Concludere che la trasformazione SubBytes non è una funzione affine (cioé una funzione della forma $\alpha x + \beta$) da $GF(2^8)$ a $GF(2^8)$ (*suggerimento:* si veda l'Esercizio 3(a)).

Capitolo 6

Algoritmo RSA

6.1 Algoritmo RSA

Vedi appunti MD

Alice vuole inviare un messaggio a Bob. Il problema è che non hanno mai avuto alcun contatto in precedenza e che non vogliono aspettare che un corriere consegni una chiave. In questo modo, tutte le informazioni che Alice invia a Bob possono essere carpite da un osservatore malintenzionato come Eva. In realtà, è ancora possibile inviare un messaggio in modo che possa essere letto da Bob, ma non da Eva, anche se una cosa del genere sarebbe impossibile con tutti i metodi discussi in precedenza. Alice dovrebbe inviare una chiave, che Eva potrebbe intercettare e usare per decifrare ogni ulteriore messaggio. Uno schema di questo tipo, chiamato **crittosistema a chiave pubblica**, fu inizialmente proposto da Diffie e Hellman nel loro classico articolo [Diffie-Hellman]. Tuttavia, non avevano ancora un'implementazione pratica (anche se presentarono una procedura alternativa di scambio della chiave che funzionava su canali pubblici; si veda il Paragrafo 7.4). Negli anni che seguirono, vennero proposti vari metodi. Quello che ha avuto maggior successo, ossia l'algoritmo RSA, si basa sulla difficoltà di fattorizzare gli interi in fattori primi e fu proposto da Rivest, Shamir e Adleman nel 1977.
Si è a lungo affermato che le agenzie crittografiche governative avessero scoperto l'algoritmo RSA parecchi anni prima, anche se, per motivi di sicurezza, non ne avevano lasciato traccia alcuna. Infine, nel 1997 il CESG, un'agenzia crittografica inglese, rilasciò dei documenti che mostravano che nel 1970 James Ellis aveva scoperto la crittografia a chiave pubblica e che nel 1973 Clifford Cocks aveva scritto un documento interno in cui veniva descritta una versione dell'algoritmo RSA in cui l'esponente di cifratura e (si veda la discussione che segue) era uguale al modulo n.
Ecco come funziona l'algoritmo RSA. Bob sceglie due primi p e q grandi e distinti e li moltiplica per formare il numero

$$n = pq.$$

Sceglie anche un esponente di cifratura e tale che

$$\text{MCD}(e,(p-1)(q-1))=1.$$

Invia ad Alice la coppia (n,e) ma tiene segreti i valori di p e q. In particolare, Alice, che potrebbe anche essere un nemico di Bob, non ha mai bisogno di conoscere p e q per inviare un messaggio a Bob in modo sicuro. Alice scrive il messaggio come un numero m. Se m è più grande di n, spezza il messaggio in blocchi, ognuno minore di n. Per il momento, per semplicità, supponiamo che $m<n$. Alice calcola

$$c\equiv m^e \pmod n$$

e invia c a Bob. Conoscendo p e q, Bob può calcolare $(p-1)(q-1)$ e quindi può trovare l'esponente di decifrazione d con

$$de\equiv 1 \pmod{(p-1)(q-1)}.$$

Poiché, come vedremo più avanti,

$$m\equiv c^d \pmod n,$$

Bob può leggere il messaggio.
L'algoritmo è sintetizzato nella seguente tabella.

Algoritmo RSA

1. Bob sceglie due primi p e q segreti e calcola $n=pq$.
2. Bob sceglie e con $\text{MCD}(e,(p-1)(q-1))=1$.
3. Bob calcola d in modo che $de\equiv 1 \pmod{(p-1)(q-1)}$.
4. Bob rende pubblici n ed e e tiene segreti p, q e d.
5. Alice cifra m come $c\equiv m^e \pmod n$ e invia c a Bob.
6. Bob decifra calcolando $m\equiv c^d \pmod n$.

Esempio. Bob sceglie i primi $p=885320963$ e $q=238855417$, in modo che

$$n=p\cdot q=211463707796206571.$$

Scelto $e=9007$ come esponente di cifratura, Bob manda ad Alice i valori di n ed e. Il messaggio di Alice è *cat*. Abbandoniamo la precedente convenzione di numerare le lettere a partire da $a=0$ e conveniamo di iniziare da $a=01$ fino ad arrivare a $z=26$. Con il metodo precedente la lettera a sparirebbe ogni volta che apparisse all'inizio di un messaggio, poiché si otterrebbe un numero m che inizia con 00. Con questa nuova convenzione, il messaggio di Alice è

$$m=30120.$$

Alice calcola

$$c\equiv m^e\equiv 30120^{9007}\equiv 113535859035722866 \pmod n$$

e invia c a Bob.
Poiché Bob conosce p e q, conosce anche $(p-1)(q-1)$. Mediante l'algoritmo euclideo esteso (si veda il Paragrafo 3.2) calcola d in modo che

$$de\equiv 1 \pmod{(p-1)(q-1)},$$

ottenendo

$$d=116402471153538991.$$

Infine, Bob calcola

$$c^d\equiv 113535859035722866^{116402471153538991}\equiv 30120 \pmod n,$$

ottenendo così il messaggio originale. ■

- Ci sono vari elementi da spiegare, ma forse il più importante è perché $m\equiv c^d \pmod n$. Per il teorema di Eulero (Paragrafo 3.6) se $\text{MCD}(a,n)=1$, allora $a^{\varphi(n)}\equiv 1 \pmod n$. Nel nostro caso, $\varphi(n)=\varphi(pq)=(p-1)(q-1)$. Si supponga che $\text{MCD}(m,n)=1$. Questo è molto probabile che sia vero. Infatti, poiché p e q sono grandi, probabilmente m non possiede nessuno dei due come fattore. Poiché $de\equiv 1 \pmod{\varphi(n)}$, si può scrivere $de=1+k\varphi(n)$ per un opportuno intero k. Quindi

$$c^d\equiv (m^e)^d\equiv m^{1+k\varphi(n)}\equiv m\cdot(m^{\varphi(n)})^k\equiv m\cdot 1^k\equiv m \pmod n.$$

Si è così mostrato che Bob può recuperare il messaggio. Se $\text{MCD}(m,n)\neq 1$, Bob può ancora recuperare il messaggio (si veda l'Esercizio 19).

- Cosa fa Eva? Intercetta n, e, c, ma non conosce p, q, d. Supponiamo che Eva non abbia la possibilità di fattorizzare n. Il modo ovvio di calcolare d richiede la conoscenza di $\varphi(n)$. Mostreremo in seguito che questo è equivalente alla conoscenza di p e q. Esiste un altro modo? Mostreremo che se Eva può trovare d, allora probabilmente può fattorizzare n. Quindi è poco probabile che Eva riesca a trovare l'esponente di decifrazione d.
- Poiché Eva conosce $c\equiv m^e \pmod n$, perché non prende semplicemente la radice e-esima di c? Questo funziona bene se non si sta lavorando modulo n, ma è molto difficoltoso nel nostro caso. Per esempio, se si sa che $m^3\equiv 3 \pmod{85}$, non si può calcolare la radice cubica di 3, ossia 1,4422..., con la propria calcolatrice e poi ridurre modulo 85. Naturalmente, una ricerca caso per caso alla fine porterebbe a trovare $m=7$, ma questo metodo non è utilizzabile quando n è grande.
- In che modo Bob sceglie p e q? Dovrebbero essere scelti a caso, indipendentemente l'uno dall'altro. Quanto grandi dipende dal livello di sicurezza richiesto, anche se

sembra che dovrebbero avere almeno 100 cifre. Per ragioni che discuteremo più avanti, forse è meglio sceglierli di lunghezze leggermente diverse. Quando discuteremo i test di primalità, vedremo che primi di questo tipo possono essere trovati abbastanza rapidamente. Si dovrebbero fare alcuni altri test su p e q per essere sicuri che vadano bene. Per esempio, se $p-1$ ha solo fattori primi piccoli, allora n si fattorizza facilmente mediante il metodo $p-1$ (si veda il Paragrafo 6.4) e quindi p dovrebbe essere scartato e sostituito con un altro primo.

- Perché Bob richiede che $\text{MCD}(e,(p-1)(q-1))=1$? Si tenga presente (dal Paragrafo 3.3) che $de \equiv 1 \pmod{(p-1)(q-1)}$ possiede una soluzione d se e solo se $\text{MCD}(e,(p-1)(q-1))=1$. Quindi questa condizione è necessaria per l'esistenza di d. L'algoritmo euclideo esteso può essere usato per calcolare d rapidamente. Poiché $p-1$ è pari, $e=2$ non può essere usato. Si potrebbe allora essere tentati di usare $e=3$. Tuttavia, è pericoloso usare valori piccoli di e (si vedano il Paragrafo 6.2, il Problema al calcolatore 14 e il Paragrafo 17.3). Per questo motivo di solito si raccomanda di usare un esponente più grande. Per esempio, si potrebbe scegliere e come un primo moderatamente grande. In questo caso non ci sono difficoltà ad assicurare che $\text{MCD}(e,(p-1)(q-1))=1$.
- Nel processo di cifratura, Alice calcola $m^e \pmod n$. Si tenga presente che questo calcolo può essere svolto abbastanza rapidamente e senza molta memoria, per esempio mediante quadrature successive. Questo è decisamente uno dei vantaggi dell'aritmetica modulare: se Alice tentasse di calcolare prima m^e e di ridurlo poi modulo n, la memorizzazione di m^e potrebbe portare il suo computer a un overflow di memoria. Analogamente, anche il processo di decifrazione dato dal calcolo di $c^d \pmod n$ può essere svolto efficientemente. Quindi tutte le operazioni necessarie per la cifratura e la decifrazione possono essere svolte rapidamente (cioè in un tempo pari a una potenza di $\log n$). La sicurezza deriva dall'ipotesi che n non possa essere fattorizzato.

Le due affermazioni fatte in precedenza verranno ora giustificate. Il punto di queste due affermazioni era che trovare $\varphi(n)$ o trovare l'esponente di decifrazione d è essenzialmente difficile quanto fattorizzare n. Quindi, se fattorizzare è difficile, allora non dovrebbero esistere modi veloci e ingegnosi per trovare d.

Affermazione 1. *Sia $n=pq$ il prodotto di due primi distinti. Se n e $\varphi(n)$ sono noti, allora p e q possono essere calcolati rapidamente.*

Dimostrazione. Poiché $n-\varphi(n)+1 = pq-(p-1)(q-1)+1 = p+q$, si conoscono pq e $p+q$. Pertanto si può considerare il polinomio

$$X^2-(n-\varphi(n)+1)X+n = X^2-(p+q)X+pq = (X-p)(X-q)$$

che ammette p e q come radici. A questo punto p e q possono essere calcolati mediante la formula

$$p,q = \frac{n-\varphi(n)+1 \pm \sqrt{(n-\varphi(n)+1)^2-4n}}{2}$$

che dà le radici del polinomio precedente. □

Per esempio, se $n=221$ e $\varphi(n)=192$, si può considerare l'equazione

$$X^2-30X+221=0$$

che ammette le radici

$$p,q = \frac{30 \pm \sqrt{30^2-4\cdot 221}}{2} = 13, 17.$$

Affermazione 2. *Se d ed e sono noti, allora n può essere probabilmente fattorizzato.*

Dimostrazione. Nella discussione sui metodi di fattorizzazione del Paragrafo 6.4, si mostra che se si ha un esponente universale $b>0$ tale che $a^b \equiv 1 \pmod n$ per ogni a con $\text{MCD}(a,n)=1$, allora si può probabilmente fattorizzare n. Poiché $de-1$ è un multiplo di $\varphi(n)$, diciamo $de-1=k\varphi(n)$, si ha

$$a^{de-1} \equiv (a^{\varphi(n)})^k \equiv 1 \pmod n$$

ogniqualvolta $\text{MCD}(a,n)=1$. A questo punto basta applicare il metodo dell'esponente universale. □

Supponiamo, per esempio, che un insieme di banche voglia essere in grado di scambiare internamente dati finanziari. Se sono coinvolte varie centinaia di banche, allora non è realistico che ogni singola coppia di banche condivida una chiave per le loro comunicazioni segrete. In questa situazione si può usare l'algoritmo RSA. Ogni banca sceglie due interi n ed e, come prima. Questi numeri vengono poi resi pubblici. Se la banca A vuole inviare dei dati alla banca B, allora prende i numeri n ed e di B e li usa per inviare il messaggio. In pratica, l'algoritmo RSA non è sufficientemente veloce per inviare grandi quantità di dati. Di conseguenza, spesso l'algoritmo RSA viene usato per inviare una chiave per un metodo di cifratura più veloce, come DES.

- PGP (*Pretty Good Privacy*) è un metodo comune per cifrare le e-mail. Quando Alice invia una e-mail a Bob, prima firma il messaggio usando un algoritmo di firma digitale come quelli discussi nel Capitolo 9, poi cifra il messaggio usando un cifrario a blocchi come Triplo DES (o come IDEA o CAST-128) con una chiave di 128 bit scelta a caso (per ogni trasmissione si sceglie una nuova chiave casuale), infine cifra questa chiave usando la chiave pubblica RSA di Bob (si possono usare anche altri metodi a chiave pubblica). Quando Bob riceve l'e-mail, usa il proprio esponente RSA privato per decifrare la chiave casuale, poi usa questa chiave casuale per decifrare il messaggio e infine controlla la firma per vedere se il messaggio proviene effettivamente da Alice. Per un'ulteriore discussione su PGP, si veda il Paragrafo 10.6.

6.2 Attacchi a RSA

L'algoritmo RSA risulta essere utilizzabile in pratica solo se viene implementato correttamente. Negli Esercizi verranno mostrati alcuni possibili errori di implementazione. Qui presentiamo alcune altre potenziali difficoltà. Per maggiori informazioni sugli attacchi a RSA, si veda [Boneh].

Teorema. *Sia m il numero delle cifre di $n=pq$. Se si conoscono le prime $m/4$, o le ultime $m/4$, cifre di p, allora si può fattorizzare n in modo efficiente.*

In altre parole, se p e q hanno 100 cifre e se si conoscono le prime (o le ultime) 50 cifre di p, allora si può fattorizzare n. Quindi, se si sceglie un punto di partenza casuale per scegliere il nostro primo p, il metodo dovrebbe essere tale che gran parte di p non sia predicibile. Per esempio, si supponga di prendere a caso un numero N di 50 cifre e di testare la primalità dei numeri della forma $N \cdot 10^{50} + k$, per $k = 1,3,5,\ldots$, finché non si trova un primo p (cosa che dovrebbe accadere per $k < 1000$). Un attaccante che sa che si usa questo metodo conosce 47 delle ultime 50 cifre (saranno tutte 0 tranne le ultime 3 cifre). Utilizzando il metodo del teorema per $k < 1000$, alla fine si arriva alla fattorizzazione di n.
Per i dettagli del risultato precedente, si veda [Coppersmith2]. Un risultato analogo è il seguente.

Teorema. *Sia (n,e) una chiave pubblica RSA, sia m il numero delle cifre di n e sia d l'esponente di decifrazione. Se si hanno almeno le ultime $m/4$ cifre di d, allora si può trovare d in modo efficiente in un tempo che è lineare in $e \log_2 e$.*

Questo significa che il tempo per trovare d è limitato come una funzione lineare in $e \log_2 e$. Se e è piccolo, allora è piuttosto veloce trovare d quando si conosce una parte consistente di d. Se e è grande, per esempio circa n, il teorema non è migliore di una ricerca di d caso per caso. Per i dettagli, si veda [Boneh et al.].

6.2.1 Attacchi con esponenti bassi

Gli esponenti di cifratura o di decifrazione bassi sono attraenti perché accelerano la cifratura o la decifrazione. Tuttavia, ci sono alcuni pericoli, che devono essere evitati. Una trappola nella quale si può cadere usando $e = 3$ è presentata nel Problema al calcolatore 14. Un'altra difficoltà è discussa nel Capitolo 17 (Metodi basati su reticoli). Questi problemi possono essere evitati usando un esponente molto più grande. Una scelta comune è $e = 65537 = 2^{16} + 1$. Questo numero è primo e quindi è probabile che sia relativamente primo con $(p-1)(q-1)$. Poiché è più grande di 1 di una potenza di 2, l'elevamento a potenza per questo numero può essere eseguito rapidamente: per calcolare x^{65537}, basta elevare al quadrato x sedici volte e poi moltiplicare il risultato per x.
L'esponente di decifrazione d dovrebbe ovviamente essere scelto sufficientemente grande, in modo che sia impossibile trovarlo con la sola forza bruta. Tuttavia, bisogna stare ancora più attenti, come si vede dal seguente risultato. Un modo per ottenere le proprietà desiderate di d è di scegliere d primo e trovare poi e tale che $de \equiv 1 \pmod{\varphi(n)}$.
Supponiamo che Bob voglia essere in grado di decifrare i messaggi rapidamente e che pertanto scelga un valore piccolo di d. Il seguente teorema di M. Wiener [Wiener] dimostra che molto spesso Eva potrà trovare d con grande facilità. In pratica, se le disuguaglianze nelle ipotesi della proposizione sono indebolite, allora Eva può ancora usare il metodo per ottenere d in molti casi. Per questo motivo si consiglia di scegliere d piuttosto grande.

Teorema. *Siano p e q due primi con $q < p < 2q$. Sia $n = pq$ e siano $1 \le d,e < \varphi(n)$ tali che $de \equiv 1 \pmod{(p-1)(q-1)}$. Se $d < \frac{1}{3}n^{1/4}$, allora d può essere calcolato rapidamente (cioé in un tempo polinomiale in $\log n$).*

Dimostrazione. Poiché $q^2 < pq = n$, si ha $q < \sqrt{n}$. Quindi, poiché $p < 2q$,

$$n - \varphi(n) = pq - (p-1)(q-1) = p + q - 1 < 3q < 3\sqrt{n}.$$

Sia $ed = 1 + \varphi(n)k$ per un intero $k \ge 1$. Poiché $e < \varphi(n)$, si ha

$$\varphi(n)k < ed < \frac{1}{3}\varphi(n)n^{1/4},$$

e così $k < \frac{1}{3}n^{1/4}$. Quindi

$$kn - ed = k(n - \varphi(n)) - 1 < k(n - \varphi(n)) < \frac{1}{3}n^{1/4}(3\sqrt{n}) = n^{3/4}.$$

Inoltre, poiché $k(n - \varphi(n)) - 1 > 0$, si ha $kn - ed > 0$. Dividendo per dn si ottiene

$$0 < \frac{k}{d} - \frac{e}{n} < \frac{1}{dn^{1/4}} < \frac{1}{3d^2},$$

poiché, per ipotesi, $3d < n^{1/4}$.
Ora abbiamo bisogno di un risultato sulle frazioni continue. Si tenga presente che se x è un numero reale positivo e k e d sono interi positivi tali che

$$\left|\frac{k}{d} - x\right| < \frac{1}{2d^2},$$

allora k/d nasce dallo sviluppo in frazione continua di x (si veda il Paragrafo 3.12). Quindi, nel nostro caso, k/d nasce dallo sviluppo in frazione continua di e/n. Allora Eva esegue le seguenti operazioni.

1. Calcola la frazione continua di e/n. Dopo ogni passo, ottiene una frazione A/B.
2. Usa $k = A$ e $d = B$ per calcolare $C = (ed-1)/k$. (Poiché $ed = 1 + \varphi(n)k$, questo valore di C è un candidato per $\varphi(n)$.)
3. Se C non è un intero, procede al passo successivo della frazione continua.
4. Se C è un intero, allora trova le radici r_1 e r_2 di $X^2 - (n - C + 1)X + n$ (questo può essere il polinomio $X^2 - (n - \varphi(n) + 1)X + n = (X - p)(X - q)$ di prima). Se r_1 e r_2 sono interi, allora Eva ha fattorizzato n. Altrimenti, Eva procede al passo successivo dell'algoritmo della frazione continua.

Poiché il numero di passi nello sviluppo in frazione continua di e/n è al più una costante per $\log n$ e poiché l'algoritmo della frazione continua si ferma quando si ottiene la frazione e/n, l'algoritmo termina rapidamente. Quindi Eva trova rapidamente la fattorizzazione di n. □

Osservazione. Si tenga presente che le approssimazioni razionali di un numero x che provengono dall'algoritmo delle frazioni continue sono alternativamente più grandi di x e più piccole di x. Poiché $0 < \frac{k}{d} - \frac{e}{n}$, delle frazioni che nascono dalla fattorizzazione se ne deve considerare una sì e una no.
Cosa accade se Eva raggiunge e/n senza trovare la fattorizzazione di n? Questo significa che le ipotesi della proposizione non sono soddisfatte. Tuttavia, talvolta, anche quando le ipotesi non sono soddisfatte, è possibile che questo metodo produca la fattorizzazione di n.

Esempio. Sia $n = 1966981193543797$ e sia $e = 323815174542919$. La frazione continua di e/n è

$$[0;6,13,2,3,1,3,1,9,1,36,5,2,1,6,1,43,13,1,10,11,2,1,9,5]$$

$$= \cfrac{1}{6+\cfrac{1}{13+\cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\cdots}}}}}}}.$$

Poiché la prima frazione è $1/6$, si prova con $k = 1$ e $d = 6$. Tuttavia, poiché d deve essere dispari, questa possibilità può essere scartata.
Per l'osservazione, si può saltare alla terza frazione:

$$\frac{1}{6+\frac{1}{13+\frac{1}{2}}} = \frac{27}{164}.$$

Anche questa possibilità può scartata, sempre perché d deve essere dispari.
La quinta frazione è $121/735$. Questa dà $C = (e \cdot 735 - 1)/121$, che non è un intero.
La settima frazione è $578/3511$. Questa dà $C = 1966981103495136$ come candidato per $\varphi(n)$. Le radici di

$$X^2 - (n - C + 1)X + n$$

sono 37264873 e 52783789, con una precisione di parecchie cifre decimali. Poiché

$$n = 37264873 \cdot 52783789,$$

si è ottenuta la fattorizzazione di n.

6.2.2 Testo in chiaro corto

Un uso comune di RSA è quello di trasmettere chiavi da usare in DES o AES. Tuttavia, un'implementazione ingenua potrebbe portare a una perdita di sicurezza. Si supponga che una chiave DES di 56 bit sia scritta come un numero $m \approx 10^{17}$. Questo viene cifrato con RSA ottenendo $c \equiv m^e \pmod{n}$. Anche se m è piccolo, il testo cifrato c è probabilmente un numero della stessa grandezza di n, per esempio di circa 200 cifre. Tuttavia, Eva attacca il sistema nel modo seguente. Genera due liste:

1. $cx^{-e} \pmod{n}$ per ogni x con $1 \le x \le 10^9$.
2. $y^e \pmod{n}$ per ogni y con $1 \le y \le 10^9$.

Poi cerca una corrispondenza tra un elemento della prima lista e un elemento della seconda lista. Se ne trova una, allora ha $cx^{-e} \equiv y^e$ per qualche x e y. Pertanto

$$c \equiv (xy)^e \pmod{n}$$

e quindi $m \equiv xy \pmod{n}$. Questo attacco ha probabilità di avere successo? Se m è il prodotto di due interi x e y, entrambi minori di 10^9, allora x e y daranno a Eva una corrispondenza. Non tutti gli m avranno questa proprietà, anche se molti valori di m sono il prodotto di due interi minori di 10^9. Per questi numeri, Eva otterrà m.

Questo attacco è molto più efficiente del provare tutte le 10^{17} possibilità per m, un compito quasi impossibile su un computer e che richiederebbe un tempo molto lungo anche con molte migliaia di computer operanti in parallelo. In questo attacco, Eva deve calcolare e conservare gli elementi di una lista di lunghezza 10^9, poi deve calcolare gli elementi dell'altra lista e confrontare ognuno di essi con la prima lista. Quindi Eva esegue circa $2 \cdot 10^9$ calcoli (e confronti con la lista fino a 10^9 volte). Questo lavoro può essere svolto facilmente con un singolo computer. Per saperne di più su questo attacco, si veda [Boneh-Joux-Nguyen].
È facile prevenire questo attacco. Invece di usare un valore piccolo di m, si aggiunge a caso qualche cifra all'inizio e alla fine di m in modo da formare un testo in chiaro molto più lungo. Quando Bob decifra il testo cifrato, semplicemente rimuove queste cifre casuali e ottiene m.
Nel 1994 Bellare e Rogaway [Bellare-Rogaway2] introdussero un metodo più sofisticato di elaborare preliminarmente il testo in chiaro, ossia l'Optimal Asymmetric Encryption Padding (OAEP).
Si supponga che Alice voglia inviare un messaggio m a Bob, la cui chiave pubblica RSA è (n,e), dove n è formato da k bit. Si fissano due interi positivi k_0 e k_1 con $k_0 + k_1 < k$. Il messaggio di Alice può essere formato da $k - k_0 - k_1$ bit. I valori tipici sono $k = 1024$, $k_0 = k_1 = 128$, $k - k_0 - k_1 = 768$. Sia G una funzione che prende in input stringhe di k_0 bit e restituisce in output stringhe di $k - k_0$ bit. Sia H una funzione che prende in input $k - k_0$ bit e restituisce in output k_0 bit. Le funzioni G e H sono solitamente costruite mediante funzioni hash (per una discussione sulle funzioni hash si veda il Capitolo 8). Quando Alice deve cifrare m, innanzitutto lo allunga a $k - k_0$ bit aggiungendo k_1 bit uguali a zero, ottenendo $m0^{k_1}$. Poi sceglie a caso una stringa r di k_0 bit e calcola

$$x_1 = m0^{k_1} \oplus G(r), \qquad x_2 = r \oplus H(x_1).$$

Se la concatenazione $x_1||x_2$ è un numero binario più grande di n, Alice sceglie un nuovo numero casuale r e calcola di nuovo x_1 e x_2. Non appena ottiene $x_1||x_2 < n$ (questo accade con probabilità almeno $1/2$ per ogni r, se $G(r)$ produce output abbastanza casuali), forma il testo cifrato

$$E(m) = (x_1||x_2)^e \pmod{n}.$$

Per decifrare un testo cifrato c, Bob usa il proprio esponente d di decifrazione RSA privato per calcolare $c^d \pmod{n}$. Il risultato viene scritto nella forma

$$c^d \pmod{n} = y_1||y_2,$$

dove y_1 è formato da $k - k_0$ bit e y_2 è formato da k_0 bit. Bob allora calcola

$$m0^{k_1} = y_1 \oplus G\big(H(y_1) \oplus y_2\big).$$

La correttezza di questa decifrazione può essere giustificata nel modo seguente. Se il testo cifrato è la cifratura di m, allora

$$y_1 = x_1 = m0^{k_1} \oplus G(r) \qquad \text{e} \qquad y_2 = x_2 = r \oplus H(x_1).$$

Quindi

$$H(y_1) \oplus y_2 = H(x_1) \oplus r \oplus H(x_1) = r$$

e

$$y_1 \oplus G\big(H(y_1) \oplus y_2\big) = x_1 \oplus G(r) = m0^{k_1}.$$

Bob rimuove i k_1 bit uguali a zero dalla fine di $m0^{k_1}$ e ottiene m. Inoltre, Bob ha un controllo sull'integrità del testo cifrato. Se non ci sono k_1 zeri alla fine, allora il testo cifrato non corrisponde a una cifratura valida.
Questo metodo viene anche chiamato cifratura *plaintext-aware*. Si noti che il riempimento con x_2 dipende dal messaggio m e dal parametro casuale r. Questo rende più difficoltosi gli attacchi di testo cifrato scelto al sistema.

6.2.3 Attacchi basati sul tempo di esecuzione

Un altro tipo di attacco a RSA e a sistemi simili fu scoperto da Paul Kocher nel 1995, mentre era studente a Stanford. Mostrò come fosse possibile scoprire l'esponente di decifrazione misurando accuratamente i tempi di calcolo per una serie di decifrazioni. Anche se esistono modi per contrastare questo attacco, tale scoperta fu sconvolgente. Si pensava che il sistema fosse piuttosto sicuro poiché la struttura matematica sottostante era ben compresa. Invece l'attacco di Kocher dimostrò che un sistema poteva anche avere vulnerabilità inaspettate.
Ecco come funziona un attacco basato sul tempo di esecuzione. Si supponga che Eva sia in grado di osservare Bob che decifra vari testi cifrati y. Eva misura il tempo utilizzato per ogni y. La conoscenza di ogni y e del tempo che esso richiede per essere decifrato permette a Eva di trovare l'esponente di decifrazione d. Ma, innanzitutto, come può Eva ottenere tali informazioni? Ci sono molte situazioni in cui i messaggi cifrati vengono inviati a Bob e il suo computer li decifra e risponde automaticamente. La misurazione dei tempi di risposta è sufficiente per i presenti scopi.
Bisogna sapere quale hardware viene usato per calcolare $y^d \pmod n$. Si può usare questa informazione per valutare i tempi di calcolo per i vari passi che potenzialmente si verificano nel processo.
Si supponga che $y^d \pmod n$ venga calcolato mediante l'algoritmo presentato nell'Esercizio 23 del Capitolo 3, che qui procede nel modo seguente.

> *Sia $d = b_1 b_2 \dots b_w$ scritto in binario (per esempio, se $x = 1011$ si ha $b_1 = 1$, $b_2 = 0$, $b_3 = 1$, $b_4 = 1$). Siano y e n due interi. Si esegue la seguente procedura.*
>
> 1. *Partire con $k = 1$ e $s_1 = 1$.*
> 2. *Se $b_k = 1$, porre $r_k \equiv s_k y \pmod n$. Se $b_k = 0$, porre $r_k = s_k$.*
> 3. *Porre $s_{k+1} \equiv r_k^2 \pmod n$.*
> 4. *Se $k = w$, stop. Se $k < w$, aggiungere 1 a k e andare al punto (2).*
>
> *Allora $r_w \equiv y^d \pmod n$.*

Si noti che la moltiplicazione $s_k y$ viene effettuata solo quando il bit $b_k = 1$. Supponiamo, come accade in molti casi, che il tempo richiesto da questa moltiplicazione presenti una variabilità ragionevolmente grande.
Prima di continuare, abbiamo bisogno di alcuni strumenti della probabilità. Si supponga di avere un processo casuale che produce come output numeri reali t. Per noi t sarà il tempo in cui il computer completa un calcolo, dato un input y casuale. La media è il valore medio di questi output. Se registriamo gli output $t_1, \dots, t_n$, la media dovrebbe essere circa $m = (t_1 + \dots + t_n)/n$. La varianza di un processo casuale è circa

$$\mathrm{Var}(\{t_i\}) = \frac{(t_1 - m)^2 + \dots + (t_n - m)^2}{n}.$$

La deviazione standard è la radice quadrata della varianza e dà una misura di quanta variabilità c'è nei valori delle t_i.
Il fatto importante di cui avremo bisogno è che quando due processi casuali sono indipendenti, la varianza della somma dei loro output è la somma delle varianze dei due processi. In questo modo, per esempio, potremo spezzare il calcolo eseguito dal computer in due processi indipendenti, che impiegheranno rispettivamente i tempi t' e t''. Poiché il tempo totale t è $t' + t''$, $\mathrm{Var}(\{t_i\})$ dovrebbe essere circa $\mathrm{Var}(\{t_i'\}) + \mathrm{Var}(\{t_i''\})$.
Supponiamo ora che Eva conosca i testi cifrati $y_1, \dots, y_n$ e i tempi necessari per calcolare ogni $y_i^d \pmod n$. Supponiamo che conosca i bit $b_1, \dots, b_{k-1}$ dell'esponente d. Poiché conosce l'hardware che viene usato, sa quanto tempo è stato usato per calcolare $r_1, \dots, r_{k-1}$ nell'algoritmo precedente. Quindi conosce, per ogni y_i, il tempo t_i necessario per calcolare $r_k, \dots, r_w$.
Eva vuole determinare b_k. Se $b_k = 1$, si avrà una moltiplicazione $s_k y \pmod n$ per ogni testo cifrato y_i che viene elaborato. Se $b_k = 0$, questa moltiplicazione non c'è.
Sia t_i' il tempo richiesto dal computer per eseguire la moltiplicazione $s_k y \pmod n$, anche se Eva non sa ancora se questa moltiplicazione avviene realmente. Sia $t_i'' = t_i - t_i'$. Eva calcola $\mathrm{Var}(\{t_i\})$ e $\mathrm{Var}(\{t_i''\})$. Se $\mathrm{Var}(\{t_i\}) > \mathrm{Var}(\{t_i''\})$, allora Eva conclude che $b_k = 1$. Altrimenti $b_k = 0$. Dopo aver determinato b_k, passa a trovare tutti i bit nello stesso modo.
Perché questa tecnica funziona? Se la moltiplicazione viene eseguita, t_i'' è il tempo che il computer richiede per completare il calcolo dopo la moltiplicazione. È ragionevole supporre che t_i' e t_i'' siano output indipendenti l'uno dall'altro. Quindi

$$\mathrm{Var}(\{t_i\}) \approx \mathrm{Var}(\{t_i'\}) + \mathrm{Var}(\{t_i''\}) > \mathrm{Var}(\{t_i''\}).$$

Se la moltiplicazione non viene eseguita, t_i' è il tempo necessario per un'operazione che non ha legami con il calcolo e quindi è ragionevole supporre che t_i e t_i' siano indipendenti. Pertanto

$$\mathrm{Var}(\{t_i''\}) \approx \mathrm{Var}(\{t_i\}) + \mathrm{Var}(\{-t_i'\}) > \mathrm{Var}(\{t_i\}).$$

Non si poteva usare la media al posto della varianza, perché la media di $\{-t_i\}$ sarebbe stata negativa e l'ultima disuguaglianza non sarebbe stata valida. Tutto ciò che può essere dedotto dalla media è il numero totale di bit non nulli nella rappresentazione binaria di d.

Quanto appena visto dà una versione abbastanza semplice del metodo. In pratica, potrebbero essere necessarie varie modifiche, in base alla specifica situazione. Ma la strategia generale rimane la stessa. Per maggiori dettagli, si veda [Kocher].
Un analogo attacco a RSA funziona misurando la potenza consumata durante i calcoli. Si veda [Kocher et al.]. Attacchi come questo e l'attacco basato sul tempo di esecuzione possono essere prevenuti con accorgimenti appropriati nel progetto dell'implementazione fisica.

6.3 Test di primalità

Si supponga di avere un intero di 200 cifre e di dover stabilire se è o meno primo. Perché non dividere per tutti i primi minori della sua radice quadrata? Ci sono circa $4 \cdot 10^{97}$ primi minori di 10^{100}. Questo numero è molto maggiore del numero di particelle presenti nell'universo. Se il computer può elaborare 10^9 primi al secondo, il calcolo impiegherebbe circa 10^{81} anni. Chiaramente è necessario trovare metodi migliori. Alcuni di questi sono discussi in questo paragrafo.
Molti metodi di fattorizzazione si basano sul seguente

Principio fondamentale. *Sia n un intero. Se esistono due interi x e y per cui $x^2 \equiv y^2 \pmod n$ e $x \not\equiv \pm y \pmod n$, allora n è un numero composto. Inoltre,* $\text{MCD}(x-y, n)$ *è un fattore non banale di n.*

Dimostrazione. Sia $d = \text{MCD}(x-y, n)$. Se $d = n$, allora $x \equiv y \pmod n$, contro le ipotesi. Sia $d = 1$. Un risultato fondamentale sulla divisibilità è che se $a|bc$ e $\text{MCD}(a,b) = 1$, allora $a|c$ (si veda l'Esercizio 7 del Capitolo 3). Nel nostro caso, poiché n divide $x^2 - y^2 = (x-y)(x+y)$ e $d = 1$, si deve avere che n divide $x+y$, contro l'ipotesi che $x \not\equiv -y \pmod n$. Quindi $d \neq 1, n$, ossia d è un fattore non banale di n. □

Esempio. Poiché $12^2 \equiv 2^2 \pmod{35}$ e $12 \not\equiv \pm 2 \pmod{35}$, si ha che 35 è composto. Inoltre $\text{MCD}(12-2, 35) = 5$ è un fattore non banale di 35. ■

Anche se può sembrare sorprendente, fattorizzare e testare la primalità non sono la stessa cosa. È molto più facile dimostrare che un numero è composto che non fattorizzarlo. Esistono molti interi grandi che non sono mai stati fattorizzati anche se si sa che sono composti. Come può accadere questo? Vediamo un semplice esempio. Per il teorema di Fermat, se p è primo allora $2^{p-1} \equiv 1 \pmod p$. Usiamo questo fatto per dimostrare che 35 non è primo. Mediante quadrature successive, si ha che

$$\begin{aligned} 2^4 &\equiv 16 \\ 2^8 &\equiv 256 \equiv 11 \\ 2^{16} &\equiv 121 \equiv 16 \\ 2^{32} &\equiv 256 \equiv 11 \end{aligned}$$

(dove tutte le congruenze sono modulo 35). Quindi

$$2^{34} \equiv 2^{32}2^2 \equiv 11 \cdot 4 \equiv 9 \not\equiv 1 \pmod{35}.$$

Per il teorema di Fermat, 35 non può essere primo e così si è dimostrato che 35 è composto senza trovare alcun fattore.
Lo stesso ragionamento porta al seguente

Test di primalità di Fermat. *Sia $n > 1$ un intero. Sia a un intero casuale tale che $1 < a < n-1$. Se $a^{n-1} \not\equiv 1 \pmod n$, allora n è composto. Se $a^{n-1} \equiv 1 \pmod n$, allora n è probabilmente primo.*

Anche se i test di questo tipo sono normalmente chiamati "test di primalità", in realta dicono se un numero è o meno composto, poiché danno una risposta completamente certa solo nel caso in cui n sia composto. Il test di Fermat è molto accurato per n grande. Se dice che un numero è composto, allora è certamente così. Se dice che un numero è probabilmente primo, allora i risultati empirici mostrano che è molto probabile che sia così. Inoltre, poiché l'elevamento a potenza modulare è veloce, il test di Fermat può essere eseguito rapidamente.
L'elevamento a potenza modulare si ottiene mediante elevamenti al quadrato successivi. Se questi elevamenti vengono eseguiti opportunamente, il test di Fermat può essere combinato con il Principio Fondamentale per ottenere il seguente risultato più forte.

Test di primalità di Miller-Rabin. *Sia $n > 1$ un intero dispari. Sia $n - 1 = 2^k m$ con m dispari. Si scelga un intero casuale a tale che $1 < a < n-1$. Sia $b_0 \equiv a^m \pmod n$. Se $b_0 \equiv \pm 1 \pmod n$, allora ci si ferma e si dichiara che n è probabilmente primo. Altrimenti, sia $b_1 \equiv b_0^2 \pmod n$. Se $b_1 \equiv 1 \pmod n$, allora n è composto (e* $\text{MCD}(b_0 - 1, n)$ *è un fattore non banale di n). Se $b_1 \equiv -1 \pmod n$, allora ci si ferma e si dichiara che n è probabilmente primo. Altrimenti, sia $b_2 \equiv b_1^2 \pmod n$. Se $b_2 \equiv 1 \pmod n$, allora n è composto. Se $b_2 \equiv -1 \pmod n$, allora n è probabilmente primo. Si continua in questo modo finché non ci si ferma o non si raggiunge b_{k-1}. Se $b_{k-1} \not\equiv -1 \pmod n$, allora n è composto.*

Esempio. Sia $n = 561$. Allora $n - 1 = 560 = 16 \cdot 35$ e quindi $2^k = 2^4$ e $m = 35$. Sia $a = 2$. Allora

$$\begin{aligned} b_0 &\equiv 2^{35} \equiv 263 \pmod{561} \\ b_1 &\equiv b_0^2 \equiv 166 \pmod{561} \\ b_2 &\equiv b_1^2 \equiv 67 \pmod{561} \\ b_3 &\equiv b_2^2 \equiv 1 \pmod{561}. \end{aligned}$$

Poiché $b_3 \equiv 1 \pmod{561}$, si ha che 561 è composto. Inoltre, $\text{MCD}(b_2 - 1, 561) = 33$ è un fattore non banale di 561. ■

Se n è composto e $a^{n-1} \equiv 1 \pmod n$, allora si dice che n è pseudoprimo per la base a. Se a e n sono tali che n supera il test di Miller-Rabin, si dice che n è uno pseudoprimo forte per la base a. Nel Paragrafo 3.6 abbiamo mostrato che $2^{560} \equiv 1 \pmod{561}$. Quindi 561 è uno pseudoprimo per la base 2. Tuttavia, il precedente calcolo mostra che 561 non è uno pseudoprimo forte per la base 2. Per una data base, i pseudoprimi forti sono molto più rari degli pseudoprimi.
Fino a 10^{10}, ci sono 455052511 primi. Per la base 2 ci sono 14884 pseudoprimi e 3291 pseudoprimi forti. Quindi, calcolare $2^{n-1} \pmod n$ non riconosce correttamente un

numero composto in questo intervallo con probabilità inferiore a 1 su 30.000 e usando il test di Miller-Rabin con $a = 2$ fallisce con probabilità minore di 1 su 100.000.

Si può dimostrare che la probabilità che il test di Miller-Rabin fallisca nel riconoscere un numero composto per un a scelto a caso, è al più $1/4$. In realtà, fallisce molto meno frequentemente di quanto indicato. Si veda [Damgård et al.]. Se, per esempio, si ripete il test 10 volte con valori di a scelti a caso, allora ci si aspetta che la probabilità di certificare un numero composto come primo è al più $(1/4)^{10} \simeq 10^{-6}$. In pratica, usare il test per un singolo a è abbastanza accurato.

Anche se gli pseudoprimi forti sono rari, si è dimostrato (si veda [Alford et al.]) che, per ogni insieme finito B di basi, esistono infiniti interi che sono pseudoprimi forti per ogni $b \in B$. Il primo pseudoprimo forte per tutte le basi $b = 2, 3, 5, 7$ è 3215031751. Esiste un numero di 337 cifre che è uno pseudoprimo forte per tutte le basi date da primi < 200.

Si supponga di dover trovare un primo di circa 100 cifre. Il teorema dei numeri primi afferma che la densità dei primi attorno a x è approssimativamente $1/\ln x$. Quando $x = 10^{100}$, questo dà una densità di circa $1/\ln(10^{100}) = 1/230$. Poiché si possono saltare i numeri pari, questo può essere elevato a $1/115$. Si prenda un punto di partenza a caso e si buttino via i numeri pari (e i multipli degli altri primi piccoli). Si controlli mediante il test di Miller-Rabin ogni numero che rimane, uno dopo l'altro. Questo tenderà a eliminare tutti i numeri composti. In media, occorrono meno di 100 applicazioni del test di Miller-Rabin per trovare un candidato verosimilmente primo e quindi questo può essere svolto abbastanza rapidamente. Se dobbiamo essere completamente certi che il numero in questione sia primo, ci sono test di primalità più sofisticati che possono testare un numero di 100 cifre in pochi secondi.

Perché il test funziona? Si supponga, per esempio, che $b_3 \equiv 1 \pmod n$. Questo significa che $b_2^2 \equiv 1^2 \pmod n$. Applicando il Principio Fondamentale, si ha che $b_2 \equiv \pm 1 \pmod n$ o che $b_2 \not\equiv \pm 1 \pmod n$ e n è composto. In quest'ultimo caso, $\text{MCD}(b_2 - 1, n)$ è fattore non banale di n. Nel primo caso, l'algoritmo si sarebbe fermato al passo precedente. Se si raggiunge b_{k-1}, si è calcolato $b_{k-1} \equiv a^{(n-1)/2} \pmod n$. Il quadrato di questo è a^{n-1}, che, se n è primo, deve essere $1 \pmod n$, per il teorema di Fermat. Quindi, se n è primo, $b_{k-1} \equiv \pm 1 \pmod n$. Tutte le altre scelte significano che n è composto. Inoltre, se $b_{k-1} \equiv 1$, allora, se non ci si è fermati a un passo precedente, $b_{k-2}^2 \equiv 1^2 \pmod n$ con $b_{k-2} \not\equiv \pm 1 \pmod n$. Questo significa che n è composto (e che può essere fattorizzato).

In pratica, se n è composto, in genere si raggiunge b_{k-1} e questo numero è diverso da $\pm 1 \pmod n$. Infatti, di solito $a^{n-1} \not\equiv 1 \pmod n$. Questo significa che il teorema di Fermat fallisce e che pertanto n non è primo.

Per esempio, sia $n = 299$ e $a = 2$. Poiché $2^{298} \equiv 140 \pmod{299}$, il teorema di Fermat e anche il test di Miller-Rabin dicono che 299 non è primo (senza fattorizzarlo). Il motivo per cui questo accade è il seguente. Si noti che $299 = 13 \cdot 23$. Un semplice calcolo mostra che $2^{12} \equiv 1 \pmod{13}$ e che nessun esponente più piccolo va bene. Infatti, $2^j \equiv 1 \pmod{13}$ se e solo se j è un multiplo di 12. Poiché 298 non è un multiplo di 12, si ha $2^{298} \not\equiv 1 \pmod{13}$ e quindi $2^{298} \not\equiv 1 \pmod{299}$. Analogamente, $2^j \equiv 1 \pmod{23}$ se e solo se j è un multiplo di 11, da cui si può ancora dedurre che $2^{298} \not\equiv 1 \pmod{299}$. Se in questo caso il teorema di Fermat (e il test di Miller-Rabin) avessero dato la risposta sbagliata, allora $13 \cdot 23 - 1$ sarebbe dovuto essere un multiplo di $12 \cdot 11$.

Consideriamo ora il caso generale in cui $n = pq$ è prodotto di due primi. Per semplicità, si consideri il caso in cui $p > q$ e si supponga che $a^k \equiv 1 \pmod p$ se e solo se $k \equiv 0 \pmod{p-1}$. Questo significa che a è una radice primitiva modulo p (ci sono $\varphi(p-1)$ tali a modulo p). Poiché $0 < q - 1 < p - 1$, si ha

$$n - 1 \equiv pq - 1 \equiv q(p-1) + q - 1 \not\equiv 0 \pmod{p-1}.$$

Quindi, per la nostra scelta di a, si ha $a^{n-1} \not\equiv 1 \pmod p$ e quindi $a^{n-1} \not\equiv 1 \pmod n$. Un analogo ragionamento mostra che di solito $a^{n-1} \not\equiv 1 \pmod n$ anche per molte altre scelte di a.

Ma se ci si trova nel caso in cui $a^{n-1} \equiv 1 \pmod n$, cosa accade? Consideriamo l'esempio di $n = 561$. Poiché $561 = 3 \cdot 11 \cdot 17$, andiamo a vedere cosa accade alla successione b_0, b_1, b_2, b_3 modulo 3, modulo 11 e modulo 17:

$$\begin{aligned}
b_0 &\equiv -1 \pmod 3, & &\equiv -1 \pmod{11}, & &\equiv 8 \pmod{17}\\
b_1 &\equiv 1 \pmod 3, & &\equiv 1 \pmod{11}, & &\equiv -4 \pmod{17}\\
b_2 &\equiv 1 \pmod 3, & &\equiv 1 \pmod{11}, & &\equiv -1 \pmod{17}\\
b_3 &\equiv 1 \pmod 3, & &\equiv 1 \pmod{11}, & &\equiv 1 \pmod{17}.
\end{aligned}$$

Poiché $b_3 \equiv 1 \pmod{561}$, si ha $b_2^2 \equiv b_3 \equiv 1$ modulo tutti e tre i primi. Ma non c'è ragione che sia b_3 il primo elemento per cui $b_i \equiv 1$ modulo un particolare primo. Si ha già $b_1 \equiv 1$ modulo 3 e modulo 11, ma bisogna aspettare b_3 quando si lavora modulo 17. Quindi, $b_2^2 \equiv b_3 \equiv 1$ modulo 3, modulo 11 e modulo 17, ma b_2 è congruente a 1 solo modulo 3 e modulo 11. Quindi $b_2 - 1$ contiene i fattori 3 e 11, ma non il fattore 17. Questo è dovuto al fatto che $\text{MCD}(b_2 - 1, 561)$ trova il fattore 33 di 561. Il motivo per cui si può fattorizzare 561 usando questo metodo è che la successione b_0, b_1, ... raggiunge 1 modulo i primi in tempi diversi.

Più in generale, si consideri il caso $n = pq$ (il caso in cui n è prodotto di tre o più primi è analogo) e si supponga $a^{n-1} \equiv 1 \pmod n$. Come osservato in precedenza, è molto improbabile che questo si verifichi; ma se si verifica, si va a vedere cosa accade modulo p e modulo q. È probabile che le successioni $b_i \pmod p$ e $b_i \pmod q$ raggiungano -1 e poi 1 in momenti diversi, proprio come nell'esempio di 561. In questo caso, esisterà un indice i per cui $b_i \equiv -1 \pmod p$, ma $b_i \equiv 1 \pmod q$. Di conseguenza $b_i^2 \equiv 1 \pmod n$, ma $b_i \not\equiv \pm 1 \pmod n$. Quindi n potrà essere fattorizzato.

Il solo caso in cui n può superare il test di Miller-Rabin è quando $a^{n-1} \equiv 1 \pmod n$ e le successioni $b_i \pmod p$ e $b_i \pmod q$ raggiungono 1 allo stesso tempo. Questo però accade raramente.

Il seguente test di primalità, simile al test di Miller-Rabin, sfrutta il simbolo di Jacobi (si veda il Paragrafo 3.10).

Test di primalità di Solovay-Strassen. *Sia n un intero dispari. Si scelgano vari interi casuali a con $1 < a < n - 1$. Se*

$$\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod n$$

per qualche a, allora n è composto. Se

$$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod n$$

per ogni a, allora n è probabilmente primo.

Se n è primo, allora (per la Proposizione del Paragrafo 3.10) il test dichiarerà che n è un probabile primo.
Il simbolo di Jacobi può essere calcolato rapidamente, come nel Paragrafo 3.10. Anche l'elevamento a potenza modulare può essere eseguito rapidamente.
Per esempio, poiché

$$\left(\frac{2}{15}\right) = -1 \not\equiv 23 \equiv 2^{(15-1)/2} \pmod{15},$$

si ha che 15 non è primo. Come nei test di Miller-Rabin, in genere $a^{(n-1)/2} \pmod n$ non si riduce a ± 1. Ecco un caso in cui capita:

$$\left(\frac{2}{341}\right) = -1 \not\equiv +1 \equiv 2^{(341-1)/2} \pmod{341}.$$

Di conseguenza, 341 è composto.

Entrambi i test di Miller-Rabin e di Solovay-Strassen possono essere eseguiti rapidamente in pratica. Tuttavia, quando p è primo, questi test non danno una dimostrazione rigorosa del fatto che p è primo. Esistono anche test che effettivamente dimostrano la primalità di p, ma sono parecchio più lenti e vengono usati solo quando è indispensabile dimostrare la primalità del numero. Questi metodi sono quasi tutti probabilistici, ossia non garantiscono il successo anche se funzionano con probabilità molto alta in ogni caso particolare. Nel 2002 Agrawal, Kayal e Saxena [Agrawal et al.] diedero un algoritmo deterministico a tempo polinomiale che stabilisce se un numero è primo o meno. Questo significa che il tempo di calcolo è sempre (e non solo probabilmente) limitato da una costante per una potenza di $\log p$. Questo algoritmo ha rappresentato un grande passo in avanti dal punto di vista teorico, ma per il momento non è ancora stato migliorato al punto di poter competere con gli algoritmi probabilistici.
Per ulteriori informazioni sui test di primalità e sulla loro storia, si veda [Williams].

6.4 Fattorizzazione

Passiamo ora alla fattorizzazione. Il semplice metodo che consiste nel dividere un intero n per tutti i primi $p \leq \sqrt{n}$ è troppo lento nella maggior parte dei casi. Per molti anni, si è lavorato per sviluppare algoritmi più efficienti. Qui ne presenteremo alcuni. Nel Capitolo 16 verrà presentato un metodo che utilizza le curve ellittiche e nel Capitolo 19 si mostrerà come un ipotetico computer quantistico possa fattorizzare efficientemente.
Un altro metodo, anch'esso troppo lento, è il metodo di **fattorizzazione di Fermat**. L'idea è quella di esprimere n come differenza di due quadrati: $n = x^2 - y^2$. Allora $n = (x+y)(x-y)$ dà una fattorizzazione di n. Per esempio, per fattorizzare $n = 295927$ si calcolano i numeri $n+1^2$, $n+2^2$, $n+3^2$, ... fino a quando non si trova un quadrato. Nel nostro caso, $295927 + 3^2 = 295936 = 544^2$. Quindi

$$295927 = (544+3)(544-3) = 547 \cdot 541.$$

Il metodo di Fermat funziona bene quando n è il prodotto di due primi molto vicini tra loro. Se $n = pq$, occorrono $|p-q|/2$ passi per trovare la fattorizzazione. Ma se p e q sono due primi di 100 cifre scelti a caso, è probabile che anche $|p-q|$ sia molto grande, probabilmente sarà anch'esso di circa 100 cifre. Di conseguenza, è improbabile che la fattorizzazione di Fermat funzioni. Tuttavia, per sicurezza, i primi per un modulo RSA vengono spesso scelti di grandezza leggermente diversa.
Passiamo ora a metodi più moderni. Se uno dei fattori primi di n ha una qualche proprietà speciale, a volte è più facile fattorizzare n. Per esempio, se p divide n e $p-1$ ha solo fattori primi piccoli, è efficace il seguente metodo inventato da Pollard nel 1974.

Algoritmo di fattorizzazione $p-1$. *Si scelga un intero $a > 1$. Spesso si usa $a = 2$. Si scelga una limitazione B. Si calcoli $b \equiv a^{B!} \pmod n$ nel modo seguente. Siano $b_1 \equiv a \pmod n$ e $b_j \equiv b_{j-1}^j \pmod n$. Allora $b_B \equiv b \pmod n$. Sia $d = \mathrm{MCD}(b-1, n)$. Se $1 < d < n$, si è trovato un fattore non banale di n.*

Sia p un fattore primo di n per cui $p-1$ ha solo fattori primi piccoli. Allora è probabile che $p-1$ divida $B!$, ossia che $B! = (p-1)k$. Per il teorema di Fermat si ha $b \equiv a^{B!} \equiv (a^{p-1})^k \equiv 1 \pmod p$ e quindi p apparirà nel massimo comune divisore di $b-1$ e n. Se q è un altro fattore primo di n, è improbabile che $b \equiv 1 \pmod q$, a meno che anche $q-1$ abbia solo fattori primi piccoli. Se $d = n$, non tutto è perduto. In questo caso, si ha un esponente r (ossia $B!$) e un a tali che $a^r \equiv 1 \pmod n$. C'è una buona probabilità che il metodo di fattorizzazione dell'esponente (che verrà spiegato più avanti in questo paragrafo) permetta di fattorizzare n. In alternativa, si può scegliere un valore più piccolo di B e ripetere il calcolo.
Come si sceglie la limitazione B? Se si sceglie un B piccolo, allora l'algoritmo sarà molto rapido ma la probabilità di successo sarà molto bassa. Se si sceglie un B molto grande, allora l'algoritmo sarà molto lento. Il valore che viene effettivamente usato dipende, di volta in volta, dalla particolare situazione.
Nelle applicazioni, si usano interi $n = pq$ che sono il prodotto di due primi, ma che sono difficili da fattorizzare. Quindi, bisogna garantire che $p-1$ abbia almeno un fattore primo grande. Questo è facile. Supponiamo di volere che p abbia circa 100 cifre. Si sceglie un primo p_0 grande, magari pari a circa 10^{40}. Si cercano gli interi della forma $kp_0 + 1$, con k che varia tra alcuni interi attorno a 10^{60}. Si controlla la primalità di $kp_0 + 1$ mediante il test di Miller-Rabin, come prima. In media, questo dovrebbe dare uno dei valori desiderati di p in meno di 100 passi. Ora si sceglie un primo q_0 grande e si segue la stessa procedura per ottenere q. In questo modo $n = pq$ sarà difficile da fattorizzare mediante il metodo $p-1$.
Il metodo di fattorizzazione basato sulle curve ellittiche (si veda il Paragrafo 16.3) generalizza il metodo $p-1$. Tuttavia, esso utilizza alcuni numeri casuali vicini a $p-1$ e richiede soltanto che almeno uno di essi abbia solo fattori primi piccoli. Questo permette al metodo di trovare molti più primi p, non soltanto quelli in cui $p-1$ ha solo fattori primi piccoli.

6.4.1 Crivello quadratico

Essendo alla base dei migliori metodi di fattorizzazione attuali, enunciamo nuovamente il seguente risultato ottenuto nel Paragrafo 6.3.

Principio fondamentale. *Sia n un intero. Se esistono due interi x e y tali che $x^2 \equiv y^2 \pmod{n}$ e $x \not\equiv \pm y \pmod{n}$, allora n è composto. Inoltre, $\mathrm{MCD}(x-y, n)$ è un fattore non banale di n.*

Supponiamo di dover fattorizzare $n = 3837523$. Osserviamo che

$$\begin{aligned} 9398^2 &\equiv 5^5 \cdot 19 \pmod{3837523} \\ 19095^2 &\equiv 2^2 \cdot 5 \cdot 11 \cdot 13 \cdot 19 \pmod{3837523} \\ 1964^2 &\equiv 3^2 \cdot 13^3 \pmod{3837523} \\ 17078^2 &\equiv 2^6 \cdot 3^2 \cdot 11 \pmod{3837523}. \end{aligned}$$

Se si moltiplicano le relazioni, si ha

$$\begin{aligned} (9398 \cdot 19095 \cdot 1964 \cdot 17078)^2 &\equiv (2^4 \cdot 3^2 \cdot 5^3 \cdot 11 \cdot 13^2 \cdot 19)^2 \\ 2230387^2 &\equiv 2586705^2. \end{aligned}$$

Poiché $2230387 \not\equiv \pm 2586705 \pmod{3837523}$, ora si può fattorizzare 3837523 calcolando

$$\mathrm{MCD}(2230387 - 2586705, 3837523) = 1093.$$

L'altro fattore è $3837523/1093 = 3511$.
Ecco un modo di vedere il calcolo appena svolto. Innanzitutto si generano i quadrati che quando sono ridotti modulo $n = 3837523$ possono essere scritti come prodotti di primi piccoli (nel nostro caso, primi minori di 20). Questo insieme di primi è chiamato **base di fattorizzazione**. Diremo come generare questi quadrati tra breve. Ognuno di questi quadrati dà la riga di una matrice, dove gli elementi sono dati dagli esponenti dei primi 2, 3, 5, 7, 11, 13, 17, 19. Per esempio, la relazione $17078^2 \equiv 2^6 \cdot 3^2 \cdot 11 \pmod{3837523}$ dà la riga 6, 2, 0, 0, 1, 0, 0, 0.
In aggiunta alle relazioni precedenti, supponiamo di aver trovato anche le relazioni seguenti:

$$\begin{aligned} 8077^2 &\equiv 2 \cdot 19 \pmod{3837523} \\ 3397^2 &\equiv 2^5 \cdot 5 \cdot 13^2 \pmod{3837523} \\ 14262^2 &\equiv 5^2 \cdot 7^2 \cdot 13 \pmod{3837523}. \end{aligned}$$

Si ottiene la matrice

$$\begin{array}{c} 9398 \\ 19095 \\ 1964 \\ 17078 \\ 8077 \\ 3397 \\ 14262 \end{array} \begin{pmatrix} 0 & 0 & 5 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 6 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Ora si cercano le dipendenze lineari modulo 2 tra le righe della matrice. Alcune di esse sono le seguenti.

Anno	Numero di cifre
1964	20
1974	45
1984	71
1994	129
1999	155
2003	174
2005	200

Tabella 6.1 Record di fattorizzazione

1. $1° + 5° + 6° = (6,0,6,0,0,2,0,2) \equiv 0 \pmod{2}$
2. $1° + 2° + 3° + 4° = (8,4,6,0,2,4,0,2) \equiv 0 \pmod{2}$
3. $3° + 7° = (0,2,2,2,0,4,0,0) \equiv 0 \pmod{2}$

Quando si ha una dipendenza di questo tipo, il prodotto dei numeri dà un quadrato. Per esempio, le tre dipendenze precedenti danno

1. $(9398 \cdot 8077 \cdot 3397)^2 \equiv 2^6 \cdot 5^6 \cdot 13^2 \cdot 19^2 \equiv (2^3 \cdot 5^3 \cdot 13 \cdot 19)^2$
2. $(9398 \cdot 19095 \cdot 1964 \cdot 17078)^2 \equiv (2^3 \cdot 3^2 \cdot 5^3 \cdot 11 \cdot 13^2 \cdot 19)^2$
3. $(1964 \cdot 14262)^2 \equiv (3 \cdot 5 \cdot 7 \cdot 13^2)^2$

Quindi, si ha $x^2 \equiv y^2 \pmod{n}$ per vari valori di x e y. Se $x \not\equiv \pm y \pmod{n}$, allora $\mathrm{MCD}(x-y, n)$ dà un fattore non banale di n. Se $x \equiv \pm y \pmod{n}$, allora $\mathrm{MCD}(x-y, n) = 1$ o n e così non si ottiene alcuna fattorizzazione. Nei nostri tre esempi, si ha

1. $3590523^2 \equiv 247000^2$ ma $3590523 \equiv -247000 \pmod{3837523}$
2. $2230387^2 \equiv 2586705^2$ e $\mathrm{MCD}(2230387 - 2586705, 3837523) = 1093$
3. $1147907^2 \equiv 17745^2$ e $\mathrm{MCD}(1147907 - 17745, 3837523) = 1093$

Torniamo alla domanda fondamentale: come si trovano i numeri 9398, 19095 e così via? L'idea è di generare i quadrati che sono leggermente più grandi di un multiplo di n, in modo che siano piccoli modulo n. Questo significa che c'è una buona probabilità che siano prodotti di primi piccoli. Un modo facile è quello di considerare numeri della forma $\lfloor \sqrt{in} + j \rfloor$ per j piccolo e per vari valori di i. Qui $\lfloor x \rfloor$ indica il più grande intero minore o uguale a x. Il quadrato di tale numero è circa $in + 2j\sqrt{in} + j^2$, che è circa $2j\sqrt{in} + j^2$ modulo n. Finché i non è troppo grande, questo numero è abbastanza piccolo e quindi c'è una buona probabilità che sia un prodotto di primi piccoli. Nel calcolo precedente, per esempio, si ha $8077 = \lfloor \sqrt{17n} + 1 \rfloor$ e $9398 = \lfloor \sqrt{23n} + 4 \rfloor$.
Il metodo appena usato è alla base di molti dei migliori metodi di fattorizzazione attuali. Il passo principale è quello di generare relazioni di congruenza del tipo

$$x^2 \equiv \text{prodotto di primi piccoli}.$$

Una versione migliorata del metodo precedente è chiamata crivello quadratico. Un metodo recente, il crivello del campo numerico, usa tecniche più sofisticate per generare tali relazioni e in molte situazioni è alquanto più veloce. Si veda [Pomerance] per una descrizione di questi due metodi e per una discussione sulla storia dei metodi di fattorizzazione. Si veda anche l'Esercizio 28.

Una volta che si hanno varie relazioni di congruenza, esse sono messe in una matrice, come prima. Se si hanno più righe che colonne, si ha necessariamente almeno una relazione di dipendenza lineare modulo 2 tra le righe. Questo porta a una congruenza $x^2 \equiv y^2 \pmod{n}$. Naturalmente, come nel caso di $1° + 5° + 6° \equiv 0 \pmod 2$ considerato precedentemente, si può finire con $x \equiv \pm y$, nel qual caso non si ottiene una fattorizzazione. Ma questa situazione è attesa al più metà delle volte. Così se si hanno abbastanza relazioni (per esempio, se ci sono molte più righe che colonne) allora si dovrebbe avere una relazione che dà $x^2 \equiv y^2$ con $x \not\equiv \pm y$. In questo caso $\text{MCD}(x-y, n)$ è un fattore non banale di n.

Nell'ultima metà del ventesimo secolo, ci sono stati enormi progressi nella fattorizzazione, in parte per lo sviluppo dei computer e in parte per il miglioramento degli algoritmi. Un impulso ancora maggiore è stato dato dall'uso della fattorizzazione nella crittologia, specialmente nell'algoritmo RSA. La Tabella 6.1 riporta i record di fattorizzazione (in termini del numero di cifre decimali) per vari anni.

6.4.2 Metodi teorici

A prima vista, sembra che il test di Miller-Rabin porti alla fattorizzazione di n piuttosto spesso; ma ciò che accade normalmente è che si raggiunge b_{k-1} senza aver mai $b_u \equiv \pm 1 \pmod n$. Il problema è che in genere $a^{n-1} \not\equiv 1 \pmod n$. D'altra parte, se si ha un esponente r (anche diverso da $n-1$) tale che $a^r \equiv 1 \pmod n$ per ogni a con $\text{MCD}(a, n) = 1$, allora spesso è possibile fattorizzare n. Si noti che tale esponente r deve essere pari (se $n > 2$); poiché si può prendere $a \equiv -1 \pmod n$, occorre che $(-1)^r \equiv 1$.

Metodo di fattorizzazione dell'esponente universale. *Sia $r > 0$ un esponente tale che $a^r \equiv 1 \pmod n$ per ogni a con $\text{MCD}(a, n) = 1$. Sia $r = 2^k m$ con m dispari. Si scelga a caso un a con $1 < a < n-1$. Si assuma che $\text{MCD}(a, n) = 1$ (altrimenti, se $\text{MCD}(a, n) \neq 1$, si avrebbe un fattore di n). Posto $b_0 \equiv a^m \pmod n$, sia $b_{u+1} \equiv b_u^2 \pmod n$ per $0 \leq u \leq k-1$. Se $b_0 \equiv 1 \pmod n$, allora ci si ferma e si prova con un a differente. Se esiste un u per cui $b_u \equiv -1 \pmod n$, ci si ferma e si prova con un a differente. Se esiste un u per cui $b_{u+1} \equiv 1 \pmod n$, ma $b_u \not\equiv \pm 1 \pmod n$, allora $\text{MCD}(b_u - 1, n)$ è un fattore non banale di n.*

Questo metodo somiglia molto al test di Miller-Rabin. La differenza è che l'esistenza di r garantisce che $b_{u+1} \equiv 1 \pmod n$ per qualche u, cosa che non capita così spesso con il metodo di Miller-Rabin. Provando con pochi valori di a si ha una probabilità molto alta di fattorizzare n.

Naturalmente, ci si può chiedere come trovare un esponente r. In generale, sembra che sia molto difficile trovare tale esponente e che questo test non possa essere usato in pratica. Tuttavia, questo test è utile per dimostrare che la conoscenza dell'esponente di decifrazione nell'algoritmo RSA permette di fattorizzare il modulo.

In alcune situazioni, non si conosce un esponente universale, ma si conosce un esponente r che funziona per un valore di a. A volte questo permette di fattorizzare n.

Metodo di fattorizzazione dell'esponente universale. *Siano $r > 0$ un esponente e a un intero tale che $a^r \equiv 1 \pmod n$. Sia $r = 2^k m$ con m dispari. Posto $b_0 \equiv a^m \pmod n$, sia $b_{u+1} \equiv b_u^2 \pmod n$ per $0 \leq u \leq k-1$. Se $b_0 \equiv 1 \pmod n$, allora stop; la procedura fallisce nel fattorizzare n. Se esiste un u per cui $b_u \equiv -1 \pmod n$, ci si ferma; la procedura fallisce nel fattorizzare n. Se esiste un u per cui $b_{u+1} \equiv 1 \pmod n$, ma $b_u \not\equiv \pm 1 \pmod n$, allora $\text{MCD}(b_u - 1, n)$ è un fattore non banale di n.*

Naturalmente, se si prende $a = 1$, allora qualunque r va bene. Ma allora $b_0 = 1$ e quindi il metodo fallisce. Ma se a e r sono ottenuti mediante un metodo ragionevolmente sensato, c'è una buona probabilità che questo metodo riesca a fattorizzare n.

6.5 Sfida RSA

Quando l'algoritmo RSA fu reso pubblico per la prima volta nel 1977, gli autori lanciarono la seguente sfida, nota con il nome di *RSA Challenge*.

Sia

$$n = 114381625757888867669235779976146612010218296721242362\\562561842935706935245733897830597123563958705058989075\\147599290026879543541$$

il modulo RSA e sia $e = 9007$ l'esponente di cifratura. Il testo cifrato è

$$c = 968696137546220614771409222543558829057599911245743198\\746951209308162982251457083569314766228839896280133919\\90551829945157815154.$$

Trovare il messaggio.

L'unico modo che si conosce per trovare il testo in chiaro è quello di fattorizzare n. Nel 1977 si stimava che i metodi di fattorizzazione di allora avrebbero impiegato $4 \cdot 10^{16}$ anni per ottenere questa fattorizzazione e questo permetteva agli autori di sentirsi sicuri nell'offrire 100$ a chiunque fosse riuscito a decifrare il messaggio prima del 1° Aprile 1982. Tuttavia, le tecniche migliorarono e nel 1994 Atkins, Graff, Lenstra e Leyland riuscirono a fattorizzare n. Usarono 524339 primi "piccoli", ovvero quelli minori di 16333610, e in più permisero che le fattorizzazioni potessero includere fino a due primi "grandi" compresi tra 16333610 e 2^{30}. L'idea di permettere primi grandi è la seguente: se un primo grande q appare in due relazioni differenti, queste possono essere moltiplicate per ottenere una relazione con q al quadrato. Moltiplicando per $q^{-2} \pmod n$ si ottiene una relazione che coinvolge solo primi piccoli. Nello stesso modo, se ci sono più relazioni, ognuna con gli stessi due primi grandi, un analogo processo porta a una relazione con solo primi piccoli. Per il "paradosso del compleanno" (si veda il Paragrafo 8.4) dovrebbero esistere vari casi in cui un primo grande compare in più di una relazione.

Seicento persone, con un totale di 1600 computer che lavoravano nel tempo libero, trovarono relazioni di congruenza del tipo desiderato. Queste relazioni venivano poi

inviate per e-mail a una macchina centrale, che eliminava le ripetizioni e memorizzava i risultati in una matrice molto grande. Dopo sette mesi, ottennero una matrice con 524339 colonne e 569466 righe. Fortunatamente, la matrice era sparsa, ossia la maggior parte degli elementi era uguale a zero e quindi poteva essere memorizzata efficientemente. Mediante l'eliminazione gaussiana, la matrice venne ridotta a una matrice non sparsa con 188160 colonne e 188614 righe. Questo richiese un po' meno di 12 ore. Con altre 45 ore di calcolo, trovarono 205 dipendenze. Le prime tre diedero una fattorizzazione banale di n, ma la quarta diede i fattori

$$p = 3490529510847650949147849619903898133417764638493387843990820577,$$

$$q = 3276913299326670954996198819083446141317764296799294253979828853 3.$$

Calcolando $9007^{-1} \pmod{(p-1)(q-1)}$ si ottenne l'esponente di decifrazione

$$d = 1066986143685780244428687713289201547807099066339378628012262244966310631259117744708733401685974623065539685445132771090536060 95.$$

Infine, calcolando $c^d \pmod{n}$ si ottenne il messaggio di testo in chiaro

$$2008050013010709030023151804190001180500191721050113091908001519190906180107 05,$$

che, tradotto in lettere usando $a = 01$, $b = 02$, ..., blank $= 00$, diventa

the magic words are squeamish ossifrage

(uno *squeamish ossifrage* è, letteralmente, un "gipeto ipersensibile"; il messaggio fu scelto in modo che nessuno potesse decifrarlo indovinando il testo in chiaro e mostrando che si cifrava nel testo cifrato). Per maggiori dettagli su questa fattorizzazione, si veda [Atkins et al.].

6.6 Applicazione alla verifica di trattati

Le nazioni A e B hanno firmato un trattato in cui si bandiscono i test nucleari. Ora ognuna di esse vuole essere sicura che l'altra non sperimenti alcuna bomba. Come può, per esempio, la nazione A usare dei dati sismici per monitorare la nazione B? La nazione A vuole mettere dei sensori sul territorio di B che gli inviino i dati. Si hanno due problemi.

1. La nazione A vuole essere sicura che la nazione B non modifichi i dati.
2. La nazione B vuole vedere il messaggio prima che venga inviato, per essere sicura che non vengano trasmessi altri dati (per esempio, per fini spionistici).

Queste richieste apparentemente contraddittorie possono essere soddisfatte usando RSA al contrario. Innanzitutto, A sceglie $n = pq$ come prodotto di due primi grandi e sceglie l'esponente e di cifratura e l'esponente d di decifrazione. I numeri n ed e vengono dati a B, mentre p, q e d vengono tenuti segreti. Il sensore (collocato in profondità nel terreno e a prova di manomissione) raccoglie i dati x e usa d per cifrare x in $y \equiv x^d \pmod{n}$. Entrambi x e y vengono inviati inizialmente alla nazione B che controlla se $y^e \equiv x \pmod{n}$. Se è così, la nazione B sa che il messaggio cifrato y corrisponde al dato x e quindi inoltra la coppia x, y ad A. Allora anche la nazione A controlla se $y^e \equiv x \pmod{n}$. Se è così, A può essere sicura che il numero x non è stato modificato, poiché (assegnato x) risolvere $y^e \equiv x \pmod{n}$ rispetto a y è equivalente a decifrare il messaggio RSA x e questo è ritenuto un problema difficile. Naturalmente, B avrebbe potuto scegliere un numero y primo e poi porre $x \equiv y^e \pmod{n}$. In questo caso però x probabilmente non sarebbe stato un messaggio dotato di senso e quindi A avrebbe capito che qualcosa era stato cambiato.
Il metodo precedente è essenzialmente lo schema di firma RSA, che verrà studiato nel Paragrafo 9.1.

6.7 Concetto di chiave pubblica

Nel 1976 Diffie e Hellman descrissero il concetto di chiave pubblica crittografica, anche se a quel tempo non era nota alcuna realizzazione pubblica di tale concetto (come osservato nell'introduzione di questo capitolo, Clifford Cocks dell'agenzia crittografica inglese CESG aveva però inventato una versione segreta di RSA già nel 1973). In questo paragrafo, verrà presentata la teoria generale dei sistemi a chiave pubblica.
Esistono varie implementazioni della crittografia a chiave pubblica oltre RSA. Negli ultimi capitoli descriveremo tre di essi. Il primo è dovuto a ElGamal e si basa sulla difficoltà di trovare i logaritmi discreti. Il secondo è NTRU ed usa metodi che si basano su reticoli. Il terzo è dovuto a McEliece e fa uso dei codici correttori di errori. Esistono anche sistemi a chiave pubblica basati sul problema dello zaino (anche se in questo libro non verranno trattati). Alcune versioni sono state violate e in generale si sospetta che siano più deboli di altri sistemi come RSA e ElGamal.
Un **crittosistema a chiave pubblica** è formato da varie componenti. Innanzitutto, c'è l'insieme M dei messaggi possibili (potenziali testi in chiaro e testi cifrati). C'è anche l'insieme K delle "chiavi". Queste non sono esattamente le chiavi di cifratura/decifrazione; in RSA una chiave k è una terna (e,d,n) con $ed \equiv 1 \pmod{\varphi(n)}$. Per ogni chiave k, esiste una funzione di cifratura E_k e una funzione di decifrazione D_k. Di solito, E_k e D_k sono funzioni da M ad M, anche se ci sono delle varianti in cui i testi in chiaro e i testi cifrati possono appartenere a insiemi diversi. Queste componenti devono soddisfare le seguenti proprietà.

1. $E_k(D_k(m)) = m$ e $D_k(E_k(m)) = m$ per ogni $m \in M$ e per ogni $k \in K$.
2. Per ogni m e per ogni k, i valori di $E_k(m)$ e $D_k(m)$ si possono calcolare facilmente.
3. Per quasi ogni $k \in K$, se si conosce solo la funzione E_k, il problema di trovare un algoritmo che calcoli D_k è computazionalmente intrattabile.
4. Dato $k \in K$, è facile trovare le funzioni E_k e D_k.

La richiesta (1) dice che la cifratura e la decifrazione si annullano a vicenda. La richiesta (2) è necessaria per avere cifratura e decifrazione efficienti. Grazie a (4), un utente può scegliere a caso in K una k segreta e ottenere le funzioni E_k e D_k. La richiesta (3) è ciò che rende il sistema a chiave pubblica. Poiché è difficile determinare D_k a partire da E_k, è possibile pubblicare E_k senza compromettere la sicurezza del sistema.

Verifichiamo che RSA soddisfi tutte queste richieste. Come spazio dei messaggi si può considerare l'insieme di tutti gli interi non negativi. Come osservato in precedenza, una chiave RSA è una terna $k = (e,d,n)$. La funzione di cifratura è

$$E_k(m) = m^e \pmod{n},$$

dove m viene spezzato in blocchi se $m \geq n$. La funzione di decifrazione è

$$D_k(m) = m^d \pmod{n},$$

dove, ancora, m viene spezzato in blocchi quando è necessario. Le funzioni E_k e D_k vengono immediatamente determinate dalla conoscenza di k (richiesta (4)) e sono facili da calcolare (richiesta (2)). Sono una l'inversa dell'altra poiché $ed \equiv 1 \pmod{\varphi(n)}$ e così anche (1) è soddisfatta. Se si conosce E_k, ossia se si conoscono e e n, allora si è visto che è (probabilmente) computazionalmente intrattabile la determinazione di d, e quindi di D_k. Pertanto anche (3) è (probabilmente) soddisfatta.

Una volta che un sistema a chiave pubblica è configurato, ogni utente genera una chiave k e determina E_k e D_k. La funzione di cifratura E_k è resa pubblica mentre D_k è tenuta segreta. Per prevenire il problema degli impostori, si può utilizzare un'autorità fidata per la distribuzione e la verifica delle chiavi.

In un sistema simmetrico, Bob può essere sicuro che un messaggio che decifra con successo provenga da Alice (che in realtà può essere un gruppo di utenti autorizzati) o da qualcuno che ha la chiave di Alice. La chiave è stata data solo ad Alice e quindi nessun altro può produrre il testo cifrato. Tuttavia, Alice potrebbe negare di aver mandato il messaggio, poiché Bob potrebbe avere semplicemente cifrato il messaggio egli stesso. Quindi l'autenticazione è facile (Bob sa che il messaggio proviene da Alice, se non lo ha contraffatto egli stesso), mentre non è facile garantire la proprietà di non ripudiabilità (si veda il Paragrafo 1.2).

In un sistema a chiave pubblica, chiunque può cifrare un messaggio e inviarlo a Bob, che quindi non avrà idea da dove proviene. Certamente sarà in grado di dimostrare che proviene da Alice. Quindi sono necessari più passi per l'autenticazione e la non ripudiabilità. Tuttavia, tutto questo può essere facilmente realizzato nel modo seguente. Alice prende il messaggio m e calcola $E_{k_b}(D_{k_a}(m))$, dove k_a è la chiave di Alice e k_b è la chiave di Bob. Allora Bob può decifrarlo usando D_{k_b} per ottenere $D_{k_a}(m)$. Usa la funzione E_{k_a} disponibile pubblicamente per ottenere $E_{k_a}(D_{k_a}(m)) = m$. Bob sa che il messaggio deve provenire da Alice, poiché nessun altro avrebbe potuto calcolare $D_{k_a}(m)$. Per la stessa ragione, Alice non può negare di aver mandato il messaggio. Naturalmente, tutto questo nell'ipotesi in cui quasi tutti i "messaggi" casuali siano privi di senso, così che sia improbabile che una stringa casuale di simboli si decifri in un messaggio dotato di senso, a meno che la stringa non sia la cifratura di qualcosa di sensato.

Alcune versioni concrete di questi metodi di autenticazione verranno discusse nel Capitolo 9 dedicato alle firme digitali.

Per costruire un crittosistema a chiave pubblica si possono utilizzare funzioni unidirezionali con opportune proprietà. Sia $f(m)$ una funzione unidirezionale invertibile. Questo significa che $f(x)$ può essere calcolata facilmente, mentre (fissato y) è computazionalmente intrattabile trovare l'unico valore di x per cui $y = f(x)$. Supponiamo ora che $f(x)$ possieda una *trapdoor*, ossia che esista un modo semplice di risolvere $y = f(x)$ rispetto a x, ma solo con ulteriori informazioni note solo all'ideatore della funzione. Supponiamo, inoltre, che per chiunque che non sia l'ideatore della funzione, sia computazionalmente intrattabile determinare questa trapdoor. Se esiste una famiglia molto grande di funzioni unidirezionali con scappatoie, questa può essere usata per formare un crittosistema a chiave pubblica. Ogni utente genera una funzione appartenente alla famiglia in modo da essere l'unico a conoscere la scappatoia. La funzione dell'utente viene poi pubblicata come un algoritmo di cifratura pubblica. Quando Alice vuole mandare a Bob un messaggio m, prende la funzione $f_b(x)$ di Bob e calcola $y = f_b(m)$. Poi invia y a Bob. Poiché Bob conosce la trapdoor di $f_b(x)$, può risolvere $y = f_b(m)$ e quindi trovare m.

In RSA le funzioni $f(x) = x^e \pmod{n}$, per opportuni n ed e, formano la famiglia delle funzioni unidirezionali. La trapdoor segreta è la fattorizzazione di n. Nel sistema di ElGamal (Paragrafo 7.5), la funzione unidirezionale è l'elevamento a potenza modulo un primo e la trapdoor è data dalla conoscenza di un logaritmo discreto. In NTRU (Paragrafo 17.4), la trapdoor è una coppia di polinomi piccoli. Nel sistema di McEliece (Paragrafo 18.10), la trapdoor è data da un modo efficiente di trovare la più vicina parola di codice per certi codici correttori d'errore binari lineari.

6.8 Esercizi

1. Il testo cifrato 5859 è stato ottenuto mediante l'algoritmo RSA usando $n = 11413$ ed $e = 7467$. Usando la fattorizzazione $11413 = 101 \cdot 113$, trovare il testo in chiaro.

2. Sia $n = 55 = 5 \cdot 11$ il modulo e sia $e = 3$ l'esponente di cifratura RSA.

   (a) Trovare l'esponente d di decifrazione.

   (b) Sia MCD$(m,55) = 1$. Mostrare che se $c \equiv m^3 \pmod{55}$ è il testo cifrato, allora il testo in chiaro è $m \equiv c^d \pmod{55}$. Non usare il fatto che la decifrazione RSA funziona, perché è proprio ciò che si deve dimostrare in questo caso specifico.

3. Il testo cifrato 75 è stato ottenuto usando RSA con $n = 437$ ed $e = 3$. Se il testo in chiaro è 8 oppure 9, determinare il testo in chiaro senza fattorizzare n.

4. Se si cifrano i messaggi m calcolando $c \equiv m^3 \pmod{101}$, come si decifra? (In altre parole, si vuole un esponente di decifrazione d tale che $c^d \equiv m \pmod{101}$. Si noti che 101 è primo.)

5. Sia p un primo grande. Si cifra un messaggio x calcolando $y \equiv x^e \pmod{p}$ per un esponente di cifratura e opportunamente scelto. Come si fa a trovare un esponente di decifrazione d tale che $y^d \equiv x \pmod{p}$?

6. Sia n il prodotto di due primi grandi. Alice vuole mandare a Bob un messaggio m, dove $\text{MCD}(m,n) = 1$. Alice e Bob scelgono due interi a e b relativamente primi a $\varphi(n)$. Alice calcola $c \equiv m^a \pmod{n}$ e invia c a Bob. Bob calcola $d \equiv c^b \pmod{n}$ e invia d ad Alice. Poiché conosce a, Alice trova un a_1 tale che $aa_1 \equiv 1 \pmod{\varphi(n)}$. Poi calcola $e \equiv d^{a_1} \pmod{n}$ e invia e a Bob. Spiegare cosa deve fare ora Bob per ottenere m e mostrare che funziona. (*Osservazione:* in questo protocollo non c'è bisogno che i fattori primi di n siano tenuti segreti. Invece la sicurezza dipende dal tenere segreti a, b, a_1, b_1. Questo protocollo è una versione meno efficiente del protocollo a tre vie del Paragrafo 3.6).

7. L'ingenuo Nelson usa RSA per ricevere un singolo testo cifrato c, corrispondente al messaggio m. Il suo modulo pubblico è n e il suo esponente di cifratura pubblico è e. Poiché gli sembra uno spreco usare il suo sistema soltanto una volta, acconsente a decifrare qualunque testo cifrato gli venga inviato, purché non sia c, e a rimandare la risposta. La malvagia Eva gli invia il testo cifrato $2^e c \pmod{n}$. Mostrare come questo permette a Eva di trovare m.

8. Per aumentare la sicurezza, Bob sceglie n e due esponenti di cifratura e_1 ed e_2. Chiede ad Alice di cifrare il messaggio m calcolando dapprima $c_1 \equiv m^{e_1} \pmod{n}$ e poi cifrando c_1 in modo da ottenere $c_2 \equiv c_1^{e_2} \pmod{n}$. Alice allora invia c_2 a Bob. Questa cifratura doppia aumenta la sicurezza rispetto a una singola cifratura? Dire perché sì o perché no.

9. Siano p e q due primi dispari distinti e sia $n = pq$. Sia x un intero tale che $\text{MCD}(x,pq) = 1$.

(a) Mostrare che $x^{\frac{1}{2}\varphi(n)} \equiv 1 \pmod{p}$ e $x^{\frac{1}{2}\varphi(n)} \equiv 1 \pmod{q}$.

(b) Usare (a) per mostrare che $x^{\frac{1}{2}\varphi(n)} \equiv 1 \pmod{n}$.

(c) Usare (b) per mostrare che se $ed \equiv 1 \pmod{\frac{1}{2}\varphi(n)}$ allora $x^{ed} \equiv x \pmod{n}$. (Questo mostra che in RSA si può lavorare con $\frac{1}{2}\varphi(n)$ invece che con $\varphi(n)$. Infatti, grazie a un analogo ragionamento, si può anche usare il minimo comune multiplo di $p-1$ e $q-1$ al posto di $\varphi(n)$.)

10. Gli esponenti $e = 1$ ed $e = 2$ non dovrebbero essere usati in RSA. Perché?

11. Ci sono due utenti su una rete. Siano n_1 e n_2 i loro moduli RSA, con n_1 diverso da n_2. Se si sa che n_1 e n_2 non sono primi tra loro, come si possono violare i loro sistemi?

12. Si sta cercando di fattorizzare $n = 642401$. Si supponga di scoprire che

$$516107^2 \equiv 7 \pmod{n} \qquad \text{e} \qquad 187722^2 \equiv 2^2 \cdot 7 \pmod{n}.$$

Usare questa informazione per fattorizzare n.

13. Se si scopre che

$$880525^2 \equiv 2 \pmod{2288233}, \quad 2057202^2 \equiv 3 \pmod{2288233},$$

$$648581^2 \equiv 6 \pmod{2288233}, \quad 668676^2 \equiv 77 \pmod{2288233},$$

come si possono usare queste informazioni per fattorizzare 2288233? Dire quali sono i passi da compiere senza eseguire i calcoli numerici.

14. Siano p e q due primi grandi distinti. Spiegare come si può trovare un intero x tale che

$$x^2 \equiv 49 \pmod{pq}, \qquad x \not\equiv \pm 7 \pmod{pq}.$$

15. Sia n un numero dispari grande. Si calcola $2^{(n-1)/2} \equiv k \pmod{n}$, dove k è un intero con $k \not\equiv \pm 1 \pmod{n}$.

(a) Se $k^2 \not\equiv 1 \pmod{n}$, dire perché questo implica che n non è primo.

(b) Se $k^2 \equiv 1 \pmod{n}$, dire come si può usare questa informazione per fattorizzare n.

16. Alice e Bob hanno lo stesso modulo RSA dato da n. I loro esponenti di cifratura e_A e e_B sono primi tra loro. Carlo vuole inviare il messaggio m ad Alice e a Bob e quindi lo cifra in $c_A \equiv m^{e_A}$ e in $c_B \equiv m^{e_B} \pmod{n}$. Mostrare come Eva può trovare m se intercetta c_A e c_B.

17. Alice usa il metodo RSA nel modo seguente. Parte con un messaggio formato da varie lettere, utilizzando i valori $a = 1$, $b = 2$, ..., $z = 26$. Poi cifra ogni lettera separatamente. Per esempio, se il messaggio è *cat*, calcola $3^e \pmod{n}$, $1^e \pmod{n}$ e $20^e \pmod{n}$. Infine invia il messaggio cifrato a Bob. Spiegare in che modo Eva può trovare il messaggio senza fattorizzare n. In particolare, siano $n = 8881$ ed $e = 13$. Eva intercetta il messaggio

$$4461 \quad 794 \quad 2015 \quad 2015 \quad 3603.$$

Trovare il messaggio senza fattorizzare 8881.

18. Mostrare che se $x^2 \equiv y^2 \pmod{n}$ e $x \not\equiv \pm y \pmod{n}$, allora $\text{MCD}(x+y,n)$ è un fattore non banale di n.

19. Sia $n = pq$ il prodotto di due primi distinti.

(a) Sia m un multiplo di $\varphi(n)$. Mostrare che se $\text{MCD}(a,n) = 1$, allora $a^m \equiv 1 \pmod{p}$ e $\pmod{q}$.

(b) Sia m come nella parte (a) e sia a arbitrario (eventualmente con $\text{MCD}(a,n) \neq 1$). Mostrare che $a^{m+1} \equiv a \pmod{p}$ e $\pmod{q}$.

(c) Siano e e d gli esponenti di cifratura e di decifrazione RSA con modulo n. Mostrare che $a^{ed} \equiv a \pmod{n}$ per ogni a. Questo mostra che non è necessario richiedere che $\text{MCD}(a,n) = 1$ per usare RSA.

(d) Se p e q sono grandi, perché è probabile che MCD$(a,n) = 1$ per un a scelto a caso?

20. Sia $n = pqr$ il prodotto di tre primi distinti. Come lavora uno schema di tipo RSA in questo caso? In particolare, e e d che relazione soddisfano? *Osservazione:* non sembra ci sia alcun vantaggio a usare tre primi invece di due. I tempi di esecuzione di alcuni metodi di fattorizzazione dipendono dalla grandezza del fattore primo più piccolo. Quindi, se si usano tre primi, bisogna aumentare la grandezza di n per raggiungere lo stesso livello di sicurezza che si ottiene con due primi.

21. Siano $p = 7919$ e $q = 17389$. Sia $e = 66909025$. Un calcolo mostra che $e^2 \equiv 1 \pmod{(p-1)(q-1)}$. Alice decide di cifrare il messaggio $m = 12345$ usando RSA con modulo $n = pq$ ed esponente e. Poiché vuole che la cifratura sia molto sicura, cifra il testo cifrato usando ancora n ed e (in questo modo ha cifrato doppiamente il testo in chiaro originale). Qual è il testo cifrato finale che invia? Giustificare la risposta senza usare un calcolatore.

22. **(a)** Mostrare che se MCD$(e,24) = 1$, allora $e^2 \equiv 1 \pmod{24}$.

(b) Mostrare che se si usa $n = 35$ come modulo RSA, allora l'esponente di cifratura e è sempre uguale all'esponente di decifrazione d.

23. Il vostro avversario cifra un messaggio m usando RSA con $n = pq$ e con esponente di cifratura e. Questo dà il testo cifrato $c \equiv m^e \pmod{n}$. Una spia vi dice che, per questo messaggio, $m^{12345} \equiv 1 \pmod{n}$. Dire come si determina m, assumendo per semplicità che MCD$(12345,e) = 1$. Non è necessario conoscere p, q, $\varphi(n)$ o l'esponente di decifrazione d segreto. Tuttavia, si deve trovare un esponente di decifrazione che funzioni per questo particolare testo cifrato. Inoltre, si deve spiegare accuratamente perché la decifrazione proposta funziona (dicendo anche in che modo si usa l'informazione della spia).

24. Si supponga di usare RSA (con modulo $n = pq$ ed esponente di cifratura e) e di voler restringere i propri messaggi ai numeri m tali che $m^{1000} \equiv 1 \pmod{n}$.

(a) Mostrare che se d è tale che $de \equiv 1 \pmod{1000}$, allora d funziona da esponente di decifrazione per questi messaggi.

(b) Se p e q sono entrambi congruenti a 1 modulo 1000, dire quanti messaggi soddisfano la condizione $m^{1000} \equiv 1 \pmod{n}$. Si può assumere e usare il fatto che $m^{1000} \equiv 1 \pmod{r}$ ha 1000 soluzioni quando r è un primo congruente a 1 modulo 1000.

25. Supponiamo che $m^{270300} \equiv 1 \pmod{1113121}$ per ogni intero m tale che MCD$(m, 1113121) = 1$. Siano e e d tali che $ed \equiv 1 \pmod{270300}$ e sia m un messaggio tale che $0 < m < 1113121$ e MCD$(m, 1113121) = 1$. Cifrare m come $c \equiv m^e \pmod{1113121}$. Mostrare che $m \equiv c^d \pmod{1113121}$. Mostrare esplicitamente come si usa il fatto che $ed \equiv 1 \pmod{270300}$ e il fatto che $m^{270300} \equiv 1 \pmod{1113121}$. (*Osservazione:* il teorema di Eulero non vale, poiché $\varphi(1113121) \neq 270300$.)

26. L'azienda di cifratura di Bob produce due macchine A e B, entrambe implementazioni di RSA che usano lo stesso modulo $n = pq$ per due primi p e q non noti. Tutte e due le macchine usano lo stesso esponente di cifratura e. Ogni macchina riceve un messaggio m e fornisce in output un testo cifrato dato da $m^e \pmod{n}$. La macchina A fornisce sempre l'output corretto. Tuttavia, la macchina B, a causa di errori di implementazione e nell'hardware, fornisce sempre in output un testo cifrato $c \pmod{n}$ tale che $c \equiv m^e \pmod{p}$ e $c \equiv m^e + 1 \pmod{q}$. Come si possono usare le due macchine A e B per trovare p e q? (Per una discussione su come possa insorgere tale situazione, si veda il Problema al calcolatore 11.)

27. **(a)** Alice vuole inviare un messaggio m corto. Per prevenire l'attacco del messaggio corto del Paragrafo 6.2, dice a Bob che ha aggiunto 100 zeri alla fine del suo testo in chiaro, ossia che sta usando $m_1 = 10^{100}m$ come testo in chiaro e che sta inviando $c_1 \equiv m_1^e$. Se Eva sa che Alice sta facendo tutto questo, come può modificare l'attacco del testo in chiaro corto ed eventualmente trovare il testo in chiaro?

(b) Alice si rende conto che il metodo della parte (a) non fornisce sicurezza. Quindi decide di rendere il testo in chiaro più lungo ripetendolo due volte: $m||m$ (dove $x||y$ significa che si scrivono le cifre di x seguite dalle cifre di y in modo da ottenere un numero più lungo). Se Eva sa che Alice ha preso questa precauzione, come può modificare l'attacco del testo in chiaro corto ed eventualmente trovare il testo in chiaro? Si supponga che Eva conosca la lunghezza di m. (*Suggerimento:* esprimere $m||m$ come un multiplo di m.)

28. Questo esercizio fornisce alcuni dettagli sul modo in cui il crivello quadratico ottiene le relazioni che sono usate per fattorizzare un intero dispari n grande. Sia s il più piccolo intero maggiore della radice quadrata di n e sia $f(x) = (x+s)^2 - n$. Supponiamo che la base di fattorizzazione $\mathcal{B}$ sia formata dai primi minori o uguali a un qualche limite B. Vogliamo trovare i quadrati che sono congruenti modulo n a un prodotto di primi in $\mathcal{B}$. Un modo per farlo è di trovare i valori di $f(x)$ che sono prodotti di primi in $\mathcal{B}$. Cercheremo in un intervallo $0 \leq x \leq A$, per qualche A.

(a) Supponendo che $0 \leq x < (\sqrt{2}-1)\sqrt{n} - 1$, mostrare che $0 \leq f(x) < n$, ossia che $f(x) \pmod{n}$ è semplicemente $f(x)$. (*Suggerimento:* mostrare che $x + s < x + \sqrt{n} + 1 < \sqrt{2n}$.) Da qui in poi, assumeremo che $A < (\sqrt{2}-1)\sqrt{n} - 1$, in modo da considerare valori di x tali che $f(x) < n$.

(b) Sia p un primo in $\mathcal{B}$. Mostrare che se esiste un intero x per cui $f(x)$ è divisibile per p, allora n è un quadrato modulo p. Questo mostra che possiamo scartare i primi in $\mathcal{B}$ in corrispondenza dei quali n non è un quadrato modulo p. Da qui in poi, assumeremo che tali primi siano stati scartati.

(c) Sia $p \in \mathcal{B}$ tale che n è un quadrato modulo p. Mostrare che se p è dispari e $p \nmid n$, allora esistono esattamente due valori $x_{p,1}$ e $x_{p,2}$ di x modulo p per cui $f(x) \equiv 0 \pmod{p}$. (*Nota:* nell'improbabile caso in cui $p|n$, si trova un fattore, che è il nostro obiettivo.)

(d) Per ogni x con $0 \le x \le A$, inizializzare un registro con il valore $\log f(x)$. Per ogni primo $p \in \mathcal{B}$, sottrarre $\log p$ dai registri degli x per i quali $x \equiv x_{p,1}$ o $x_{p,2} \pmod p$. (*Osservazione:* questo è la parte di setacciamento in cui si usa il "crivello".) Mostrare che se $f(x)$ (con $0 \le x \le A$) è un prodotto di primi distinti in $\mathcal{B}$, allora alla fine di questo processo il registro per x diventa 0.

(e) Spiegare perché se $f(x)$ (con $0 \le x \le A$) è un prodotto di primi (non necessariamente distinti) in $\mathcal{B}$ allora è probabile che il risultato finale per il registro per x sia piccolo (in confronto al registro per un x tale che $f(x)$ ha un fattore primo non in $\mathcal{B}$).

(f) Perché la procedura della parte (d) è più veloce della divisione di ogni $f(x)$ per ogni elemento di $\mathcal{B}$ e perché l'algoritmo sottrae $\log p$ invece di dividere $f(x)$ per p?

In pratica, il crivello prende in considerazione anche le soluzioni di $f(x) \equiv 0$ modulo qualche potenza di primi piccoli in $\mathcal{B}$. Una volta che il lavoro del crivello è terminato, i registri con valori piccoli vengono controllati per vedere quali di essi corrispondono a una $f(x)$ che si spezza nel prodotto di primi appartenenti a $\mathcal{B}$. Queste danno le relazioni "quadrato $\equiv$ prodotto di primi in $\mathcal{B}$ modulo n" che sono usate per fattorizzare n.

6.9 Problemi al calcolatore

Nota: *molti dei numeri che compaiono nei seguenti problemi sono troppo grandi per MATLAB® senza l'assistenza del motore simbolico Maple®.*

1. Un amico di Paul Revere è di guardia su una torre e gli dice che gli invierà il messaggio *one* se gli Inglesi arriveranno per terra e *two* se arriveranno per mare.[1] Poiché sanno che RSA sarà inventato nell'area di Boston, decidono che il messaggio dovrà essere cifrato usando RSA con $n = 712446816787$ ed $e = 6551$. Paul Revere riceve il testo cifrato 273095689186. Qual'era il testo in chiaro?

2. In un crittosistema RSA, si conoscono $n = 718548065973745507$, $e = 3449$ e $d = 543546506135745129$. Fattorizzare n.

3. Si scelgano due primi p e q di 30 cifre e un esponente di cifratura e. Cifrare ognuno dei testi in chiaro *cat, bat, hat, encyclopedia, antidisestablishmentarianism.* Guardando i testi cifrati si può dire che i primi tre testi in chiaro differiscono solo per una lettera o che gli ultimi due testi in chiaro sono molto più lunghi dei primi tre?

4. Fattorizzare 618240007109027021 mediante il metodo $p-1$.

[1] L'Autore colloca scherzosamente l'esercizio nel contesto del Boston Tea Party, avvenuto il 16 dicembre 1773 nel porto di Boston e visto da molti come la scintilla che diede inizio alla Rivoluzione Americana. (*N.d.Rev.*)

5. Fattorizzare 883488458709081464637245989037741896276690 7 mediante il metodo $p-1$ (si veda il file *n1*).

6. Fattorizzare $n = 537069139875071$, sapendo che

$$859753244431662 \equiv 4624361062612 \pmod n.$$

7. Sia $n = 985739879 \cdot 1388749507$. Trovare x e y con $x^2 \equiv y^2 \pmod n$ ma $x \not\equiv \pm y \pmod n$.

8. **(a)** Se si sa che

$$33335^2 \equiv 670705093^2 \pmod{670726081},$$

usare questa informazione per fattorizzare 670726081.

(b) Se si sa che $3^2 \equiv 670726078^2 \pmod{670726081}$, perché questa informazione non aiuta a fattorizzare 670726081?

9. Se si sa che

$$\begin{aligned} 2^{958230} &\equiv 1488665 \pmod{3837523} \\ 2^{1916460} &\equiv 1 \pmod{3837523}, \end{aligned}$$

come si possono usare queste informazioni per fattorizzare 3837523? Osservare che l'esponente 1916460 è il doppio dell'esponente 958230.

10. **(a)** Se i primi p e q usati nell'algoritmo RSA sono primi consecutivi, come si può fattorizzare $n = pq$?

(b) Il testo cifrato 10787770728 è stato messo in cifra usando $e = 113$ e $n = 10993522499$. I fattori p e q di n sono stati scelti in modo che $q - p = 2$. Decifrare il messaggio.

(c) Il testo cifrato c è stato cifrato modulo n usando l'esponente e, dove

$$n = 152415787501905985701881832150835089037858868621211004433$$

$$e = 9007$$

$$c = 141077461765569500241199505617854673388398574333341423525.$$

I fattori primi p e q di n sono primi consecutivi. Decifrare il messaggio. (Per il numero n si veda il file *naive* e per c si veda il file *cnaive*.) (*Nota:* in Mathematica®, il comando **Round[N[Sqrt[n],50]]** valuta la radice quadrata di n fino a 50 cifre decimali e poi arrotonda all'intero più vicino. In Maple, prima si usa il comando **Digits:=50** per ottenere un'accuratezza fino a 50 cifre, poi si usa il comando **round(sqrt(n*1.))** per scrivere n in forma decimale, si prende la sua radice quadrata e si arrotonda all'intero più vicino.)

11. Siano $p = 123456791$, $q = 987654323$ e $e = 127$. Sia $m = 14152019010605$ il messaggio.

(a) Calcolare $m^e \pmod p$ e $m^e \pmod q$. Usare poi il teorema cinese del resto per metterli insieme e ottenere $c \equiv m^e \pmod{pq}$.

(b) Si cambi una cifra di $m^e \pmod p$ (questo potrebbe essere causato, per esempio, da una radiazione). Poi lo si combini con $m^e \pmod q$ per ottenere un valore sbagliato f di $m^e \pmod{pq}$. Calcolare $\text{MCD}(c - f, pq)$. Perché questo permette di fattorizzare pq?

Il metodo in (a) per calcolare $m^e \pmod{pq}$ è attraente poiché non richiede una precisione aritmetica inferiore che lavorare direttamente modulo pq. Tuttavia, come mostra la parte (b), se un attaccante può causare un errore in qualche bit, allora pq può essere fattorizzato.

12. Siano $p = 76543692179$, $q = 343434343453$ ed $e = 457$. Il testo cifrato $c \equiv m^e \pmod{pq}$ viene trasmesso, ma durante la trasmissione si verifica un errore. Il testo cifrato che viene ricevuto è 2304329328016936947195. Il ricevente è in grado di determinare che le cifre ricevute sono corrette, ma che manca l'ultima cifra. Determinare la cifra mancante e decifrare il messaggio.

13. Controllare la primalità di 38200901201 usando il test di Miller-Rabin con $a = 2$ e poi con $a = 3$. Il primo test dice che 38200901201 è probabilmente primo, mentre il secondo test dice che è composto. Un numero composto come 38200901201 che supera il test di Miller-Rabin per un numero a è detto **a-pseudoprimo forte**.

14. Ci sono tre utenti con moduli n_1, n_2, n_3 primi tra loro a coppie. I loro esponenti di cifratura sono tutti $e = 3$. Ad ognuno di essi viene inviato lo stesso messaggio m. Si intercettano i testi cifrati $c_i \equiv m^3 \pmod{n_i}$ per $i = 1, 2, 3$.

(a) Mostrare che $0 \leq m^3 < n_1 n_2 n_3$.

(b) Usare il teorema cinese del resto per trovare m^3 (come intero esatto e *non* solo come $m^3 \pmod{n_1 n_2 n_3}$) e quindi m (senza fattorizzare).

(c) Siano

$$n_1 = 2469247531693,\ n_2 = 11111502225583,\ n_3 = 44444222221411$$

e siano

$$359335245251, \qquad 10436363975495, \qquad 5135984059593$$

i corrispondenti testi cifrati, ottenuti tutti usando $e = 3$. Trovare il messaggio.

Capitolo 7

Logaritmo discreto

7.1 Logaritmo discreto

Come si è visto con l'algoritmo RSA, si possono ideare crittositemi utili sfruttando il problema della difficile fattorizzazione di un numero grande. Anche il problema del logaritmo discreto, un altro problema della teoria dei numeri, ha analoghe applicazioni. Fissato un primo p, siano α e β due interi non nulli modulo p tali che

$$\beta \equiv \alpha^x \pmod p.$$

Il problema di trovare x è chiamato il **problema del logaritmo discreto**. Se n è il più piccolo intero positivo per cui $\alpha^n \equiv 1 \pmod p$, si può assumere $0 \leq x < n$ e scrivere

$$x = L_\alpha(\beta).$$

È chiamato il logaritmo discreto di β rispetto ad α (in questa notazione il primo p viene sottinteso).

Per esempio, siano $p = 11$ e $\alpha = 2$. Poiché $2^6 \equiv 9 \pmod{11}$, si ha $L_2(9) = 6$. Naturalmente, essendo $2^6 \equiv 2^{16} \equiv 2^{26} \equiv 9 \pmod{11}$, si può considerare come logaritmo discreto ognuno dei numeri 6, 16, 26. Tuttavia si sceglie il più piccolo valore non negativo, ossia 6. Si noti che in questo caso il logaritmo discreto poteva essere definito come la classe di congruenza 6 modulo 10. Anche se questo sarebbe stato, in un certo senso, più naturale, esistono delle applicazioni in cui è più conveniente avere un numero e non una classe di congruenza.

Spesso α è una radice primitiva modulo p. In questo modo ogni β è una potenza di $\alpha \pmod p$. Se α non è una radice primitiva, allora il logaritmo discreto non sarà definito per certi valori di β.

Dato un primo p, in molti casi è abbastanza facile trovare una radice primitiva (si veda l'Esercizio 21 del Capitolo 3).