Crittografia

Riccardo Zanotto

7 ottobre 2019

Indice

Capitolo 1

NT algs

1.1 Conti di base

Alcuni algoritmi standard e trick vari per velocizzare i conti.

1.1.1 Alg euclideo esteso

Dati $a, b \in A$ anello che possiede una divisione euclidea, posso trovare u, v tali che $ua + vb = \gcd(a, b)$.

Definisco $v_0 = \begin{pmatrix} a \\ 1 \\ 0 \end{pmatrix}, v_1 = \begin{pmatrix} b \\ 0 \\ 1 \end{pmatrix}$ e poi una successione per ricorrenza $v_{i+1} = v_{i-1} - q_i v_i$ dove $r_{i-1} = q_i r_i + r_{i+1}$ è la divisione con resto e r_i è la prima coordinata di v_i.

Detti s_i, t_i la seconda e la terza componente di v_i, valgono un po' di cose:

- $r_i = as_i + bt_i$
- Eventualmente $r_{k+1} = 0$ e allora $r_k = \gcd(a, b)$
- Ad ogni passo s_i, t_i sono "piccoli"

Costo computazionale: Supposto $a > b$, il numero di iterazioni è $O(\log a)$. Il costo totale dunque può essere stimato con $O(\log^3 a)$.

Osservazione. Questo vuol dire che trovare l'inverso modulo n costa $\log^3 n$

1.1.2 Fast mod exp

Vogliamo calcolare $b^n \pmod m$. Scriviamo n in base 2 e calcoliamo con quadrati ripetuti le potenze $b, b^2, b^4, b^8, \ldots$ (sempre riducendo modulo m), moltiplicando quando ci sono gli 1 in n.

Costo computazionale: $O(\log n \cdot \log^2 m)$. Stiamo facendo $\log n$ moltiplicazioni tra numeri grossi al più m^2.

1.1.3 Montgomery multiplication

Per calcolare $ab \pmod m$ si fa il conto negli interi e poi la divisione con resto per m, che è un po' lenta.

Scegliamo allora un $r > m$ del tipo $r = 10^k$ (o comunque per cui è facile ridurre un numero).

Precalcoliamo $rr' = 1 + mm'$ con $0 < r' < m$ e $0 < m' < r$.

Lemma 1.1.1. *Dato un $x < mr$ so calcolare $xr' \pmod m$ solo con divisioni per r che sono veloci.*

Sia $s = xm' \mod r$. Allora $sm \equiv xmm' \pmod{mr}$, e aggiungendo x si ha $x + sm \equiv x(1 + mm') \equiv xrr' \pmod{rm}$.
Perciò vale $z = \frac{x+sm}{r} \in \mathbb{Z}$ ed è proprio quello che cercavamo $z \equiv xr' \pmod m$.

A questo punto se devo fare tanti conti con $a_1, \dots, a_n$ modulo m, calcolo subito $w = r^2 \mod m$ (con la divisione solita); poi porto tutto in rappresentanti di Montgomery: $b_i \equiv a_i r \pmod m$ e questo lo faccio tramite $a_i r \equiv a_i w r' \pmod m$.

Il rappresentante di una somma è banalmente la somma; il rappresentante del prodotto è facile da calcolare: $xyr \equiv (xr)(yr)r' \pmod m$. Quindi se ho i rappresentanti di x, y grazie al lemma posso trovare in fretta il rappresentante di xy.

Finiti tutti i conti posso di nuovo ritrasformare il risultato nella sua forma standard.

1.1.4 Quadrati mod p

Simboli di Legendre/reciprocità Sono noti. Lo abbiamo fatto con il conto su $G = \sum_{i=0}^{p-1} \left(\frac{i}{p}\right) \xi^i$.

Vogliamo anche trovare le radici quadrate modulo p. Cioè risolvere $x^2 \equiv a \pmod p$ con a residuo quadratico.

Algoritmo di Cipolla: Prendiamo un n tale che $n^2 - a$ non sia un quadrato. Sia $w = \sqrt{n^2 - a}$ nel campo $\mathbb{F}_{p^2}$. Allora $z = (n + w)^{\frac{p+1}{2}} \in \mathbb{F}_p$ è una radice quadrata di a.

Complessità: $O(\log^3 p)$.

Algoritmo di Tonelli-Shanks: Prendiamo n un nonresiduo. Scriviamo $p - 1 = 2^\alpha \cdot s$; calcoliamo $b = n^s \mod p$ e $r = a^{\frac{s+1}{2}} \mod p$.
Osserviamo che b è una radice 2^α-esima primitiva dell'unità, poiché non è un quadrato.
Inoltre $(r^2 a^{-1})^{2^{\alpha-1}} \equiv (a^s)^{2^{\alpha-1}} \equiv a^{\frac{p-1}{2}} \equiv 1 \pmod p$, perciò $r^2 a^{-1} \equiv b^{-2j} \pmod p$ per qualche $j < 2^{\alpha-1}$, cioè $a \equiv (rb^j)^2 \pmod p$.
Vogliamo allora trovare j e lo facciamo per induzione, trovandone le cifre binarie $j = j_0 + 2j_1 + \cdots + j_{\alpha-2} 2^{\alpha-2}$, in modo che $(b^{j_0 + \cdots + j_{k-1} 2^{k-1}} r)^2 a^{-1}$ sia una radice $2^{\alpha-k-1}$-esima.
Vale $\left((b^{j_0 + \cdots + j_{k-1} 2^{k-1}} r)^2 a^{-1}\right)^{2^{\alpha-k-2}} \equiv b^{-j_k 2^{\alpha-1}} \equiv (-1)^{j_k} \pmod p$, dunque calcolando LHS trovo $j_k = 0, 1$ a seconda se la potenza fa $1, -1$.

Complessità: $O(\log^2 p(\log p + \alpha^2))$ se conosco già n.

1.1.5 Karatsuba

Finora abbiamo detto che per moltiplicare due interi x, y di n cifre servono n^2 operazioni.

Tuttavia si può fare meglio: fissati un qualche B, m possiamo scrivere $x = x_1B^m + x_0, y = y_1B^m + y_0$ e osservare che $xy = x_1y_1B^{2m} + B^m(x_0y_1 + x_1y_0) + x_0y_0$.

Per calcolare questo numero servono 4 moltiplicazioni (e degli shift in base B), ma è possibile farlo in 3 poiché $x_0y_1+x_1y_0 = (x_0+x_1)(y_0+y_1)-x_0y_0-x_1-y_1$.

Complessità: Alla fine arriviamo a $O(n^{\log_2 3})$, scegliendo $B = 2$ e per ricorsione $m = n/2$.

1.2 Test di primalità

PRIMES in P. Ma non l'abbiamo fatto :(

1.2.1 Pseudoprimi

Uno dei test più ovvi per verificare se n è primo e vedere se $b^{n-1} \equiv 1 \pmod{n}$; se la congruenza fallisce, n è composto, altrimenti diciamo che n è uno *pseudoprimo* rispetto a b.

Proposizione 1.2.1. *Se n è uno pseudoprimo rispetto ad almeno un b, allora è uno pseudoprimo per almeno metà dei $b \in (\mathbb{Z}/n\mathbb{Z})^*$.*

Definizione 1.2.2. Un intero n si dice di *Carmichael* se è uno pseudoprimo rispetto a ogni $b \in (\mathbb{Z}/n\mathbb{Z})^*$.

Chi sono i numeri di Carmichael?

Proposizione 1.2.3. *Sia n un intero dispari. Se n non è squarefree, non è di Carmichael.*
Se n è squarefree, n è di Carmichael se e solo se $p - 1 \mid n - 1 \; \forall p \mid n$

Ecco un altro tipo di pseudoprimi:

Definizione 1.2.4. Diciamo che n intero dispari è un *pseudoprimo di Eulero* rispetto a b se vale

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

Proposizione 1.2.5. *Ogni n è pseudoprimo di Eulero per al più metà dei b coprimi con n.*

Abbiamo allora l'algoritmo di **Solovay-Strassen**: scegliamo k interi $0 < b < n$ random. Se n non è uno pseudoprimo rispetto a un qualche b, allora n è composto. Se invece n è pseudoprimo per tutti, allora è primo con probabilità circa $1 - 2^{-k}$.

1.2.2 Miller-Rabin

Definizione 1.2.6. Sia n un intero dispari, e scriviamo $n - 1 = 2^s t$ con t dispari. Diciamo allora che n è un *pseudoprimo forte* rispetto a b se $b^t \equiv \pmod{n}$, oppure $b^{t2^r} \equiv -1 \pmod{n}$ per qualche $r < s$.

Proposizione 1.2.7. *Se n è primo, è uno pseudoprimo forte per tutti i b.*
Se n è composto, allora è pseudoprimo forte per al più $1/4$ dei possibili b.

Il test consiste dunque dei seguenti step:

1. Scrivo $n - 1 = 2^s t$.

2. Scelgo un b random; calcolo $a \equiv b^t \pmod{n}$. Se $a \equiv 1 \pmod{n}$, restituisco "forse primo".
3. Faccio $a \mapsto a^2 \pmod{n}$ finché non trovo un -1 restituendo "forse primo".
4. Se non ho trovato nessun -1 restituisco "composto".
5. Se ho ottenuto "forse primo" rifaccio dal punto 2.

A questo punto se ho eseguito il punto 2 almeno k volte e non ho mai ottenuto "composto", so che n è primo con probabilità cir a $1 - 4^{-k}$.

1.2.3 Lucas e Pocklington-Lehmer

Proposizione 1.2.8. *Fissato un n intero positivo. Supponiamo che esista un $1 < a < n$ tale che $a^{n-1} \equiv 1 \pmod{n}$ e inoltre $a^{\frac{n-1}{q}} \not\equiv 1 \pmod{n}$ per ogni $q \mid n-1$ fattore primo. Allora n è primo*

La seconda condizione infatti dice che $(\mathbb{Z}/n\mathbb{Z})^*$ ha ordine $n-1$.
Tuttavia occorre fattorizzare $n-1$, che può essere difficile a piacere.

Proposizione 1.2.9. *Sia n un intero, e supponiamo che esistano a e p primo tali che:*

- $a^{n-1} \equiv 1 \pmod{n}$
- $p \mid n-1$ *e* $p > \sqrt{n} - 1$
- $\gcd(a^{\frac{n-1}{p}} - 1, n) = 1$

Allora n è primo.

Anche queste condizioni sono difficili da soddisfare in realtà, perché potrebbe non esistere un p che soddisfa la seconda condizione.

Proposizione 1.2.10. *Sia n un intero, e scriviamo $n-1 = ab$ con $a > \sqrt{n}$ e di cui conosciamo la fattorizzazione. Supponiamo che per ogni $p \mid a$ primo esiste un m_p tale che $m_p^{n-1} \equiv 1 \pmod{n}$ e $\gcd(m_p^{\frac{n-1}{p}} - 1, n) = 1$. Allora n è primo.*

Dunque alla fine il test di Pocklington consiste nel trovare molti fattori piccoli di $n-1$ sperando di superare $\sqrt{n}$ (e questa è la parte molto difficile). A quel punto si cercano degli a_p che soddisfino le condizioni; spesso $a_p = 2$ basta già da solo.

1.3 Fattorizzazione di polinomi

Per i polinomi ci pensa Knuth.

1.3.1 Polinomi su $\mathbb{F}_q$

Algoritmo di Berlekamp

Abbiamo $f \in \mathbb{F}_q[x]$ di grado n; possiamo supporlo squarefree dividendo per $\gcd(f, f')$, ovvero $f = f_1 \cdots f_r$.

Consideriamo la mappa $\varphi : {}^{\mathbb{F}_q[x]}\!/_{f(x)} \to {}^{\mathbb{F}_q[x]}\!/_{f(x)}$ data dall'elevamento alla q. Per il teorema cinese, la mappa $\varphi - \mathrm{id}$ è un endomorfismo di ${}^{\mathbb{F}_q[x]}\!/_{f_1(x)}\times$

$\ldots \times \mathbb{F}_q[x]/f_r(x)$ che è prodotto di campi finiti di grado potenze di q; in particolare $\ker(\varphi - \mathrm{id}) = \mathbb{F}_q^r$.

D'altra parte osserviamo che $v \in \ker(\varphi - \mathrm{id})$ se e solo se $v(x)^q \equiv v(x) \pmod{f(x)}$; inoltre $v^q - v = \prod_{s\in\mathbb{F}_q}(v-s)$. Dato che $\deg v < \deg f$, otteniamo una fattorizzazione non banale $f(x) = \prod_{s\in\mathbb{F}_q} \gcd(f(x), v(x) - s)$.

Per trovare questi v basta calcolare il nucleo di una matrice, in particolare quella data dal cambio base $x^{iq} \equiv Q_{i,n-1}x^{n-1} + \cdots + Q_{i,0} \pmod{f(x)}$.

L'algoritmo è allora dato da

1. Divido f per $\gcd(f, f')$.
2. Creo la matrice Q di cambio base
3. Trovo una base del nucleo di $Q - I$ con operazioni elementari
4. Calcolo tutti i $\gcd(f, v - s)$ con $v \in \ker(\varphi - \mathrm{id})$ e $s \in \mathbb{F}_q$.

Per q piccolo, allora i tempi di esecuzione sono di

1. $O(n^2)$ tramite algoritmo euclideo.
2. $O(qn^2)$: calcolo per ricorrenza i coefficienti di x^k per $k = 1, \ldots, qn$.
3. $O(n^3)$ con triangolazione gaussiana.
4. $O(qrn^2)$: provo tutti i gcd con tutti gli $s \in \mathbb{F}_q$.

Per q grosso, le moltiplicazioni costano $\log^2 q$, e lo step 4 chiede di provare troppi valori di s. Inoltre facciamo lo step 2 con la fast-exp (per fare il quadrato ci basta tenere i coefficienti fino a x^{2n}).

Usiamo poi il seguente step 4':

Dato $v \in \ker(\varphi - \mathrm{id})$ sappiamo $f \mid v^p - v = v(v^{\frac{p-1}{2}} - 1)(v^{\frac{p-1}{2}} + 1)$; abbastanza spesso calcolando $\gcd(f, v^{\frac{p-1}{2}} - 1)$ otteniamo un fattore non banale.
In particolare se $v(x) \equiv s_j \pmod{f_j(x)}$, vediamo che $f_j \mid v^{\frac{p-1}{2}} - 1$ se e solo se s_j è un quadrato, e questo accade circa $q/2$ volte. La probabilità che scelto un v a caso, il gcd scritto sopra ci dia informazioni non banali è esattamente $1 - \left(\frac{q-1}{2q}\right)^r - \left(\frac{q+1}{2q}\right)^r \geq \frac{4}{9}$.

Perciò dopo $O(\log r)$ pesche casuali di v abbiamo trovato tutti gli r fattori di f; il tempo totale di questo step 4' è di $O(n^2 \log^3 q \log r)$.

Cantor-Zassenhaus

Osserviamo che possiamo calcolare facilmente una fattorizzazione di $f = F_1 \cdots F_s$ con $F_i = \prod_{\deg j_f = i} f_j$, cioè una fattorizzazione di f in parti con fattori dello stesso grado. Questo perché vale $x^{q^d} - x = \prod_{\substack{\deg g \mid d \\ g \text{ irr}}} g(x)$, e allora ottengo $F_1 = \gcd(f, x^q - x)$, e poi $F_{i+1} = \gcd(\frac{f}{F_1 \cdots F_i}, x^{q^{i+1}} - x)$.

L'algoritmo di Cantor-Zassenhaus fattorizza poi ciascuno degli F_i, usando la formula

$$F_i(x) = \gcd(F_i, t) \cdot \gcd(F_i, t^{\frac{q^d-1}{2}} - 1) \cdot \gcd(F_i, t^{\frac{q^d-1}{2}} + 1)$$

valida per ogni $t \in \mathbb{F}_q[x]$, poiché $t(\alpha)^{q^d} = t(\alpha)$ per ogni α di grado d su $\mathbb{F}_q$.

Scelto un $t(x)$ random di grado $\leq 2d-1$, la formula sopra dà un fattore non banale circa il 50% delle volte

1.3.2 Polinomi su $\mathbb{Q}$

(leggere sul Childs)

La strategia è fattorizzare $f \in \mathbb{Z}[x]$ modulo un M molto grosso, in particolare più del doppio dei possibili valori assoluti di coefficienti di fattori di f. A quel punto i fattori che abbiamo trovato o corrispondono a fattori veri in $\mathbb{Z}[x]$, oppure f è irriducibile.

Ci servono due cose: il bound e un modo per fattorizzare modulo M. Potremmo prendere M primo e usare Berlekamp, ma useremo un altro metodo.

Lemma 1.3.1 (sollevamento di Hensel)**.** *Sia $f \in \mathbb{Z}[x]$ monico tale che $f \equiv g_1 h_1 \pmod m$, con g_1, h_1 coprimi modulo m.*
Allora esistono polinomi monici $g_2, h_2 \in \mathbb{Z}[x]$ tali che $g_1 \equiv g_2 \pmod m$, $h_1 \equiv h_2 \pmod m$ e $f \equiv g_2 h_2 \pmod{m^2}$; inoltre g_2, h_2 sono coprimi modulo m^2 e sono unici modulo m^2.

Dimostrazione. Sappiamo che $f = g_1 h_1 + mk$ con $\deg k < \deg(g_1 h_1)$. Cerchiamo $g_2 = g_1 + mb, h_2 = h_1 + mc$.
Mi serve $k \equiv bh_1 + cg_1 \pmod m$; dato che h_1, g_1 sono coprimi, trovo b, c di grado basso.
Per vedere che sono coprimi solleviamo anche la relazione $r_1 g_1 + s_1 h_1 \equiv 1 \pmod m$. □

Quindi una volta fattorizzato f su $\mathbb{F}_p$ in pochi fattori irriducibili, posso sollevare la fattorizzazione ad ogni p^{2^ℓ}, finché non supero il bound sui coefficienti.

Proposizione 1.3.2 (Mignotte)**.** *Sia $f = \sum_{i=0}^n a_i x^i$ e $g = \sum_{i=0}^d b_i x^i$. Allora se $g \mid f$ vale $\sum_{i=0}^d |b_i| \leq \left|\frac{b_d}{a_n}\right| 2^d \sqrt{\sum_{j=0}^n a_j^2}$.*

(serve che la misura di Mahler di un polinomio è $\geq$ della sua norma L^2, vedi qua)

1.4 Fattorizzazione in $\mathbb{Z}$

Sia n il numero da fattorizzare

1.4.1 Pollard's ρ

Consideriamo una funzione pseudo-random $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, tipo $f(x) = x^2 + 1$.
Prendiamo allora la successione $a_{i+1} = f(a_i)$ con a_0 scelto da noi.
Se $p \mid n$ è un fattore, allora da un certo punto in poi gli a_i sono periodici modulo p, ovvero $a_i \equiv a_j \pmod p$ e in particolare $p \mid \gcd(a_i - a_j, n)$ e abbiamo trovato un fattore di n.

La cosa che si fa solitamente è calcolare la sequenza a due velocità diverse, e dunque fare $\gcd(x_{2i} - x_i, n)$ ad ogni step.

Si può verificare che se $m \mid n$, allora vale $x_{2i} \equiv x_i \pmod m$ per i circa dell'ordine di $O(\sqrt{m})$.

Dunque la complessità dell'algoritmo (assumendo f random) è $O(\sqrt[4]{n})$.

1.4.2 Pollard $p - 1$

Definizione 1.4.1. Dato un bound B si dice che m è B-liscio se le potenze dei primi che dividono n sono minori di B.

Se $p \mid n$ e $p-1$ fosse B-liscio, preso $Q = \text{lcm}(1, 2, \ldots, B)$ avremmo che $p-1 \mid Q$.

Ma allora $p \mid a^Q - 1$ per ogni a, e in particolare per trovare p si calcola $\gcd(a^Q - 1, n)$.

L'algoritmo fissa dunque B, a e calcola per potenze successive $a^Q \mod n$, calcolando infine il gcd.

Purtroppo questo algoritmo ha molti point of failures: la scelta di B influenza moltissimo, ma ha anche un grande peso computazionale.

1.4.3 Crivello quadratico

1.5 Logaritmo discreto

1.5.1 Baby step-giant step

1.5.2 Pollard's ρ

1.5.3 Index calculus

1.6 Curve ellittiche

Posso anche usarle per fattorizzare e test di primalità, oltre a farci le potenze.

Capitolo 2

Codici

Un codice C è semplicemente un insieme di parole formate da lettere di un certo alfabeto.

A noi interessaranno però solo codici con una certa struttura; in particolare i nostri codici saranno sempre sottoinsiemi di un qualche $\mathbb{F}_q^n$.

Quello che vogliamo fare è inviare messaggi con una certa ridondanza in modo che il ricevente possa accorgersi se sono stati effettuati errori di trasmissione ed eventualmente correggerli da sè (es: comunicare con le sonde in giro per lo spazio).

2.1 Distanze ed errori

Un errore è quando una lettera della parola ricevuta è diversa dalla corssipondente nella parola inviata.

L'assunzione è che gli errori abbiano probabilità < 0.5 e siano indipendenti: quando ci arriva una parola allora vogliamo correggerla con quella del codice con cui condivide più lettere (MLD).

Definizione 2.1.1. Date $v = (a_1, \dots, a_n)$ e $w = (b_1, \dots, b_n)$ due parole, definiamo la *distanza di Hamming* $d(v, w) = \#\{i \mid a_i \neq b_i\}$.

Data una parola v, sia il suo *peso* $\mathrm{wt}(v) = d(v, 0)$.

Dato un codice C definiamo infine la *distanza* del codice come

$$d_C = \min_{\substack{v,w \in C \\ v \neq w}} d(v, w)$$

Definizione 2.1.2. Sia v la parola inviata e w la parola ricevuta. L'*errore* è $e = w - v$.

Diciamo che il codice C *rileva* l'errore e se $v + e \notin C \ \forall v \in C$, ovvero se w non fa parte del codice.

Diciamo inoltre che il codice *corregge* e con v se vale $d(v', v+e) > d(v, v+e)$ per ogni $v \neq v' \in C$.

Vediamo ora che la distanza è una quantità fondamentale di un codice:

Proposizione 2.1.3. *Sia C un codice di distanza d. Allora*

- *il codice rileva tutti gli errori e con* $\mathrm{wt}(e) \leq d - 1$
- *il codice corregge tutti gli errori con* $\mathrm{wt}(e) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$

Abbiamo dunque capito che sono molto importanti le palle centrate in parole del codice.

Osservazione. La palla $B(v,t) = \{w \in \mathbb{F}_q^n \mid d(w,v) \leq t\}$ ha cardinalità

$$B_t = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i$$

Proposizione 2.1.4 (Hamming bound)**.** *Sia C un codice di distanza d, e $t = \left\lfloor \frac{d-1}{2} \right\rfloor$. Allora vale*

$$B_t \cdot \#C \leq q^n$$

Definizione 2.1.5. Se vale l'uguaglianza, diciamo che C è un codice *perfetto*, ovvero $\mathbb{F}_q^n$ viene partizionato completamente dalle palle di centro $v \in C$ e raggio t (che è quello di cui sappiamo correggere).

2.2 Codici lineari

Dico che un codice C è *lineare* di *dimensione* m se è un sottospazio vettoriale m-dimensionale di $\mathbb{F}_q^n$.

Osservazione. Se C è lineare, allora vale $d = \min_{v \neq 0} \mathrm{wt}(v)$.

Posso considerare allora una base $b_1, \ldots, b_m$ di C, e la matrice G che ha per righe i b_i. Ogni parola $v \in C$ è perciò della forma uG con $u \in \mathbb{F}_q^m$.

Osservo poi che C è generato da G, ma anche da ogni matrice equivalente a G con operazioni elementari di riga. In particolare posso prendere $G' =$, in modo che $uG' = (u \mid uX)$. Questa scelta di G si dice *codifica sistematica*, perché permette la decodifica immediata.

Possiamo inoltre considerare $C^\perp$ lo spazio ortogonale a C, che avrà una base $w_1, \ldots, w_{n-m}$, con matrice H detta *matrice di parità*.

Vale infatti $GH^t = 0$, ovvero $x \in C$ se e solo se $Hx = 0$. Data una parola ricevuta w, chiamiamo *sindrome* la quantità Hw, che ci dovrebbe dire dove e quali sono gli errori.

Notiamo inoltre che se $G = \left(\begin{array}{c|c} I_m & X \end{array} \right)$, allora la matrice di parità corrispondente è $H = \left(\begin{array}{c|c} -X^t & I_{n-m} \end{array} \right)$.

Proposizione 2.2.1. *Un codice lineare ha distanza d se e solo se* $\mathrm{rk}\, H = d-1$*, ovvero ogni $d-1$ righe sono indipendenti, ma esistono d righe dipendenti.*

Come correggo gli errori?

Considero $\mathbb{F}_q^n / C$ le classi laterali di C; osservo che le sindromi sono in corrispondenza biunivoca con le classi laterali.

Se mi arriva una parola w, calcolo la sindrome Hw; tra tutte le parole in $w + C$ trovo quella con peso minore v' (il cosiddetto *coset leader*), e allora traduco w con $w - v'$ che è la parola del codice più vicina a w.

MA: ci possono essere più di un coset leader...

Proposizione 2.2.2 (Singleton bound)**.** *Se C è un codice lineare di tipo (n, m, d) allora vale $d \leq n - m + 1$.*

Definizione 2.2.3. Se un codice C soddisfa $d = n - m + 1$, viene detto MDS (maximum distance separable).

Teorema 2.2.4 (Gilbert-Varshamov)**.** *Siano n, m, d fissati. Esiste un codice lineare di lunghezza n, dimensione m e distanza d su $\mathbb{F}_q$ se vale*

$$\sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i < q^{n-m}$$

2.2.1 Codice di Hamming binario

Fissato r, sia H la matrice con r righe e che ha per colonne le rappresentazioni binarie dei numeri $1, 2, \dots, 2^r - 1$.

Il codice di Hamming $\mathcal{H}_r$ è il codice binario di lunghezza $n = 2^r - 1$ che ha H per matrice di parità.

Osservazione. Se e_i è il vettore che ha zeri ovunque e un 1 in posizione i, allora He_i è la rappresentazione di i in base 2.

Proposizione 2.2.5. *Il codice $\mathcal{H}_r$ ha distanza $d = 3$ ed è perfetto.*

Dimostrazione. La distanza si vede dall'indipendenza delle righe.

Per verificare che è perfetto deve valere $b_1 \cdot \#C = 2^n$, ovvero $\left(\binom{n}{0} + \binom{n}{1}\right) 2^{n-r} = 2^n$. Ma i binomiali valgono $1 + n = 2^r$, quindi l'uguaglianza è verificata. □

2.2.2 Codice esteso

Se abbiamo un codice lineare con matrici G, H, possiamo considerare il codice C' ottenuto aggiungendo un'ultima riga di check alla matrice H, ovvero $H' = \begin{pmatrix} H & 0 \\ j & -1 \end{pmatrix}$ dove j è il vettore di tutti 1 (stiamo quindi aggiungendo il check dello XOR). Osserviamo che $\operatorname{rk} H' = \operatorname{rk} H + 1$.

La matrice G' corrispondente sarà della forma $G = (G|b)$; imponendo $G'(H')^t = 0$ ricaviamo $b = Gj$, ovvero l'ultimo valore che aggiungiamo all'encoding è la somma di tutti i precedenti.

L'utilità dei codici estesi è soprattutto su $\mathbb{F}_2$, poiché $\operatorname{wt}(v') = \operatorname{wt}(v) + 1$ se $\operatorname{wt}(v)$ era dispari: allora possiamo trasformare un codice di distanza $d = 2m - 1$ in un codice di distanza $d' = 2m$ al prezzo di un solo bit di lunghezza in più.

2.3 Codici ciclici

Definizione 2.3.1. Un codice C di lunghezza n è ciclico se $(a_1, \dots, a_n) \in C$ implica $(a_n, a_1, \dots, a_{n-1}) \in C$.

Definiamo poi la mappa $\varphi : C \to {}^{\mathbb{F}_q[x]}\!/_{x^n - 1}$ data da $\varphi((c_0, \dots, c_{n-1})) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$.

L'operazione di shift corrisponde alla moltiplicazione per x, quindi $\varphi(C)$ è un ideale dell'anello quoziente.

Esiste quindi un polinomio $g(x) = g_0 + g_1 x + \cdots + g_m x^m$ che genera l'ideale C, il che si traduce in:

- una parola $u(x)$ appartiene al codice se e solo se $g \mid u$ nel quoziente.
- il codice ha dimensione $k = n - m$, e una base è data da $g, xg, \dots, x^{k-1} g$.

Vale infine il teorema di caratterizzazione:

Teorema 2.3.2. *Un polinomio g è il generatore di un codice ciclico di lunghezza n se e solo se $g(x) \mid x^n - 1$.*

Un modo di codificare è prendendo un polinomio $u(x)$ di grado $< k$, e codificandolo con $u(x)g(x)$.

Questo corrisponde a prendere la matrice G della base $g, xg, \ldots, x^{k-1}g$, che si scrive tipo

$$\begin{pmatrix} g_0 & \cdots & g_m & & \\ & g_0 & \cdots & g_m & \\ & & \ddots & \ddots & \ddots \\ & & & g_0 & \cdots & g_m \end{pmatrix}$$

Sia poi $g(x)h(x) = x^n - 1$; questo h dà esattamente la matrice di parità, poiché $f \in C$ se e solo se $h(x)f(x) \equiv 0 \pmod{x^n - 1}$.

2.3.1 Codifica sistematica

Usiamo invece un altro modo di codificare: dato un messaggio u di grado $< k$, considero la divisione con resto $x^{n-k}u(x) = a(x)g(x) + r(x)$. Invio allora $x^{n-k}u(x) - r(x)$, che è un elemento del codice. La decodifica avverrebbe dunque guardando le ultime k cifre, mentre le prime m sono di parità.

Per scrivere esplicitamente le matrici devo considerare la base formata dagli $u = x^i$ per $i = 0, \ldots, k-1$, ovvero scrivere $x^{n-k+i} = a_i(x)g(x) + b_i(x)$, con $b_i(x) = \sum_{j=0}^{n-k-1} b_{i,j}x^j$; le matrici sono dunque

$$G = \begin{pmatrix} -b_{0,0} & \cdots & -b_{0,n-k-1} & 1 & 0 & \cdots & 0 \\ -b_{1,0} & \cdots & -b_{1,n-k-1} & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & & & \ddots & \\ -b_{k-1,0} & \cdots & -b_{k-1,n-k-1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 0 & \cdots & 0 & b_{0,0} & \cdots & b_{k-1,0} \\ 0 & 1 & \cdots & 0 & b_{0,1} & \cdots & b_{k-1,1} \\ & & \ddots & & \vdots & \ddots & \\ 0 & 0 & \cdots & 1 & b_{0,n-k-1} & \cdots & b_{k-1,n-k-1} \end{pmatrix}$$

Si può usare per l'encoding anche il polinomio h: sapendo che $hv = 0$, dove v è l'encoding di u, ci sono delle equazioni che permettono di ricavare $v_0, \ldots, v_{n-k-1}$ in funzione di $v_{n-k}, \ldots, v_{n-1} = u_0, \ldots, u_{k-1}$.

Infine per la decodifica la sindrome di w è semplicemente $w \mod g$, e si procede poi a cercarne il coset leader.

2.3.2 Zeri di polinomi

Se $(n, q) = 1$ allora il polinomio $x^n - 1$ si spezza completamente su un $\mathbb{F}_{q^m}$, ovvero $x^n - 1 = \prod(x - \alpha^i)$ per qualche α radice primitiva dell'unità.

In particolare si avrà $g(x) = (x - \alpha^{i_1}) \cdots (x - \alpha^{i_{n-k}})$ per certi indici, quindi il codice ciclico generato da g può essere visto come i polinomi $c \in \mathbb{F}_q[x]/x^n - 1$ tali che $c(\alpha^{i_j}) = 0$ per ogni j, ovvero la matrice di parità è data semplicemente dalla valutazione negli α^{i_j}.

2.4 Codici BCH

Lemma 2.4.1 (BCH bound)**.** *Sia C un codice ciclico su $\mathbb{F}_q$ di lunghezza n con generatore $g(x)$ tale che esistono $b \geq 0, \delta \geq 1$ per cui $g(\alpha^b) = \cdots = g(\alpha^{b+\delta-2}) = 0$, dove α è un generatore di $\mathbb{F}_{q^r}^*$.*
Allora C ha distanza almeno δ.

Definizione 2.4.2. Si dice codice BCH binario di lunghezza n e sistanza designata δ il codice ciclico con generatore $g = \operatorname{lcm}(m_b, \ldots, m_{b+\delta-2})$ dove $m_b(x)$ è il polinomio minimo di α^i (α è un generatore di $\mathbb{F}_{2^r}$ con $n \mid 2^r - 1$)

Solitamente si usa $b = 1, n = q^r - 1$.

2.4.1 2-error correcting

Consideriamo $\mathbb{F}_{2^r}$ e un suo generatore β; sia $n = 2^r - 1$ e prendiamo $g(x) = \mu_\beta(x)\mu_{\beta^3}(x) \mid x^n - 1$.

Allora g genera un codice ciclico di lunghezza n e dimensione $n - 2r$, la cui matrice di parità è data da

$$\begin{pmatrix} 1 & \beta & \beta^2 & \ldots & \beta^i & \ldots & \beta^{2^r-2} \\ 1 & \beta^3 & \beta^6 & \ldots & \beta^{3i} & \ldots & \beta^{3(2^r-2)} \end{pmatrix}$$

La sindrome di una parola w è esattamente $\begin{pmatrix} w(\beta) \\ w(\beta^3) \end{pmatrix}$, e da questa possiamo correggere fino a due errori, cioè fino a $e(x) = x^i + x^j$ (è un sistema di 2 equazioni in 2 incognite).

2.4.2 Decodifica

Se riceviamo una parola w, l'errore è $e = w - v$ e la sindrome è data da $s_i = w(\alpha^i) = e(\alpha^i)$. Assumiamo inoltre che $\deg e \leq t$ dove t è la capacità di correzione del codice.

Definizione 2.4.3. Sia $L = \{i \mid e_i \neq 0\}$ l'insieme delle posizioni di errore.
Il polinomio *locatore d'errore* è $\sigma(x) = \prod_{\ell \in L}(1 - x\alpha^\ell)$
Il polinomio *valutatore d'errore* è $\omega(x) = \sum_{\ell \in L} e_\ell \alpha^\ell \prod_{i \in L \setminus \{\ell\}}(1 - x\alpha^i)$
Il polinomio *di sindrome* è $S(x) = e(\alpha) + e(\alpha^2)x + \cdots + e(\alpha^{2t})x^{2t-1}$

Osservazione. I polinomi ω e σ sono coprimi.

Notiamo che c'è un errore in posizione ℓ se e solo se $\sigma(\alpha^{-\ell}) = 0$, nel qual caso troviamo l'errore tramite $e_l = -\frac{\omega(\alpha^\ell)}{\sigma'(\alpha^{-\ell})}$

La decodifica si basa sul seguente risultato, detto *equazione chiave*

Teorema 2.4.4. *I polinomi σ, ω soddisfano*

$$\sigma(x)S(x) \equiv \omega(x) \pmod{x^{2t}}$$

Inoltre sono gli unici (a meno di multipli) con $\deg \omega < \deg \sigma \leq t$

Una volta calcolato il polinomio sindrome $S(x)$ bisogna dunque risolvere l'equazione chiave. Lo facciamo con l'algoritmo euclideo.

2.4.3 Codici Reed-Solomon

È un codice BCH con $n = q - 1$; in particolare $g(x) = (x - \alpha^b) \cdots (x - \alpha^{b+\delta-2})$ con α generatore di $\mathbb{F}_q$.

Osservazione. La distanza di un RS è esattamente $d = \delta$, quindi è MDS.

Possiamo poi ricavare dal RS di distanza d su $\mathbb{F}_{p^l}$ un codice su $\mathbb{F}_p$ vedendo ogni elemento di $\mathbb{F}_{p^l}$ come una parola lunga l di $\mathbb{F}_p$: il nuovo codice ha dunque lunghezza $n' = ln = l(p^l - 1)$.

2.5 Codici di Goppa

Definizione 2.5.1. Prendiamo un insieme $L = \{\gamma_0, \ldots, \gamma_{n-1}\} \subset \mathbb{F}_{q^r}$ e un polinomio $g(x) \in \mathbb{F}_{q^r}[x]$.

$$\Gamma(L, g) = \left\{ (c_0, \ldots, c_{n-1}) \in \mathbb{F}_q^n \mid \sum \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{g(x)} \right\}$$

Osservazione. Con $g(x) = x^{\delta-1}$ e $L = \{\alpha^{-i}\}$ otteniamo un codice BCH di distanza designata δ.

Capitolo 3

Crittografia