 ha un fattore primo r grande ⇐ Pollard p – 1 factoring
    - p + 1 ha un fattore primo r grande⇐ Pollard p + 1 factoring
    - r –1 ha un fattore primo grande⇐ cycling attacks
    - Scegliere casualmente p, q in genere dimostrano avere buone proprietà
- Selezioniamo e
  - In genere 3 oppure $2^{16}+1 = 65537$

RSA: complessità

- Performance (p, q sono primi di k-bit)
  - Firma $O(k^3)$
  - Verifica $O(k^2)$
- Banda
  - Ad esempio, ISO/IEC 9796 mappa un messaggio di k-bit su $2k$-bit per una firma richiesta di $2k$-bit (efficienza pari a $\frac{1}{2}$)

Sommario

- Gli algoritmi a chiave pubblica sono fondati su problemi che ad oggi sono difficili da risolvere e quindi considerati sicuri, ma….
  - DH, ElGamal → logaritmo discreto
  - RSA → fattorizzazione
- Chiavi di dimensioni maggiori che per gli algoritmi simmetrici
- Carico computazionale maggiore che per algoritmi simmetrici.