(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).

Il logaritmo discreto si comporta per certi aspetti come l'usuale logaritmo. In particolare, se α è una radice primitiva modulo p, allora

$$L_\alpha(\beta_1\beta_2) \equiv L_\alpha(\beta_1) + L_\alpha(\beta_2) \pmod{p-1}$$

(si veda l'Esercizio 5).
Quando p è piccolo, è facile calcolare i logaritmi discreti mediante una ricerca esaustiva tra tutti i possibili esponenti. Tuttavia, quando p è grande questo non è più fattibile. Anche se più avanti vedremo alcuni modi per attaccare il problema del logaritmo discreto, in generale si pensa che i logaritmi discreti siano difficili da calcolare. Questa supposizione è alla base di molti crittosistemi.
I numeri primi più grandi per cui è possibile calcolare i logaritmi discreti sono di grandezza paragonabile a quella degli interi più grandi che possono essere fattorizzati (in entrambi i casi ci si riferisce a calcoli che possano funzionare per numeri arbitrari di queste grandezze; tuttavia, per scelte particolari di interi si possono utilizzare tecniche particolari e si può quindi avere la possibilità di calcolare il logaritmo discreto e la fattorizzazione di numeri speciali molto più grandi). Nel 2001, è stato calcolato un logaritmo discreto per un primo di 120 cifre, un record per quei tempi. Il record di fattorizzazione fino a quell'epoca era di 155 cifre.
Una funzione $f(x)$ è detta **funzione unidirezionale** se $f(x)$ è semplice da calcolare, ma, dato y, è computazionalmente infattibile calcolare un x tale che $f(x) = y$. L'elevamento a potenza modulare è probabilmente un esempio di funzione di questo tipo. È facile calcolare $\alpha^x \pmod{p}$, ma risolvere $\alpha^x \equiv \beta$ rispetto a x è probabilmente difficile. Anche moltiplicare numeri primi grandi può essere considerato come una (probabile) funzione unidirezionale: è facile moltiplicare numeri primi, ma difficile fattorizzare il risultato per recuperare i fattori primi. Le funzioni unidirezionali hanno molti usi crittografici.

7.2 Calcolo del logaritmo discreto

In questo paragrafo, verranno presentati alcuni metodi per calcolare il logaritmo discreto. Un altro metodo utile e importante, ossia l'attacco del compleanno, verrà discusso nel Paragrafo 8.4.
Per semplicità, sia α una radice primitiva modulo p, in modo che $p-1$ sia il più piccolo esponente positivo n per cui $\alpha^n \equiv 1 \pmod{p}$. Questo implica che

$$\alpha^{m_1} \equiv \alpha^{m_2} \pmod{p} \iff m_1 \equiv m_2 \pmod{p-1}.$$

Si assuma che

$$\beta \equiv \alpha^x, \quad 0 \le x < p-1.$$

Il problema è quello di determinare x.
Innanzitutto, è facile determinare $x \pmod 2$. Poiché

$$\left(\alpha^{(p-1)/2}\right)^2 \equiv \alpha^{p-1} \equiv 1 \pmod{p},$$

si ha $\alpha^{(p-1)/2} \equiv \pm 1 \pmod{p}$ (si veda l'Esercizio 8 del Capitolo 3). Tuttavia, essendo $p-1$ il più piccolo esponente che dà $+1$, si deve avere

$$\alpha^{(p-1)/2} \equiv -1 \pmod{p}.$$

Elevando entrambi i membri della congruenza $\beta \equiv \alpha^x \pmod{p}$ alla potenza $(p-1)/2$, si ha

$$\beta^{(p-1)/2} \equiv \alpha^{x(p-1)/2} \equiv (-1)^x \pmod{p}.$$

Quindi, se $\beta^{(p-1)/2} \equiv +1$, allora x è pari; altrimenti, x è dispari.

Esempio. Si supponga di dover risolvere $2^x \equiv 9 \pmod{11}$. Poiché

$$\beta^{(p-1)/2} \equiv 9^5 \equiv 1 \pmod{11},$$

x deve essere pari. Infatti, come visto in precedenza, $x = 6$. ■

7.2.1 L'algoritmo di Pohlig-Hellman

L'idea precedente è stata estesa da Pohlig e da Hellman per avere un algoritmo che calcoli i logaritmi discreti quando $p-1$ ha solo fattori primi piccoli. Sia

$$p - 1 = \prod_i q_i^{r_i}$$

la scomposizione di $p-1$ in fattori primi. Se $L_\alpha(\beta) \pmod{q^r}$ può essere calcolato per ogni fattore $q_i^{r_i}$, allora il logaritmo discreto potrà essere determinato mettendo insieme i vari risultati parziali mediante il teorema cinese del resto.
Sia q^r uno dei fattori di $p-1$. Sia

$$x = x_0 + x_1 q + x_2 q^2 + \cdots \qquad \text{con} \quad 0 \le x_i \le q-1.$$

I coefficienti $x_0, x_1, \ldots x_{r-1}$ verranno determinati uno dopo l'altro nel modo seguente. Si ha

$$\begin{aligned} x\left(\frac{p-1}{q}\right) &= x_0\left(\frac{p-1}{q}\right) + (p-1)(x_1 + x_2 q + x_3 q^2 + \cdots) \\ &= x_0\left(\frac{p-1}{q}\right) + (p-1)n, \end{aligned}$$

dove n è un intero. Elevando entrambi i membri di $\beta \equiv \alpha^x$ alla potenza $(p-1)/q$, si ottiene

$$\beta^{(p-1)/q} \equiv \alpha^{x(p-1)/q} \equiv \alpha^{x_0(p-1)/q}(\alpha^{p-1})^n \equiv \alpha^{x_0(p-1)/q} \pmod{p}.$$

L'ultima congruenza è una conseguenza del teorema di Fermat: $\alpha^{p-1} \equiv 1 \pmod{p}$. Per trovare x_0, basta cercare tra le potenze

$$\alpha^{k(p-1)/q} \pmod{p}, \quad k = 0,1,2,\ldots,q-1,$$

quella in corrispondenza della quale si ha $\beta^{(p-1)/q}$. Allora $x_0 = k$. Tale k è unico poiché $\alpha^{m_1} \equiv \alpha^{m_2}$ se e solo se $m_1 \equiv m_2 \pmod{p-1}$ e poiché gli esponenti $k(p-1)/q$ sono tutti distinti modulo $p-1$.

Estendendo questa idea si possono ottenere i restanti coefficienti. Si assuma che $q^2|p-1$. Sia

$$\beta_1 \equiv \beta\alpha^{-x_0} \equiv \alpha^{q(x_1+x_2q+\cdots)} \pmod{p}.$$

Elevando entrambi i membri alla potenza $(p-1)/q^2$, si ottiene

$$\begin{aligned}\beta_1^{(p-1)/q^2} &\equiv \alpha^{(p-1)(x_1+x_2q+\cdots)/q} \\ &\equiv \alpha^{x_1(p-1)/q}(\alpha^{p-1})^{x_2+x_3q+\cdots} \\ &\equiv \alpha^{x_1(p-1)/q} \pmod{p}\end{aligned}$$

dove l'ultima congruenza è conseguenza del teorema di Fermat. Non si può calcolare $\beta_1^{(p-1)/q^2}$ come $(\beta_1^{p-1})^{1/q^2}$, poiché gli esponenti frazionari provocano dei problemi. Si noti che tutti gli esponenti usati finora sono interi. Per trovare x_1, basta cercare tra le potenze

$$\alpha^{k(p-1)/q} \pmod{p}, \quad k = 0,1,2,\ldots,q-1,$$

quella per cui si ha $\beta_1^{(p-1)/q^2}$. Allora $x_1 = k$.
Se $q^3|p-1$, sia $\beta_2 \equiv \beta_1\alpha^{-x_1q}$. Elevando entrambi i membri alla potenza $(p-1)/q^3$, si ottiene x_2. Si può continuare in questo modo finché non si trova che q^{r+1} non divide $p-1$. Poiché non si possono usare gli esponenti frazionari, bisogna fermarsi. Ma, avendo determinato $x_0, x_1, \ldots, x_{r-1}$, si ha x modulo q^r.
Ripetendo questa procedura per tutti i fattori primi di $p-1$, si ottiene x modulo $q_i^{r_i}$ per tutti gli i. Il teorema cinese del resto permette di metterli insieme in una congruenza per x modulo $p-1$. Poiché $0 \le x < p-1$, questo determina x.

Esempio. Dati $p = 41$, $\alpha = 7$ e $\beta = 12$, si consideri l'equazione

$$7^x \equiv 12 \pmod{41}.$$

Si noti che

$$p - 1 = 41 - 1 = 2^3 \cdot 5.$$

Come prima cosa, posto $q = 2$, bisogna trovare $x \mod 2^3$. Sia $x \equiv x_0 + 2x_1 + 4x_2 \pmod 8$. Allora

$$\beta^{(p-1)/2} \equiv 12^{20} \equiv 40 \equiv -1 \pmod{41}$$
$$\alpha^{(p-1)/2} \equiv 7^{20} \equiv -1 \pmod{41}.$$

Poiché

$$\beta^{(p-1)/2} \equiv (\alpha^{(p-1)/2})^{x_0} \pmod{41},$$

si ha $x_0 = 1$. Quindi si ha

$$\beta_1 \equiv \beta\alpha^{-x_0} \equiv 12 \cdot 7^{-1} \equiv 31 \pmod{41}.$$

Ancora

$$\beta_1^{(p-1)/2^2} \equiv 31^{10} \equiv 1 \pmod{41}.$$

Poiché

$$\beta_1^{(p-1)/2^2} \equiv (\alpha^{(p-1)/2})^{x_1} \pmod{41},$$

si ha $x_1 = 0$. Infine, si ha

$$\beta_2 \equiv \beta_1\alpha^{-2x_1} \equiv 31 \cdot 7^0 \equiv 31 \pmod{41}$$

e

$$\beta_2^{(p-1)/q^3} \equiv 31^5 \equiv -1 \equiv (\alpha^{(p-1)/2})^{x_2} \pmod{41}.$$

Quindi $x_2 = 1$. Pertanto

$$x \equiv x_0 + 2x_1 + 4x_2 \equiv 1 + 4 \equiv 5 \pmod 8.$$

Ora, posto $q = 5$, bisogna trovare x modulo 5. Si ha

$$\beta^{(p-1)/5} \equiv 12^8 \equiv 18 \pmod{41}$$

e

$$\alpha^{(p-1)/q} \equiv 7^8 \equiv 37 \pmod{41}.$$

Provando i possibili valori di k, si ha

$$37^0 \equiv 1, \quad 37^1 \equiv 37, \quad 37^2 \equiv 16, \quad 37^3 \equiv 18, \quad 37^4 \equiv 10 \pmod{41}.$$

Pertanto la risposta è 37^3 e quindi $x \equiv 3 \pmod 5$.
Poiché $x \equiv 5 \pmod 8$ e $x \equiv 3 \pmod 5$, mettendo insieme queste due congruenze si ottiene $x \equiv 13 \pmod{40}$ e quindi $x = 13$. Un veloce calcolo mostra che effettivamente si ha $7^{13} \equiv 12 \pmod{41}$, come desiderato. ■

Finché i primi q coinvolti nell'algoritmo precedente restano ragionevolmente piccoli, i calcoli possono essere svolti rapidamente. Tuttavia, quando q è grande, calcolare i numeri $\alpha^{k(p-1)/q}$ per $k = 0,1,2,\ldots,q-1$ diventa impraticabile e pertanto l'algoritmo di fatto non è più utilizzabile. Questo significa che se si vuole che il logaritmo discreto sia computazionalmente intrattabile, si dovrebbe essere sicuri che $p-1$ abbia un fattore primo grande.
Si noti che se anche $p-1 = tq$ ha un fattore primo q grande, l'algoritmo può determinare i logaritmi discreti modulo t se t è composto da fattori primi piccoli. Per questa ragione, spesso β è scelto in modo da essere una potenza di α^t. Allora il logaritmo discreto è automaticamente 0 modulo t e quindi il logaritmo discreto nasconde solo informazioni modulo q che l'algoritmo non può trovare. Se il logaritmo discreto x rappresenta un segreto (o meglio, t volte un segreto), questo significa che un attaccante non ottiene informazioni parziali dallà determinazione di x modulo t, poiché non ci sono informazioni nascoste in questo modo. Questa idea è usata nell'algoritmo di firma digitale DSA, che verrà discusso nel Capitolo 9.

7.2.2 Baby Step, Giant Step

Eva, volendo trovare un x tale che $\alpha^x \equiv \beta \pmod{p}$, procede nel modo seguente. Come prima cosa, sceglie un intero N con $N^2 \geq p-1$, per esempio $N = \lfloor \sqrt{p-1} \rfloor + 1$. Poi compone le due liste seguenti.

1. $\alpha^j \pmod{p}$ per $0 \leq j < N$.
2. $\beta\alpha^{-Nk} \pmod{p}$ per $0 \leq k < N$.

Infine, cerca una coincidenza tra le due liste. Se ne trova una, allora

$$\alpha^j \equiv \beta\alpha^{-Nk}, \qquad \text{ossia} \qquad \alpha^{j+Nk} \equiv \beta,$$

e quindi $x = j + Nk$ risolve il problema del logaritmo discreto.
Ma perché dovrebbe esserci una coincidenza tra le due liste? Poiché $0 \leq x < p-1 \leq N^2$, si può scrivere x in base N come $x = x_0 + Nx_1$ con $0 \leq x_0, x_1 < N$. Infatti, $x_1 = \lfloor x/N \rfloor$ e $x_0 = x - Nx_1$. Quindi

$$j = x_0, \quad k = x_1$$

fornisce la coincidenza richiesta.
La lista α^j per $j = 0, 1, 2, \ldots$ è l'insieme dei "Baby Steps" (passi da bambino) poiché gli elementi della lista si ottengono moltiplicando per α, mentre i "Giant Steps" (passi da gigante) nella seconda lista si ottengono moltiplicando per α^{-N}. Naturalmente non è necessario calcolare tutti gli elementi della seconda lista. Ogni elemento, appena è calcolato, può essere confrontato con la prima lista. Quando si trova una coincidenza, il calcolo termina.
Il numero di passi in questo algoritmo è proporzionale a $N \approx \sqrt{p}$ e richiede di memorizzare circa N numeri. Quindi il metodo funziona per primi p fino a 10^{20}, o anche un poco più grandi, ma è intrattabile per p molto grandi.

7.2.3 Index Calculus

L'idea è simile a quella del metodo di fattorizzazione del crivello quadratico. Anche qui, si cerca di risolvere $\beta \equiv \alpha^x \pmod{p}$, dove p è un primo grande e α è una radice primitiva.
Innanzitutto, c'è una fase di precalcolo. Sia B una limitazione e siano $p_1, p_2, \ldots, p_m$ i primi minori di B. Questo insieme di primi è chiamato **base di fattorizzazione**. Si calcola $\alpha^k \pmod{p}$ per vari valori di k. Si prova a scrivere ognuno di questi numeri come prodotto di primi minori di B. Se non ci si riesce, α^k viene scartato. Se invece $\alpha^k \equiv \prod p_i^{a_i} \pmod{p}$, allora

$$k \equiv \sum a_i L_\alpha(p_i) \pmod{p-1}.$$

Quando si ottiene un numero sufficiente di tali relazioni, il corrispondente sistema può essere risolto in modo da ottenere $L_\alpha(p_i)$ per ogni i.
Ora, per interi r scelti a caso, si calcola $\beta\alpha^r \pmod{p}$. Si prova a scrivere ognuno di questi numeri come prodotto di primi minori di B. In caso di successo, si ha $\beta\alpha^r \equiv \prod p_i^{b_i} \pmod{p}$, ossia

$$L_\alpha(\beta) \equiv -r + \sum b_i L_\alpha(p_i) \pmod{p-1}.$$

Questo algoritmo può essere utilizzato in pratica se p non è troppo grande. Questo significa che, affinché il problema del logaritmo discreto sia intrattabile, p deve essere di almeno 200 cifre.

Esempio. Siano $p = 131$ e $\alpha = 2$. Scelto $B = 10$, si lavora con i primi 2, 3, 5, 7. Si ha

$$\begin{aligned} 2^1 &\equiv 2 \pmod{131} \\ 2^8 &\equiv 5^3 \pmod{131} \\ 2^{12} &\equiv 5 \cdot 7 \pmod{131} \\ 2^{14} &\equiv 3^2 \pmod{131} \\ 2^{34} &\equiv 3 \cdot 5^2 \pmod{131}. \end{aligned}$$

Quindi

$$\begin{aligned} 1 &\equiv L_2(2) \pmod{130} \\ 8 &\equiv 3L_2(5) \pmod{130} \\ 12 &\equiv L_2(5) + L_2(7) \pmod{130} \\ 14 &\equiv 2L_2(3) \pmod{130} \\ 34 &\equiv L_2(3) + 2L_2(5) \pmod{130}. \end{aligned}$$

Dalla seconda congruenza si ha $L_2(5) \equiv 46 \pmod{130}$. Sostituendo nella terza congruenza si ottiene $L_2(7) \equiv -34 \equiv 96 \pmod{130}$. Dalla sola quarta congruenza si può ottenere il valore di $L_2(3) \pmod{65}$ poiché $\text{MCD}(2,130) \neq 1$. Si hanno così due possibilità per $L_2(3) \pmod{130}$. A questo punto possiamo controllare direttamente quale di esse è quella giusta. Oppure si può usare la quinta congruenza e ottenere $L_2(3) \equiv 72 \pmod{130}$. Questo conclude la fase di precalcolo.
Si supponga ora di dover trovare $L_2(37)$. Provando con qualche esponente scelto a caso si ottiene $37 \cdot 2^{43} \equiv 3 \cdot 5 \cdot 7 \pmod{131}$ e quindi

$$L_2(37) \equiv -43 + L_2(3) + L_2(5) + L_2(7) \equiv 41 \pmod{130}.$$

In conclusione $L_2(37) = 41$. ■

Naturalmente, una volta che la base di fattorizzazione è stata precalcolata, si può riusare per calcolare vari logaritmi discreti per lo stesso primo p.

7.2.4 Calcolo dei logaritmi discreti modulo 4

Quando $p \equiv 1 \pmod{4}$, l'algoritmo di Pohlig-Hellman calcola i logaritmi discreti modulo 4 molto rapidamente. Ma cosa accade quando $p \equiv 3 \pmod{4}$? L'algoritmo di Pohlig-Hellman non funziona, poiché richiederebbe elevamenti alla potenza $(p-1)/4$, portando all'ambiguità di un esponente frazionario. Il fatto sorprendente è che se si ha un algoritmo che calcola rapidamente i logaritmi discreti modulo 4 per un primo

$p \equiv 3 \pmod 4$, allora lo si può utilizzare per calcolare rapidamente logaritmi discreti modulo p. Quindi, è improbabile che tale algoritmo esista.
C'è una ragione filosofica per cui non bisogna aspettarsi l'esistenza di tale algoritmo. Da un punto di vista naturale, il logaritmo discreto dovrebbe essere considerato come un numero modulo $p-1$. Quindi si dovrebbe essere in grado di ottenere informazioni sul logaritmo discreto solo modulo la potenza di 2 che appare in $p-1$. Quando $p \equiv 3 \pmod 4$, questo significa che porsi domande sui logaritmi discreti modulo 4 è qualcosa di innaturale. Il problema è possibile solo perché il logaritmo discreto è stato normalizzato a un intero compreso tra 0 e $p-2$. Per esempio, $2^6 \equiv 2^{16} \equiv 9 \pmod{11}$. In questo caso, $L_2(9)$ è stato definito uguale a 6. Se si fosse permesso anche il valore 16, si avrebbero avuti due valori per $L_2(9)$, ossia 6 e 16, che non sono congruenti modulo 4. Quindi, da questo punto di vista, non avrebbe nemmeno avuto senso cercare $L_2(9)$ modulo 4.
Si avrà bisogno del seguente lemma, che è simile al metodo per calcolare le radici quadrate modulo un primo $p \equiv 3 \pmod 4$ (si veda il Paragrafo 3.9).

Lemma. *Sia $p \equiv 3 \pmod 4$ un primo, sia $r \geq 2$ e sia y un intero. Se α e γ sono due numeri non nulli modulo p tali che $\gamma \equiv \alpha^{2^r y} \pmod p$, allora*

$$\gamma^{(p+1)/4} \equiv \alpha^{2^{r-1}y} \pmod p.$$

Dimostrazione. Si ha

$$\gamma^{(p+1)/4} \equiv \alpha^{(p+1)2^{r-2}y} \equiv \alpha^{2^{r-1}y}(\alpha^{p-1})^{2^{r-2}y} \equiv \alpha^{2^{r-1}y} \pmod p$$

dove la congruenza finale è conseguenza del teorema di Fermat. □

Fissato il primo $p \equiv 3 \pmod 4$, sia α una radice primitiva. Si assuma di avere una macchina che, dato in ingresso β, restituisce in uscita $L_\alpha(\beta) \mod 4$. Come si è visto in precedenza, è facile calcolare $L_\alpha(\beta) \mod 2$. Pertanto la nuova informazione fornita dalla macchina è in realtà solo il secondo bit del logaritmo discreto.
Ora si assuma $\alpha^x \equiv \beta \pmod p$. Sia $x = x_0 + 2x_1 + 4x_2 + \cdots + 2^n x_n$ la rappresentazione binaria di x. Usando la macchina $L_\alpha(\beta) \pmod 4$, si determinano x_0 e x_1. Supponendo di avere determinato $x_0, x_1, \ldots, x_{r-1}$ con $r \geq 2$, sia

$$\beta_r \equiv \beta\alpha^{-(x_0+\cdots+2^{r-1}x_{r-1})} \equiv \alpha^{2^r(x_r+2x_{r+1}+\cdots)}.$$

Usando il lemma $r-1$ volte, si trova

$$\beta_r^{((p+1)/4)^{r-1}} \equiv \alpha^{2(x_r+2x_{r+1}+\cdots)} \pmod p.$$

Applicando la macchina $L_\alpha \pmod 4$ a questa equazione si ottiene il valore di x_r. Procedendo allo stesso modo, si ottengono tutti i valori $x_0, x_1, \ldots, x_n$. In questo modo si determina x, come desiderato.
È possibile rendere questo algoritmo più efficiente. Si veda, per esempio, [Stinson1, pagina 175].
In conclusione, se è difficile calcolare i logaritmi discreti per $p \equiv 3 \pmod 4$, allora è altrettanto difficile calcolare i logaritmi discreti modulo 4.

7.3 Invio di bit

Alice afferma di avere un metodo per predire i risultati delle partite di calcio e vuole venderlo a Bob. Bob le chiede di dimostrare che il suo metodo funziona predicendo i risultati delle partite che saranno giocate quella settimana. "Non ci penso proprio", risponde Alice. "Così tu puoi fare le tue scommesse e non pagarmi. Se vuoi che ti dimostri che il mio sistema funziona, che ne dici se ti mostro la mie predizioni per le partite della settimana scorsa?" Chiaramente qui c'è un problema. Ora vedremo come risolverlo.
Ecco il problema. Alice deve inviare a Bob un bit b (uguale a 0 o a 1) in modo che siano rispettate le seguenti restrizioni.

1. Bob non può determinare il valore del bit senza l'aiuto di Alice.
2. Alice non può cambiare il bit una volta che lo ha inviato.

Alice può mettere il bit in una scatola, chiudere la scatola usando il proprio lucchetto e inviarla a Bob. Quando Bob vuole il valore del bit, Alice rimuove il proprio lucchetto e Bob apre la scatola. Vogliamo implementare questa procedura matematicamente in modo che Alice e Bob non debbano essere nella medesima stanza quando il bit è rivelato.
Ecco una soluzione. Alice e Bob si mettono d'accordo su un primo grande $p \equiv 3 \pmod 4$ e una radice primitiva α. Alice sceglie a caso un numero $x < p-1$ il cui secondo bit x_1 è b e invia a Bob $\beta \equiv \alpha^x \pmod p$. Si assume che Bob non possa calcolare logaritmi discreti per p. Come osservato nell'ultimo paragrafo, questo significa che non può calcolare logaritmi discreti modulo 4. In particolare, non può determinare il valore di $b = x_1$. Quando Bob vuole conoscere il valore di b, Alice gli invia l'intero valore di x, in modo che, cercando x modulo 4, Bob possa trovare b. Alice non può inviare un valore di x differente da quello che ha già usato, perché Bob controlla se $\beta \equiv \alpha^x \pmod p$ e questa equazione ha un'unica soluzione $x < p-1$.
Tornando al calcio, per ogni partita Alice manda $b = 1$ se predice che la squadra vincerà, $b = 0$ se predice che perderà. Dopo che la partita è stata giocata, Alice rivela il bit a Bob, che può vedere se la sua predizione era corretta. In questo modo, Bob non può trarre profitto dalle informazioni che riceve prima della partita e Alice non può cambiare la propria predizione una volta che la partita è stata giocata.
L'invio dei bit può anche essere effettuato con molte altre funzioni unidirezionali. Per esempio, Alice puó prendere a caso una stringa da 100 bit, seguita dal bit b, seguito da un'altra stringa di 100 bit. Applica la funzione unidirezionale a questa stringa e invia il risultato a Bob. Dopo la partita, manda la stringa di 201 bit a Bob, che applica la funzione unidirezionale e fa un confronto con ciò che Alice aveva inviato originariamente.

7.4 Scambio della chiave di Diffie-Hellman

Un problema importante in crittografia è quello di scegliere le chiavi da usare in protocolli crittografici come DES o AES, specialmente quando le due parti sono ampiamente separate. I metodi a chiave pubblica come RSA forniscono una soluzione.

In questo paragrafo, verrà descritto un metodo differente, dovuto a Diffie e Hellman, la cui sicurezza è strettamente legata alla difficoltà di calcolare il logaritmo discreto: il protocollo di scambio della chiave di Diffie-Hellman (*Diffie-Hellman key exchange*). Tutti gli schemi di distribuzione della chiave presentano problemi di implementazione di vario genere. Alcuni di essi sono discussi nel Capitolo 10. Qui ci limiteremo a presentare l'algoritmo fondamentale di Diffie-Hellman.
Ecco come Alice e Bob stabiliscono una chiave privata K. Ogni loro comunicazione nel seguente algoritmo avviene attraverso un canale pubblico.

1. Alice, o Bob, sceglie un numero primo p, grande e sicuro[1], e una radice primitiva $\alpha \pmod p$. p e α possono essere resi pubblici.
2. Alice sceglie a caso un x segreto con $1 \leq x \leq p-2$ e Bob seleziona a caso un y segreto con $1 \leq y \leq p-2$.
3. Alice manda $\alpha^x \pmod p$ a Bob e Bob manda $\alpha^y \pmod p$ ad Alice.
4. Usando i messaggi che hanno ricevuto, possono entrambi calcolare la chiave K per la sessione. Alice calcola K come $K \equiv (\alpha^y)^x \pmod p$ e Bob calcola K come $K \equiv (\alpha^x)^y \pmod p$.

- Non c'è ragione che Alice e Bob abbiano bisogno di usare ~~tutte~~ le K come chiavi per le loro comunicazioni. Ora che hanno lo stesso numero K, possono usare una qualche procedura concordata preventivamente per derivare la chiave. Per esempio, possono usare i 56 bit centrali di K per ottenere una chiave DES.
- Se Eva ascolta tutte le comunicazioni tra Alice e Bob, allora conosce α^x e α^y. Se è in grado di calcolare logaritmi discreti, allora può trovare il logaritmo discreto di α^x e ottenere x. Quindi eleva α^y alla potenza x e ottiene $\alpha^{xy} \equiv K$. Una volta che Eva ha K, può usare la stessa procedura di Alice e Bob per estrarre una chiave di comunicazione. Quindi, se Eva può calcolare logaritmi discreti, può violare il sistema. Tuttavia, Eva non ha bisogno necessariamente di calcolare x o y per trovare K. Ciò che deve fare è risolvere il seguente problema.

> **Problema computazionale di Diffie-Hellman.** Sia p un primo e sia α una radice primitiva modulo p. Dati $\alpha^x \pmod p$ e $\alpha^y \pmod p$, trovare $\alpha^{xy} \pmod p$.

Non è noto se questo problema sia più facile del calcolo del logaritmo discreto. Il ragionamento precedente mostra che non è più difficile. Un problema correlato è il seguente.

> **Problema di decisione di Diffie-Hellman.** Sia p un primo e sia α una radice primitiva modulo p. Dati $\alpha^x \pmod p$, $\alpha^y \pmod p$ e $c \not\equiv 0 \pmod p$, decidere se $c \equiv \alpha^{xy} \pmod p$.

In altre parole, se Eva afferma di aver trovato un c con $c \equiv \alpha^{xy} \pmod p$ e cerca di vendere questa informazione, si può decidere se stia dicendo la verità? Naturalmente,

[1] Un numero primo p è sicuro per l'uso nell'algoritmo di Diffie-Hellman se è difficile calcolare il logaritmo discreto modulo p. (*N.d.Rev.*)

- se si può risolvere il problema computazionale di Diffie-Hellman, allora basta calcolare $\alpha^{xy} \pmod p$ e controllare se è uguale a c (e quindi si può ignorare l'offerta di Eva).
- Viceversa, un metodo che risolva il problema di decisione di Diffie-Hellman fornisce una soluzione del problema computazionale di Diffie-Hellman? Al momento la risposta non è nota. Un metodo ovvio è quello di scegliere molti valori di c e controllare ogni valore finché uno non uguagli $\alpha^{xy} \pmod p$. Ma questo metodo a forza bruta richiede un tempo pari almeno a quello richiesto dal calcolo dei logaritmi discreti mediante forza bruta, e quindi non ha alcun valore pratico. Ci sono analoghe situazioni che coinvolgono le curve ellittiche dove è nota una soluzione rapida per il problema di decisione di Diffie-Hellman, ma non è nota alcuna soluzione pratica per il problema computazionale di Diffie-Hellman (si veda [Washington]).

7.5 Crittosistema a chiave pubblica di ElGamal

Nel Capitolo 6, si è studiato un crittosistema a chiave pubblica la cui sicurezza si basa sulla difficoltà di fattorizzare un numero. È anche possibile progettare un sistema la cui sicurezza si basa sulla difficoltà di calcolare i logaritmi discreti. Questo è stato fatto da ElGamal nel 1985. Questo sistema non soddisfa esattamente la definizione di crittosistema a chiave pubblica data alla fine del Capitolo 6, poiché l'insieme dei possibili testi in chiaro (interi modulo p) non è uguale all'insieme dei possibili testi cifrati (coppie di interi (r,t) modulo p). Tuttavia, questo aspetto tecnico qui è irrilevante.
Alice vuole mandare un messaggio m a Bob. Bob sceglie un primo p grande e una radice primitiva α. Si assuma che m sia un intero tale che $0 \leq m < p$. Se m è grande, lo si spezzi in blocchi più piccoli. Anche Bob sceglie un intero a segreto e calcola $\beta \equiv \alpha^a \pmod p$. L'informazione (p, α, β) è resa pubblica ed è la chiave pubblica di Bob. Alice procede nel modo seguente.

1. Scarica (p, α, β).
2. Sceglie a caso un intero k segreto e calcola $r \equiv \alpha^k \pmod p$.
3. Calcola $t \equiv \beta^k m \pmod p$.
4. Manda a Bob la coppia (r,t).

Bob decifra calcolando

$$tr^{-a} \equiv m \pmod p.$$

Questo funziona perché

$$tr^{-a} \equiv \beta^k m(\alpha^k)^{-a} \equiv (\alpha^a)^k m\alpha^{-ak} \equiv m \pmod p.$$

Se Eva determina a, allora può anche decifrare mediante la stessa procedura che usa Bob. Quindi, è importante per Bob tenere a segreto. I numeri α e β sono pubblici e $\beta \equiv \alpha^a \pmod p$. La difficoltà di calcolare i logaritmi discreti è ciò che rende a sicura. Poiché k è un intero casuale, β^k sarà un intero non nullo modulo p casuale. Quindi $t \equiv \beta^k m \pmod p$ è m moltiplicato per un intero casuale e t è casuale mod p (a meno che $m = 0$, ma ciò dovrebbe essere evitato, naturalmente). Pertanto t non fornisce a

Eva alcuna informazione su m. La conoscenza di r non sembra dare a Eva sufficienti informazioni aggiuntive.
L'intero k è difficile da determinare a partire da r, poiché questo è ancora un problema di logaritmo discreto. Tuttavia, se Eva trova k, allora può calcolare $t\beta^{-k}$, che è m.
È importante che per ogni messaggio si usi un numero casuale k differente. Si supponga che Alice codifichi due messaggi m_1 e m_2 per Bob usando lo stesso valore di k per entrambi i messaggi. Allora r sarà lo stesso per entrambi i messaggi e i testi cifrati saranno (r, t_1) e (r, t_2). Se Eva trova il testo in chiaro m_1, può anche determinare m_2. Infatti Eva conosce t_1 e t_2. Pertanto, essendo

$$t_1/m_1 \equiv \beta^k \equiv t_2/m_2 \pmod{p},$$

può calcolare $m_2 \equiv t_2 m_1/t_1 \pmod{p}$.
Nel Capitolo 16, si incontrerà un metodo analogo a quello di ElGamal basato sulle curve ellittiche.

7.5.1 Sicurezza dei testi cifrati di ElGamal

Si supponga che Eva affermi di aver ottenuto il testo in chiaro m corrispondente a un testo cifrato c RSA. È facile verificare la sua affermazione: basta calcolare $m^e \pmod{n}$ e vedere se è uguale a c. Ora si supponga che invece Eva affermi di possedere il messaggio m corrispondente a una cifratura (r,t) di ElGamal. Si può verificare la sua affermazione? Questa verifica è difficile quanto il problema di decisione di Diffie-Hellman del Paragrafo 7.4. Per questo aspetto, l'algoritmo di ElGamal è pertanto molto differente dall'algoritmo RSA (naturalmente, se si aggiunge qualche elemento di casualità a un testo in chiaro RSA, per esempio attraverso OAEP, allora anche la cifratura RSA ha una proprietà simile).

Proposizione. *Una macchina che risolve i problemi di decisione di Diffie-Hellman modulo p può essere usata per decidere la validità di testi in chiaro di ElGamal modulo p e una macchina che decide la validità di testi in chiaro di ElGamal modulo p può essere usata per risolve i problemi di decisione Diffie-Hellman modulo p.*

Dimostrazione. Si supponga di avere una macchina M_1 che può decidere se una decifrazione di ElGamal è corretta. In altre parole, quando sono dati in ingresso p, α, β, (r,t), m, la macchina fornisce in uscita "sì" se m è la decifrazione di (r,t), "no" in caso contrario. Usiamo questa macchina per risolvere il problema di decisione di Diffie-Hellman. Si supponga di avere α^x e α^y e di dover decidere se $c \equiv \alpha^{xy}$. Siano $\beta = \alpha^x$ e $r = \alpha^y \pmod{p}$. Inoltre siano $t = c$ e $m = 1$. Si immetta tutto in M_1. Si noti che in questa situazione, x è l'intero segreto a e α^y prende il posto di $r \equiv \alpha^k$. La decifrazione corretta di (r,t) è $tr^{-a} \equiv cr^{-x} \equiv c\alpha^{-xy}$. Pertanto M_1 produce in uscita "sì" esattamente quando $m = 1$ è uguale a $c\alpha^{-xy} \pmod{p}$, ossia quando $c \equiv \alpha^{xy} \pmod{p}$.
Viceversa, si supponga di avere una macchina M_2 che può risolvere il problema di decisione di Diffie-Hellman. Questo significa che, se si immettono p, α, α^x, α^y e c in M_2, allora M_2 fornisce in uscita "sì" se $c \equiv \alpha^{xy}$ e "no" in caso contrario. Sia m l'ipotetica decifrazione del testo cifrato (r,t) di ElGamal. Si immetta $\beta \equiv \alpha^a$ come α^x, cosicché $x = a$, e si immetta $r \equiv \alpha^k$ come α^y, cosicché $y = k$. Si immetta tm^{-1} (mod p) come c. Si noti che m è il testo in chiaro corretto per il testo cifrato (r,t) se e solo se $m \equiv tr^{-a} \equiv t\alpha^{-xy}$, cosa che accade se e solo se $tm^{-1} \equiv \alpha^{xy}$. Pertanto m è il testo in chiaro corretto se e solo se $c \equiv tm^{-1}$ è la soluzione del problema di Diffie-Hellman. Di conseguenza, con questi dati in entrata, M_2 fornisce in uscita "sì" esattamente quando m è il testo in chiaro corretto. □

Il ragionamento appena utilizzato può anche essere usato per mostrare che risolvere il problema computazionale di Diffie-Hellman è equivalente a violare il sistema di ElGamal.

Proposizione. *Una macchina che risolve i problemi computazionali di Diffie-Hellman modulo p può essere usata per decifrare i testi cifrati di ElGamal modulo p e una macchina che decifra i testi cifrati di ElGamal modulo p può essere usata per risolvere i problemi di Diffie-Hellman modulo p.*

Dimostrazione. Se si ha una macchina M_3 che può decifrare tutti i testi cifrati di ElGamal, allora si inserisce in entrata $\beta \equiv \alpha^x$ (così $a = x$) e $r \equiv \alpha^y$. Scelto un qualunque valore non nullo di t, si ha che M_3 fornisce in uscita $m \equiv tr^{-a} \equiv t\alpha^{-xy}$. Quindi tm^{-1} (mod p) è la soluzione α^{xy} del problema computazionale di Diffie-Hellman.
Viceversa, si supponga di avere una macchina M_4 che può risolvere i problemi computazionali di Diffie-Hellman. Se si ha un testo cifrato di ElGamal (r,t), allora si inseriscono in entrata $\alpha^x = \alpha^a \equiv \beta$ e $\alpha^y \equiv \alpha^k \equiv r$. Di conseguenza M_4 fornisce in uscita $\alpha^{xy} \equiv \alpha^{ak}$. Poiché $m \equiv tr^{-a} \equiv t\alpha^{-ak}$, si è ottenuto il testo in chiaro m. □

7.6 Esercizi

1. **(a)** Sia $p = 13$. Calcolare $L_2(3)$.
   (b) Mostrare che $L_2(11) = 7$.

2. **(a)** Calcolare $6^5 \pmod{11}$.
   (b) Sia $p = 11$. Allora 2 è una radice primitiva. Sia $2^x \equiv 6 \pmod{11}$. Senza trovare il valore di x, determinare se x è pari o dispari.

3. Si può mostrare che 5 è una radice primitiva per il primo 1223. Si vuole risolvere il problema di logaritmo discreto $5^x \equiv 3 \pmod{1223}$. Dato che $3^{611} \equiv 1 \pmod{1223}$, determinare se x è pari o dispari.

4. Sia $p = 19$. Allora 2 è una radice primitiva. Usare il metodo Pohlig-Hellman per calcolare $L_2(14)$.

5. **(a)** Sia α una radice primitiva modulo p. Mostrare che

$$L_\alpha(\beta_1\beta_2) \equiv L_\alpha(\beta_1) + L_\alpha(\beta_2) \pmod{p-1}.$$

(*Suggerimento:* usare la proposizione del Paragrafo 3.7.)

(b) Più in generale, sia α arbitrario. Mostrare che

$$L_\alpha(\beta_1\beta_2) \equiv L_\alpha(\beta_1) + L_\alpha(\beta_2) \pmod{\mathrm{ord}_p(\alpha)},$$

dove $\mathrm{ord}_p(\alpha)$ è definito nell'Esercizio 20 del Capitolo 3.

6. Sia $p = 101$. Allora 2 è una radice primitiva. Si può mostrare che $L_2(3) = 69$ e $L_2(5) = 24$.

(a) Usando il fatto che $24 = 2^3 \cdot 3$, valutare $L_2(24)$.

(b) Usando il fatto che $5^3 \equiv 24 \pmod{101}$, valutare $L_2(24)$.

7. Supponendo di sapere che

$$3^6 \equiv 44 \pmod{137} \quad \text{e} \quad 3^{10} \equiv 2 \pmod{137},$$

trovare un valore di x con $0 \le x \le 135$ tale che $3^x \equiv 11 \pmod{137}$.

8. **(a)** Sia p un primo casuale di 500 cifre. Alcune persone vogliono immagazzinare delle password, scritte come numeri. Se la password è x, allora viene immagazzinato in un file il numero $2^x \pmod p$. Quando si riceve y come password, il numero $2^y \pmod p$ viene confrontato con l'elemento corrispondente all'utente nel file. Se qualcuno ottiene l'accesso al file, perché troverà difficile dedurre le password?

(b) Se invece p è un primo di cinque cifre, perché il sistema della parte (a) non è sicuro?

9. Riconsiderare l'Esercizio 22 del Capitolo 3 dal punto di vista dell'algoritmo di Pohlig-Hellman. L'unico primo q è 2. Sia $k = x_0 + 2x_1 + \cdots + 2^{15}x_{15}$, con k come in quell'esercizio.

(a) Mostrare che l'algoritmo di Pohlig-Hellman dà

$$x_0 = x_1 = \cdots = x_{10} = 0 \quad \text{e} \quad 2 = \beta = \beta_1 = \cdots = \beta_{11}.$$

(b) Usare l'algoritmo di Pohlig-Hellman per calcolare k.

10. Nel protocollo di scambio della chiave di Diffie-Hellman, Alice e Bob scelgono una radice primitiva α per un primo p grande. Alice manda $x_1 \equiv \alpha^a \pmod p$ a Bob e Bob manda $x_2 \equiv \alpha^b \pmod p$ ad Alice. Si supponga che Eva riesca a farsi dare da Bob i valori di b e di x_2, ma non il valore di α. Se $\mathrm{MCD}(b, p-1) = 1$, si mostri come Eva può dedurre α dalla conoscenza di p, x_2 e b.

11. Nel crittosistema di ElGamal, Alice e Bob usano $p = 17$ e $\alpha = 3$. Bob sceglie il proprio segreto come $a = 6$, cosicchè $\beta = 15$. Alice manda il testo cifrato $(r,t) = (7,6)$. Determinare il testo in chiaro m.

12. Si consideri il seguente attacco *Baby Step, Giant Step* all'RSA, con modulo pubblico n. Eva conosce un testo in chiaro m e un testo cifrato c. Sceglie $N^2 \ge n$ e compila due liste. La prima lista è $c^j \pmod n$ per $0 \le j < N$. La seconda lista è $mc^{-Nk} \pmod n$ per $0 \le k < N$.

(a) Si supponga che $\mathrm{MCD}(c,n) = 1$. Perché esiste sempre una coincidenza tra le due liste e in che modo una coincidenza permette a Eva di trovare l'esponente di decifrazione d?

(b) Probabilmente la risposta data in (a) è in parte falsa. Ciò che si è in realtà trovato è un esponente d tale che $c^d \equiv m \pmod n$. Dare un esempio di una coppia testo in chiaro–testo cifrato dove il d che si trova non è l'esponente di decifrazione. (Tuttavia, normalmente d è molto vicino ad essere l'esponente di decifrazione corretto.)

(c) Perché questo non è un buon attacco all'RSA? (*Suggerimento:* quanto sono lunghe le liste in confronto al tempo necessario per fattorizzare n mediante una divisione per tentativi?)

7.7 Problemi al calcolatore

1. Sia $p = 53047$. Verificare che $L_3(8576) = 1234$.

2. Sia $p = 31$. Valutare $L_3(24)$.

3. Sia $p = 3989$. Allora 2 è una radice primitiva modulo p.

(a) Mostrare che $L_2(3925) = 2000$ e $L_2(1046) = 3000$.

(b) Calcolare $L_2(3925 \cdot 1046)$. (*Osservazione:* la risposta dovrebbe essere minore di 3988.)

4. Sia $p = 1201$. Usare l'algoritmo di Pohlig-Hellman per trovare $L_{11}(2)$.