

Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica



Corso di Algoritmi e Strutture Dati

Algoritmi di teoria dei numeri



Dimensione del problema e costo computazionale

- Nelle applicazioni della teoria dei numeri vengono usati numeri interi molto grandi
- La dimensione dell'ingresso è tipicamente valutata in termini del numero di bit necessari per rappresentare i valori in ingresso
 - Anziché in termini del numero di valori in ingresso
 - Un algoritmo ha un tempo di esecuzione polinomiale se viene eseguito in un tempo $\lg a$ quando in ingresso c'è un valore a
- La complessità computazionale è misurata in termini del numero di operazioni su bit effettuate
 - Operazioni elementari su grandi numeri sono time-consuming
 - Moltiplicare due interi rappresentati su b bit richiede $\Theta(b^2)$ operazioni su bit
 - Anche se algoritmi più efficienti esistono

Nozioni di teoria dei numeri



- Insieme dei numeri interi (relativi) $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- Insieme dei numeri naturali $\mathbb{N} = \{0, 1, 2, \dots\}$
- $d | a$ “d divide a” se $a=kd$ per qualche intero k
 - a è un multiplo di d
 - Se $a>0$, allora $|d| \leq |a|$
 - Ogni intero divide 0
 - $d | a \Leftrightarrow -d | a$
 - Se $d\geq 0$, d è un **divisore** di a
 - Un divisore di un intero a non nullo è compreso tra 1 e $|a|$
 - Ogni intero a è sempre divisibile per 1 e a (divisori banali)
 - I divisori non banali di a sono detti **fattori** di a

Numeri primi e composti



- Un intero $a > 1$ i cui unici divisori sono quelli banali è detto primo
 $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, \dots$
- I numeri non primi sono detti composti
- L'intero 1 (unità) non è né primo né composto
- Analogamente, 0 e i numeri negativi non sono né primi né composti

Teorema della divisione



Teorema 1. Per ogni intero a e ogni intero positivo n , sono unici gli interi q e r tali che $0 \leq r < n$ e $a = qn + r$

- $q = \lfloor a/n \rfloor$ è il quoziente della divisione
- $r = a \bmod n$ è il resto della divisione
- $n \mid a$ se e solo se il resto è zero
- Gli interi possono essere divisi in n classi di equivalenza sulla base del loro resto modulo n
- $[a]_n = \{a + kn : k \in \mathbb{Z}\}$
 - $[3]_7 = \{\dots, -11, -4, 3, 10, 17, \dots\}$
- $a \equiv b \pmod{n}$ è analogo a $a \in [b]_n$
- Denotiamo l'insieme di tutte le classi di equivalenza modulo n con $\mathbb{Z}_n = \{[a]_n : 0 \leq a \leq n-1\}$

Massimo comune divisore



- Se $d \mid a$ e $d \mid b$, allora d è un divisore comune di a e b
- Il massimo comune divisore di due interi a e b è il più grande tra i divisori comuni di a e b
 - es. $\gcd(24,30) = 6$, $\gcd(5,7) = 1$
 - $\gcd(a,0)=a$
 - Per convenzione $\gcd(0,0) = 0$

Interi relativamente primi



- Due interi a e b sono detti **relativamente primi** se il loro unico comune divisore è 1
 - es. 8 e 15 sono relativamente primi

Teorema 6. Siano a , b e p tre interi tali che $\gcd(a,p)=1$ e $\gcd(b,p)=1$ allora $\gcd(ab,p) = 1$

- Gli interi n_1, n_2, \dots, n_k sono **relativamente primi a coppie** se $\gcd(n_i, n_j) = 1$ se $i \neq j$

Fattorizzazione



- *Teorema 7. Sia p un numero primo e a e b due interi. Se $p \mid ab$, allora $p \mid a$ oppure $p \mid b$ (o entrambe)*
- *Teorema 8. Un intero composto a può essere scritto in un solo modo come prodotto della forma $a = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$ dove p_i sono numeri primi ed e_i interi positivi*

Massimo comune divisore



- Il massimo comune divisore di due interi positivi a e b si potrebbe calcolare a partire dalla fattorizzazione di a e b
- Se $a = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$ e $b = p_1^{f_1} p_2^{f_2} \dots p_r^{f_r}$
allora $\gcd(a, b) = p_1^{\min(e_1, f_1)} p_2^{\min(e_2, f_2)} \dots p_r^{\min(e_r, f_r)}$
 - Esponenti nulli per numeri primi non presenti nella fattorizzazione
- Tuttavia, i migliori algoritmi per la fattorizzazione non hanno un tempo di esecuzione polinomiale

Massimo comune divisore



*Teorema 9. Per ogni intero non negativo a e ogni intero positivo b,
 $\gcd(a,b) = \gcd(b, a \bmod b)$*

- L'algoritmo di Euclide si basa su questo teorema

```
Euclid (a, b)
if b = 0
    then return a
else return Euclid (b, a mod b)
```

- es. $\text{Euclid}(30,21) = \text{Euclid}(21,9) = \text{Euclid}(9,3) = \text{Euclid}(3,0) = 3$
- Il secondo argomento diminuisce ad ogni invocazione
- Si dimostra che il numero di chiamate ricorsive è $O(\lg b)$ ($b < a$)
 - Se a e b sono rappresentati su β bit, $O(\beta)$ invocazioni
 - Se le operazioni richiedono $O(\beta^2)$, complessità $O(\beta^3)$

Algoritmo di Euclide esteso



- L'algoritmo di Euclide può essere esteso per restituire due interi x e y tali che $d = \gcd(a,b) = ax+by$
 - x e y possono essere nulli o negativi

Extended-Euclid (a,b)

```
if b = 0
    then return (a,1,0)
(d',x',y') ← Extended-Euclid (b, a mod b)
(d,x,y) ← (d',y',x'-[a/b]y')
return (d,x,y)
```

- Sia S un insieme e \oplus un'operazione binaria su S . (S, \oplus) è un **gruppo** se:
 - \oplus è un'operazione interna: per ogni $a, b \in S$, $a \oplus b \in S$
 - Esiste l'elemento neutro $e \in S$: $a \oplus e = e \oplus a = a$ per ogni $a \in S$
 - Vale la proprietà associativa: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ per ogni $a, b, c \in S$
 - Per ogni $a \in S$ esiste l'elemento inverso $b \in S$ tale che $a \oplus b = b \oplus a = e$
- Se vale anche la proprietà commutativa il gruppo è detto **abeliano**
 - Per ogni $a, b \in S$ $a \oplus b = b \oplus a$
- Il gruppo è finito se la cardinalità di S è finita

Aritmetica modulare



- Per definire un gruppo su Z_n occorre definire operazioni binarie su Z_n
- Se $a \equiv a' \pmod{n}$ e $b \equiv b' \pmod{n}$ allora
 $a+b \equiv a'+b' \pmod{n}$
 $ab \equiv a'b' \pmod{n}$
- Si definisce l'addizione modulo n: $[a]_n +_n [b]_n = [a+b]_n$
- Si definisce la moltiplicazione modulo n: $[a]_n \cdot_n [b]_n = [ab]_n$
- Si dimostra che $(Z_n, +_n)$ e (Z_n^*, \cdot_n) sono gruppi abeliani finiti dove Z_n^* è l'insieme degli elementi di Z_n relativamente primi a n
 $Z_n^* = \{[a]_n \in Z_n : \gcd(a,n)=1\}$

- La cardinalità di Z_n è n
- La cardinalità di Z_n^* è data dalla funzione phi di Eulero:

$$\phi(n) = n \prod_{p|n} (1 - 1/p)$$

dove la produttoria è estesa a tutti i numeri primi p che dividono n (incluso n, se n è primo)

- Se p è primo $Z_p^* = \{1, 2, \dots, p-1\}$ e $\phi(p) = p-1$
- L'elemento neutro di (Z_n^*, \cdot_n) è $[1]_n$, quindi l'inverso (moltiplicativo) di $[a]_n$ è l'elemento $[x]_n$ tale che $[a]_n \cdot_n [x]_n = [1]_n$
 - O, più sinteticamente, $ax \equiv 1 \pmod{n}$

Equazioni lineari in modulo

- Vogliamo trovare le soluzioni alle equazioni $ax \equiv b \pmod{n}$

Corollario 21. L'equazione $ax \equiv b \pmod{n}$ ammette soluzioni se e solo se $\gcd(a,n) | b$

Corollario 22. L'equazione $ax \equiv b \pmod{n}$ o ammette $d = \gcd(a,n)$ distinte soluzioni modulo n o non ammette soluzione

Modular-Linear-Equation-Solver (a, b, n)

$(d, x', y') \leftarrow \text{Extended-Euclid } (a, n)$

if $d | b$

 then $x_0 \leftarrow x' (b/d) \pmod{n}$

 for $i \leftarrow 0$ to $d-1$

 do print $(x_0 + i(n/d)) \pmod{n}$

 else print "no solutions"

Equazioni lineari in modulo



- Se $b=1$, la soluzione che cerchiamo è l'inverso moltiplicativo di a modulo n
Corollario 26. Per ogni $n > 1$, se $\gcd(a, n) = 1$, allora l'equazione $ax \equiv 1 \pmod{n}$ ammette un'unica soluzione modulo n
- L'inverso moltiplicativo di a è il valore x restituito da Extended-Euclid, dato che $\gcd(a, n) = 1 = ax + ny$ implica $ax \equiv 1 \pmod{n}$

Modular-Linear-Equation-Solver (a, b, n)

$(d, x', y') \leftarrow \text{Extended-Euclid } (a, n)$

if $d \mid b$

 then $x_0 \leftarrow x' (b/d) \pmod{n}$

 for $i \leftarrow 0$ to $d-1$

 do print $(x_0 + i(n/d)) \pmod{n}$

 else print “no solutions”

Il Teorema cinese del resto



Corollario 29. Se n_1, n_2, \dots, n_k sono relativamente primi a coppie e $n = n_1 n_2 \cdots n_k$, allora per tutti gli interi x e a

$$x \equiv a \pmod{n_i} \quad \text{per } i=1,2,\dots,k$$

se e solo se $x \equiv a \pmod{n}$

- Esempio: $n_1=2, n_2=15, n_3=7 \Rightarrow n=210$
 $10 \equiv 220 \pmod{210}$
 $10 \equiv 220 \pmod{2}$ infatti 10 e 220 appartengono a $[0]_2$
 $10 \equiv 220 \pmod{15}$ infatti 10 e 220 appartengono a $[10]_{15}$
 $10 \equiv 220 \pmod{7}$ infatti 10 e 220 appartengono a $[3]_7$

Potenza di un elemento



- Sull'insieme Z_n^* è definita la moltiplicazione modulo n
- Per estensione, si definisce la potenza di a modulo n, con $a \in Z_n^*$
- Vale il seguente teorema di Fermat

Se p è un numero primo e $a \in Z_p^$ allora $a^{p-1} \equiv 1 \pmod{p}$*

Potenza di un elemento

- Vediamo il metodo dei quadrati ripetuti per calcolare in modo efficiente $a^b \bmod n$
- Sia $b_k, b_{k-1}, \dots, b_1, b_0$ la rappresentazione binaria di b
- Es. $a=7, b=560, n=561$

Modular-Exponentiation (a, b, n)

```
c ← 0
d ← 1
for i ← k downto 0
    do c ← 2c
        d ← (d·d) mod n
        if  $b_i = 1$ 
            then c ← c + 1
        d ← (d·a) mod n
return d
```

b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Esponenziale discreto

- Invariante:
 - c vale $b_k, b_{k-1}, \dots, b_{i+1}$,
 - $d = a^c \bmod n$
- Se a, b, n sono rappresentati su β bit
 - $O(\beta)$ operazioni aritmetiche
 - $O(\beta^3)$ operazioni su bit

b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Modular-Exponentiation (a, b, n)
 $c \leftarrow 0$
 $d \leftarrow 1$
 for $i \leftarrow k$ downto 0
 do $c \leftarrow 2c$
 $d \leftarrow (d \cdot d) \bmod n$
 if $b_i = 1$
 then $c \leftarrow c + 1$
 $d \leftarrow (d \cdot a) \bmod n$
 return d

Crittografia a chiave pubblica



- Un sistema di crittografia a chiave pubblica può essere usato per cifrare i messaggi inviati tra due entità in modo da impedire la decodifica da parte di chi ascolta i messaggi
- Un sistema di crittografia a chiave pubblica consente inoltre di apporre una sorta di firma digitale nei messaggi inviati
 - Che autentica sia l'identità del mittente che il contenuto del messaggio inviato
- Il sistema di crittografia a chiave pubblica RSA si basa sulla enorme differenza di complessità tra trovare numeri primi molto grandi (semplice) e fattorizzare il prodotto di due numeri primi molto grandi (complicato)

Crittografia a chiave pubblica

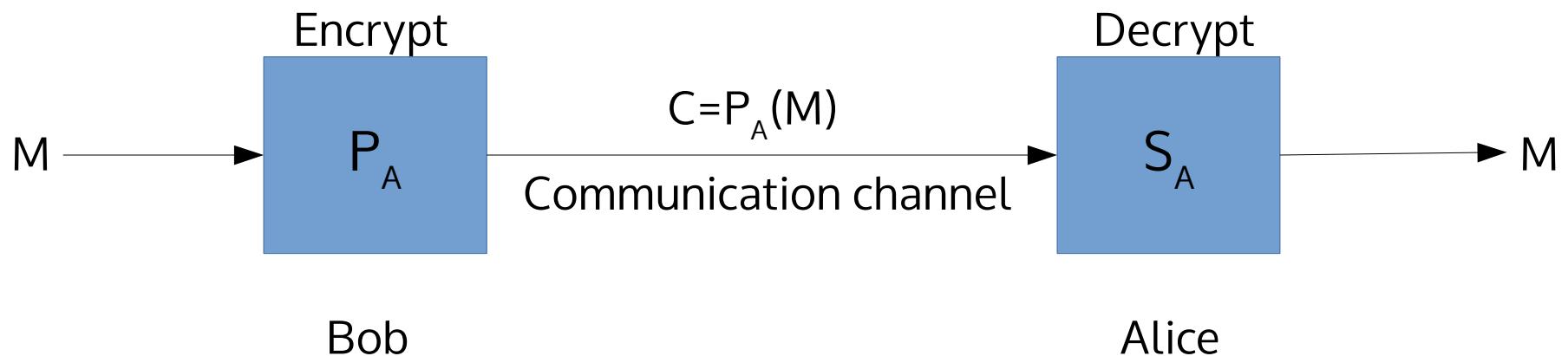


- Ogni partecipante ha sia una chiave pubblica che una chiave privata (segreta)
- Alice ha chiavi P_A e S_A , Bob ha P_B e S_B
- Le chiavi pubbliche e private sono usate come parametro di funzioni che si possono applicare a qualunque messaggio appartenente all'insieme dei messaggi possibili \mathcal{D}
- Assumiamo che il risultato sia ancora un valore in \mathcal{D}
- Le funzioni che corrispondono a P_A e S_A sono indicate con $P_A()$ e $S_A()$
 - Tali funzioni sono l'una l'inversa dell'altra:
 $M = P_A(S_A(M))$ e $M = S_A(P_A(M))$ per ogni messaggio $M \in \mathcal{D}$

- È essenziale che solo Alice sia in grado di calcolare $S_A()$ in un tempo ragionevole
 - Da questo dipendono la segretezza e l'autenticità del messaggio
 - È il motivo per cui la chiave privata va mantenuta segreta
 - Deve valere anche se sono note P_A e $P_A()$
- La maggiore difficoltà sta nel progettare un sistema di crittografia dove è difficile calcolare la funzione inversa di una funzione ($P_A()$) che viene rivelata

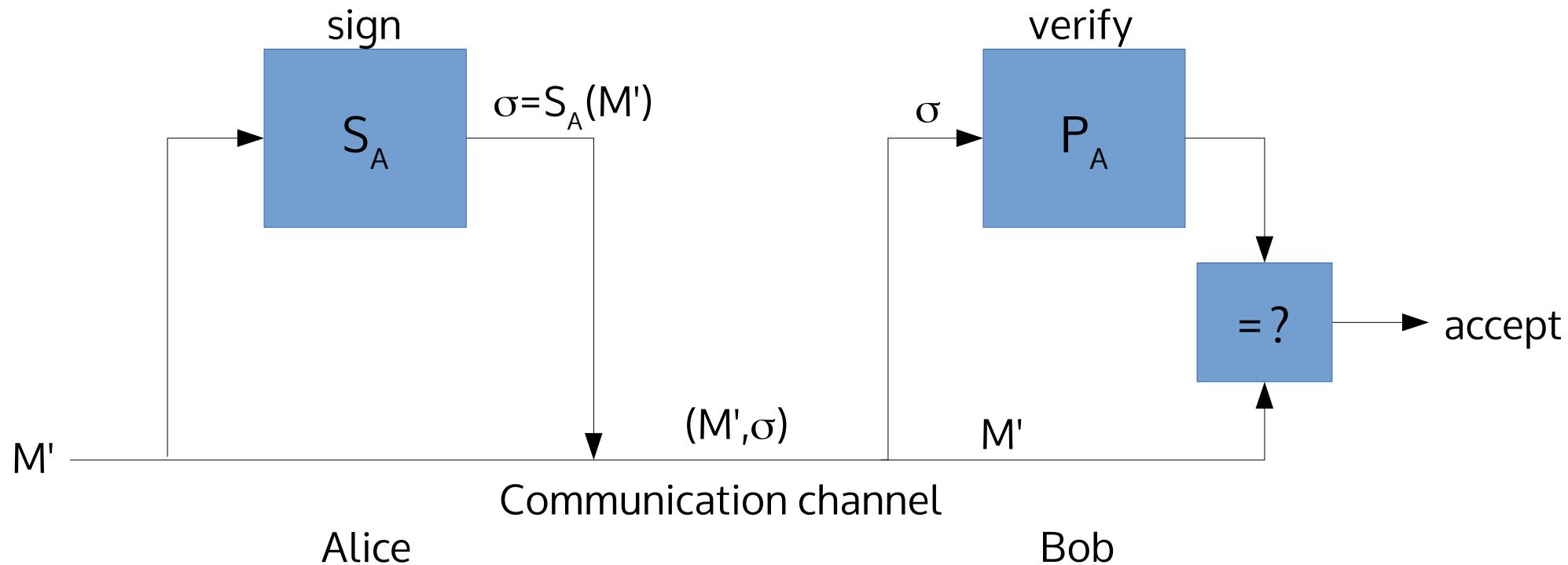
Crittografia a chiave pubblica

- Per garantire la **segretezza** del messaggio che invia, Bob cifra il messaggio usando la chiave pubblica di Alice
- Alice usa la propria chiave privata per decifrare il messaggio



Crittografia a chiave pubblica

- Per garantire l'autenticità del mittente e del messaggio, Alice calcola la firma digitale σ usando la propria chiave segreta
- Alice invia il messaggio (in chiaro) e la firma digitale
- Bob verifica che il messaggio ricevuto coincida con il risultato dell'applicazione di $P_A()$ su σ



- Si possono ottenere segretezza e autenticità combinando i due protocolli visti
- Il mittente
 - Calcola la firma digitale (con la propria chiave segreta) e la appende al messaggio
 - Cifra il messaggio e la firma (con la chiave pubblica del destinatario)
- Il ricevente
 - Decifra messaggio e firma del mittente (con la propria chiave privata)
 - Verifica la firma del mittente (con la chiave pubblica del mittente)

Il sistema di crittografia RSA



- Il sistema di crittografia a chiave pubblica RSA (Rivest, Shamir, Adleman) specifica come calcolare una coppia di chiavi pubblica e privata e le funzioni associate
- La coppia di chiavi si determina in questo modo:
 - Si scelgono due numeri primi distinti p e q molto grandi (~kbits)
 - Si calcola il prodotto $n = p \cdot q$
 - Si sceglie un (piccolo) intero dispari e che è relativamente primo con $\phi(n) = (p - 1)(q - 1)$
 - Si calcola d come l'inverso moltiplicativo di e modulo $\phi(n)$
 - Il Corollario 26 garantisce l'esistenza e unicità di d , che si può calcolare mediante Extended-Euclid
 - La **chiave pubblica** è la coppia $P = (e, n)$
 - La **chiave privata** è la coppia $S = (d, n)$

Il sistema di crittografia RSA



- L'insieme dei possibili messaggi \mathcal{D} è l'insieme Z_n
- La funzione associata alla chiave pubblica è $P(M)=M^e \pmod{n}$
- La funzione associata alla chiave privata è $S(M)=M^d \pmod{n}$
- Queste funzioni possono essere calcolate con la procedura Modular-Exponentiation
- Per calcolo complessità assumiamo $\lg e=O(1)$, $\lg d \leq \beta$, $\lg n \leq \beta$
 - Per applicare la chiave pubblica
 $O(1)$ moltiplicazioni in modulo $\Rightarrow O(\beta^2)$ operazioni su bit
 - Per applicare la chiave privata
 $O(\beta)$ moltiplicazioni in modulo $\Rightarrow O(\beta^3)$ operazioni su bit

Il sistema di crittografia RSA



- Proviamo la correttezza di RSA: $P(S(M))=S(P(M))=M$
 - Dalla definizione e proprietà delle potenze di potenze:
 $P(S(M)) = S(P(M)) = M^{ed} \pmod{n}$
 - Dato che e è l'inverso di d modulo $\phi(n)=(p-1)(q-1)$:
 $ed = 1 + k(p-1)(q-1)$ per un qualche k
 - Da cui, se $M \not\equiv 0 \pmod{p}$:
$$\begin{aligned} M^{ed} &\equiv M \cdot M^{k(p-1)(q-1)} \pmod{p} \\ &\equiv M(M^{p-1})^{k(q-1)} \pmod{p} \quad (\text{potenza di potenza}) \\ &\equiv M(1)^{k(q-1)} \pmod{p} \quad (\text{Teorema di Fermat, usa ipotesi}) \\ &\equiv M \pmod{p} \end{aligned}$$
 - Se $M \equiv 0 \pmod{p}$, allora $M^{ed} \equiv M \pmod{p}$
 - In ogni caso, dunque, $M^{ed} \equiv M \pmod{p}$

Il sistema di crittografia RSA



- $M^{ed} \equiv M \pmod{p}$
- $M^{ed} \equiv M \pmod{q}$ in modo analogo
- $M^{ed} \equiv M \pmod{n}$ per il corollario 29 (dato che $n=pq$)

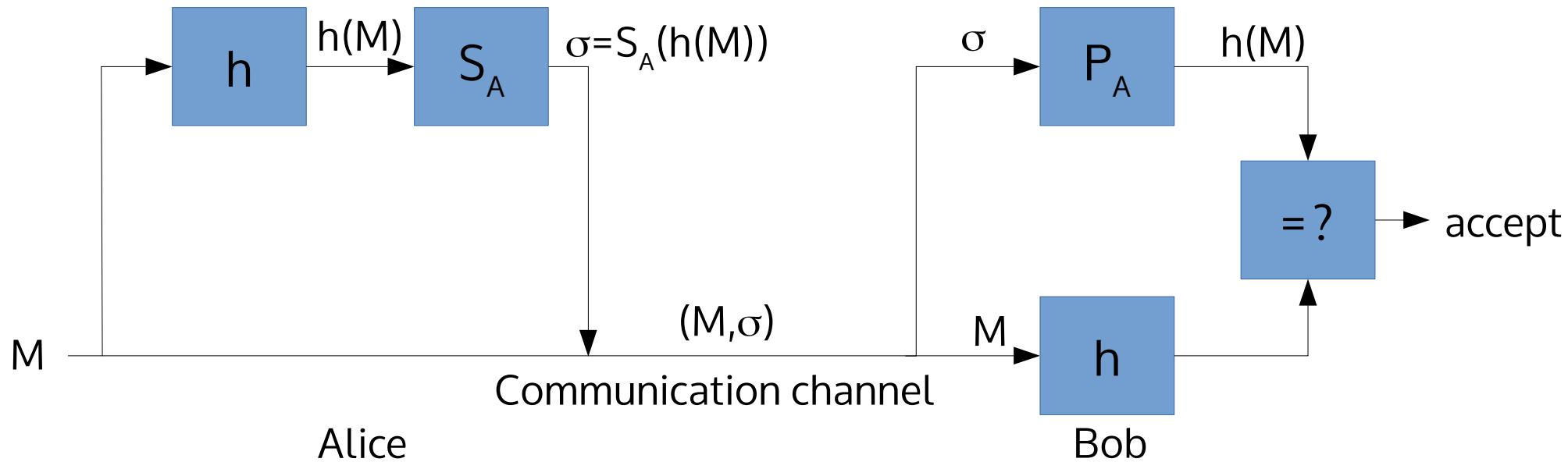
Il sistema di crittografia RSA



- La sicurezza di RSA dipende in gran parte dalla difficoltà di fattorizzare interi molto grandi
 - Se un attaccante riesce a fattorizzare n ottiene p e q e dalla conoscenza della chiave pubblica (e) si calcola la chiave privata
 - Individuare la fattorizzazione di interi molto grandi è un problema estremamente difficoltoso
 - Calcolare numeri primi molto grandi è invece semplice

Il sistema di crittografia RSA

- Avere una firma digitale della stessa lunghezza del messaggio non è agevole
- Tipicamente, si usa una funzione hash *collision-resistant*
 - Facile da calcolare
 - È difficile determinare un M' tale che $h(M)=h(M')$
 - Produce una stringa di bit di lunghezza fissata



Il sistema di crittografia RSA



- Questo sistema garantisce l'autenticità di $h(M)$, *non* di M
- Tuttavia, per le proprietà della funzione hash utilizzata, è improbabile che si riesca ad alterare il messaggio originale M ottenendo un messaggio M' tale che $h(M)=h(M')$

- Per distribuire le chiavi pubbliche dei vari soggetti si può utilizzare una “trusted authority” T
- La chiave pubblica di T è nota a tutti
- Alice può ottenere da T un certificato che dice che “La chiave pubblica di Alice is P_A ”
 - Tale certificato è firmato con la chiave privata di T
- Alice può includere tale certificato nei messaggi che firma
 - Il ricevente può verificare il certificato usando la chiave pubblica di T
 - Dal certificato, il ricevente ricava la chiave pubblica di Alice, con la quale verifica il messaggio