UNIVERSITÀ DEGLI STUDI DI PISA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CORSO DI LAUREA IN MATEMATICA

TESI DI LAUREA

CALCOLO DELLA DISTRIBUZIONE DEI PESI NEI CODICI CICLICI ACCORCIATI

27 OTTOBRE 2005

CANDIDATO
FELICE MANGANIELLO

RELATORE
PROF.SSA PATRIZIA GIANNI

CONTRORELATORE
PROF. CARLO TRAVERSO

ANNO ACCADEMICO 2004-2005

Indice

Introduzione

Il problema del trasferimento di dati e informazioni è un problema sul quale ormai si basa molta della organizzazione della vita odierna. Uno degli scopi della teoria algebrica dei codici è quello di cercare metodi per la costruzione di codici che possano trasmettere informazioni in modo sempre più efficiente ed affidabile, riducendo la probabilità che si commettano errori di decodifica. Per questo motivo sono stati creati codici in grado di rilevare gli errori di trasmissione (error detecting codes) e codici capaci anche di correggere gli errori di trasmissione rilevati (error correcting codes). La scelta di un tipo di codice o dell'altro dipende dalle necessità dell'applicazione.

Per i codici il cui scopo è solo quello di rilevare errori di trasmissione, una classe particolarmente usata è quella dei codici a ridondanza ciclica,(CRC codes). Questi codici sono una generalizzazione dei codici a controllo di parità, (parity-check codes). In questi codici si aggiungono ad ogni parola da trasmettere una o più informazioni che permettono di verificare la correttezza del messaggio ricevuto. Nei codici CRC, gli elementi di controllo aggiunti sono più di uno ed essi sono scelti in modo che le parole del codice siano rappresentate come polinomi multipli del polinomio che caratterizza il codice, chiamato polinomio generatore del codice.

Una volta decisa la classe e il tipo di codice da usare, al fine di effettuare una scelta ottimale del codice, vengono analizzati due parametri: la distanza minima, che, al variare delle parole del codice, misura il minimo numero di posizioni in cui una parola differisce da tutte le altre, e la probabilità che un errore di trasmissione non venga rilevato, indicata con P_{ue}, (undetected error probability). Infatti le prestazioni di un codice migliorano con l'aumentare della distanza minima e con il diminuire della P_{ue}.

Si possono calcolare sia la distanza minima che la P_{ue} del codice considerato,

se si conosce il peso (ossia la distanza dalla parola nulla) di ogni parola del codice, e valutando la distribuzione dei pesi, al variare di tutti i pesi possibili.

Per calcolare la distribuzione dei pesi è necessario conoscere il peso di ogni parola e questo, spesso, rende il confronto di due codici, usando la distanza minima e la P_{ue}, un'operazione molto complessa.

Nel caso speciale dei codici CRC, le proprietà algebriche che vengono usate per definire la ridondanza possono essere sfruttate per diminuire notevolmente il costo computazionale della distribuzione dei pesi. Infatti è possibile calcolare i pesi delle parole di un codice CRC, con polinomio generatore $g(x)$, studiando la struttura dell'anello $\mathcal{R}_g^q := {}^{\mathbb{F}_q[x]}/_{(g(x))}$, [CBH], [MS].

Più precisamente, il peso di una parola in $\mathcal{R}_g^q$ e quello della stessa moltiplicata per x possono differire solo per 1, -1, 0. A questo punto basta sfruttare la decomposizione di $\mathcal{R}_g^q$ come unione delle orbite determinate dall'azione della moltiplicazione per x.

In questo modo il problema del calcolo dei pesi delle parole del codice è ricondotto alla costruzione di un insieme finito di polinomi di $\mathcal{R}_g^q$ le cui orbite siano tutte disgiunte e forniscano una decomposizione di $\mathcal{R}_g^q$.

Per esplicitare tali polinomi si studia la struttura dell'anello $\mathcal{R}_g^q$. Grazie al Teorema Cinese del Resto, se il polinomio generatore del codice $g(x)$ si fattorizza come $g(x) = \prod_{i=1}^{m} {g_i}^{e_i}$, si ha:

$$\mathcal{R}_g^q \approx \prod_{i=1}^{m} \mathcal{R}_{g_i^{e_i}}^q,$$

così la ricerca dei generatori delle orbite può essere fatta su anelli del tipo $\mathcal{R}_{g_i^{e_i}}^q$ con $g_i(x)$ polinomio irriducibile. Per terminare la ricerca si studiano infine i gruppi moltiplicativi $(\mathcal{R}_{g_i^s}^q)^*$ con $0 \leq s \leq t$, che vengono decomposti a loro volta come prodotto di gruppi ciclici.

Questo tipo di approccio viene descritto più diffusamente dei tre capitoli centrali della tesi. Più precisamente, dopo un primo capitolo di richiami di algebra commutativa e teoria dei codici, vi è una seconda parte dedicata all'individuazione dei generatori dell'anello $\mathcal{R}_g^q$. Nel terzo capitolo vengono descritte le orbite che partiscono $\mathcal{R}_g^q$, ed infine nel quarto verrà mostrato l'algoritmo che permette di calcolare la distribuzione dei pesi di un codice.

La tesi trae spunto dall'articolo [CBH], dove l'argomento è trattato per codici binari, cioè codici con coefficienti in $\mathbb{F}_2$, e generalizza al caso di campi $\mathbb{F}_{p^n}$ i risultati citati nel lavoro originale.

In appendice viene poi presentata una implementazione in C++ delle procedure descritte, che utilizza la libreria NTL (Number Theory Library) per una efficiente implementazione delle operazioni sui campi finiti.

In questo modo si sono ottenuti risultati interessanti confrontando per vari codici sia la distribuzione dei pesi che la loro P_{ue}. Infine è stata anche realizzata una presentazione grafica dei dati calcolati, che permette una visualizzazione efficace dei risultati ottenuti.

Capitolo 1

Preliminari e Notazioni

Dedichiamo questo capitolo a fornire le nozioni di algebra che servono alla descrizione dell'algoritmo come scritto in [CBH].

Iniziamo il capitolo enunciando il Teorema di Caratterizzazione dei Gruppi Abeliani Finiti che servirà, nel secondo capitolo, per la descrizione di gruppi moltiplicativi associati a campi finiti. Nella sezione sui campi finiti ci soffermiamo sull'enunciato di teoremi che ne determinano le proprietà fondamentali, come ad esempio il Teorema dell'Elemento Primitivo. La parte più interessante di questo capitolo è quella in cui il lavoro si concentra sull'ordine di polinomi. Affrontiamo a tappe il problema della ricerca di tale ordine ottenendo infine un criterio per il calcolo di quest'ultimo per polinomi qualsiasi. L'utilizzo di questo oggetto matematico sarà rilevante nel terzo capitolo.

La terza sezione, la più importante del capitolo, è dedicata ad un ripasso di teoria dei codici. In particolare si concentra su una particolare classe di codici per il rilevamento di errori. La classe di codici trattati è quella dei codici ciclici accorciati definiti a partire dai codici ciclici.

Iniziamo col definire i codici ciclici, e diamo tutte le nozioni necessarie per affrontare lo studio dei codici ciclici accorciati. I codici ciclici accorciati vengono anche detti codici a ridondanza ciclica o "Cyclic Redundancy-Check Codes", e li indicheremo con CRC.

Nello studio delle proprietà di un codice sono molto importanti le definizioni di peso di una parola e di distanza tra due di esse. Queste sono applicazioni definite su tutto $\mathbb{F}_q^n$, ma che applicate ad un codice diventano di fondamentale

utilizzo in sede di rilevamento o di correzione di errori.

Introduciamo anche il codice duale di un codice lineare. Quello duale è un codice strettamente legato a quello originale. In certi casi, lo studio delle proprietà del duale, permette di trarre conclusioni sulle caratteristiche del codice originale.

In relazione a quanto appena scritto, dedichiamo una sottosezione al teorema di MacWilliams, come riportato in [MS]. Il Teorema di MacWilliams è molto importante in teoria dei codici perchè mette in relazione i *weight enumerator* di un codice con quello del suo duale. La relazione in esso contenuta permette, nel calcolo della distribuzione di pesi, il passaggio da un codice al suo duale . Il teorema è molto utilizzato nel momento in cui il numero di parole di uno dei due codici, originale o duale, risulti essere molto inferiore rispetto a quello dell'altro. Questo teorema quindi è importante perchè permette una scelta del codice da prendere in considerazione, che può rendere molto inferiore il costo computazionale di certe proprietà del codice.

Solo dopo questa parte affrontiamo da vicino i codici CRC. I CRC sono dei codici lineari derivanti direttamente dai codici ciclici. Essi sono codici particolarmente comodi perchè, nonostante non mantengano la struttura ciclica, ereditano varie proprietà dei codici ciclici da cui sono generati. Nel nostro caso sfrutteremo la caratterizzazione delle parole del codice duale, tramite il polinomio generatore, per il calcolo della distribuzione dei pesi.

Un'altra sottosezione affronta il calcolo della *undetected error probability*, P_{ue}, come troviamo in [CW]. Questa è una proprietà fondamentale dei codici per il rilevamento di errori. La sottosezione è dedicata alla definizione di questa proprietà e alla formulazione di tale probabilità rispetto alla distribuzione dei pesi del codice e a quella del suo duale.

1.1 Gruppi Abeliani Finiti

In questa sezione indicheremo con $\mathcal{G} = (\mathcal{G}, \cdot)$ un gruppo abeliano finito di ordine m. Se $p \in \mathbb{Z}$ un numero primo, diremo che $\mathcal{G}$ è un $p - gruppo$ se $m = p^k$ è una potenza di p. Inoltre se $\mathcal{H} \subset \mathcal{G}$ è un sottogruppo di $\mathcal{G}$, diremo che $\mathcal{H}$ è un $p - sottogruppo$ di $\mathcal{G}$ se $\mathcal{H}$ è un p-gruppo. Chiameremo $\mathcal{H}$ un *p-sottogruppo di Sylow* se l'ordine di $\mathcal{H} = p^s$ e se p^s è la più alta potenza di p che divide m. Rimandiamo per le dimostrazioni dei risultati richiamati a [La].

Teorema 1.1 (Classificazione dei Gruppi Abeliani Finiti). *Sia $\mathcal{G}$ un gruppo abeliano finito. Allora $\mathcal{G}$ è il prodotto diretto dei suoi p-sottogruppi $\mathcal{G}(p)$, al variare di p fra i numeri primi tali che $\mathcal{G}(p) \neq \{0\}$.*

Teorema 1.2. *Ogni p-gruppo abeliano finito $\mathcal{G}$ è isomorfo al prodotto di p-gruppi ciclici, ed è quindi della forma*

$$\mathcal{G} \cong \mathbb{Z}/(p^{r_1}\mathbb{Z}) \times \cdots \times \mathbb{Z}/(p^{r_s}\mathbb{Z})$$

con $r_1 \geq \cdots \geq r_s \geq 1$ univocamente determinati.

Corollario 1.3. *Ogni gruppo abeliano finito è prodotto di gruppi ciclici.*

1.2 Campi Finiti

In questa sezione richiamiamo alcune delle principali proprietà dei campi finiti che saranno usate nel seguito. Anche in questo caso per le dimostrazioni dei risultati presentati rimandiamo a [La].

Sia F un campo finito. F ha allora $q = p^\delta$ elementi, dove p è un numero primo, chiamato *caratteristica* di F. Inoltre se F^* è il gruppo moltiplicativo di F, allora F^* è un gruppo con con $p^\delta - 1$ elementi. Chiameremo un elemento $a \in F$ *primitivo* se ha ordine $p^\delta - 1$.

Teorema 1.4 (Elemento Primitivo). *Ogni campo finito F con p^δ elementi possiede un elemento primitivo, i.e. il gruppo F^* è ciclico.*

Osservazione. Se F ha p^δ elementi, allora ogni elemento di F^* soddisfa l'equazione $x^{p^\delta - 1} = 1$, quindi il polinomio $x^{p^\delta} - x$ si decompone in fattori lineari su F.

Definizione 1.5. *Sia $\bar{F}$ una chiusura algebrica di F, campo finito. Il* campo di spezzamento *di un polinomio $f(x) \in F[x]$ sul campo F è il minimo campo contenuto in $\bar{F}$ che contiene F e su cui $f(x)$ può essere rappresentato come prodotto di fattori lineari.*

Teorema 1.6. *Sia $\bar{\mathbb{F}}_p$ una chiusura algebrica di $\mathbb{F}_p$. Per ogni intero $\delta \geq 1$ esiste un campo finito con p^δ elementi, indicato con $\mathbb{F}_{p^\delta} \subset \bar{\mathbb{F}}_p$, che è il campo di*

spezzamento del polinomio

$$x^{p^\delta} - x.$$

Ogni campo finito con p^δ elementi è isomorfo ad un campo $\mathbb{F}_{p^\delta}$.

Vale anche:

Teorema 1.7. *Sia $f(x) \in \mathbb{F}_q[x]$ irriducibile di grado r. Allora le sue radici sono contenute in $\mathbb{F}_{q^r}$, inoltre, se $\alpha \in \mathbb{F}_{q^r}$ è una radice di f, allora lo sono anche tutti gli elementi $\alpha, \alpha^q, \ldots, \alpha^{q^{r-1}} \in \mathbb{F}_{q^r}$.*

Vogliamo ora introdurre la nozione di ordine un polinomio, per questo abbiamo bisogno del seguente:

Lemma 1.8. *Sia $g(x)$ un polinomio in $\mathbb{F}_q[x]$. Allora esiste $n \in \mathbb{N}$, tale che $g(x)$ divide $x^n - 1 \in \mathbb{F}_q[x]$, se e soltanto se il polinomio x non divide $g(x)$.*

Dim. È immediato osservare che se x divide $g(x)$, allora $g(0) = 0$ e quindi $g(x)$ non può dividere $x^n - 1$. Viceversa, supponiamo che x non divida $g(x)$ e proviamo che allora esiste $n \in \mathbb{N}$ tale che $g(x)$ divide $x^n - 1$. Poichè il polinomio x è relativamente primo con $g(x)$, l'immagine di x, nell'anello finito $\mathcal{R}_g^q$, ha ordine finito, n. Allora:

$$x^n \equiv 1 \pmod{g(x)}.$$

da cui la tesi. □

Possiamo ora definire l'ordine di un polinomio:

Definizione 1.9. *Sia $f(x) \in \mathbb{F}_q[x]$, tale che $f(0) \neq 0$. Definiamo l'*ordine di $f(x)$ *come il più piccolo intero naturale $\mathrm{ord}(f(x))$ tale che $f(x)$ divide il polinomio $x^{\mathrm{ord}(f(x))} - 1$.*

Usando la nozione di ordine di un polinomio, si ottiene:

Teorema 1.10. *Sia $f(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile di grado r tale che $f(0) \neq 0$. Allora $\mathrm{ord}(f(x))$ è uguale all'ordine moltiplicativo di ogni sua radice nel gruppo moltiplicativo $(\mathbb{F}_{q^r})^*$.*

Da 1.7 otteniamo:

Corollario 1.11. *Sia $f(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile di grado $r \geq 2$ allora $\mathrm{ord}(f(x))$ divide $q^r - 1$.*

Per analizzare l'ordine di un polinomio nel caso generale proviamo i seguenti:

Lemma 1.12. *Sia $m \in \mathbb{N}$. $f(x) \in \mathbb{F}_q[x]$, con $f(0) \neq 0$, divide $x^m - 1$, se e soltanto se $\mathrm{ord}(f(x))$ divide m.*

Corollario 1.13. *Siano $m_1, m_2 \in \mathbb{N}$, allora in $\mathbb{F}_q[x]$*

$$\gcd\left(x^{m_1} - 1, x^{m_2} - 1\right) = x^d - 1$$

dove $d = \gcd(m_1, m_2)$.

Teorema 1.14. *Sia $g(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile con $g(0) \neq 0$ e sia $f(x) = g(x)^k$ con $k \in \mathbb{N}_+$. Sia t il più piccolo intero con $p^t \geq k$, dove p è la caratteristica del campo. Allora*

$$\mathrm{ord}(f(x)) = \mathrm{ord}(g(x))p^t.$$

Dim. Per il lemma 1.12, è chiaro che $\mathrm{ord}(g(x))$ divide $\mathrm{ord}(f(x))$; inoltre, per ipotesi, $f(x)$ divide $(x^{\mathrm{ord}(g(x))} - 1)^k$ che a sua volta divide

$$(x^{\mathrm{ord}(g(x))} - 1)^{p^t} = x^{\mathrm{ord}(g(x))p^t} - 1,$$

quindi l'$\mathrm{ord}(f(x))$ divide $\mathrm{ord}(g(x))p^t$ ed esiste $u \in \mathbb{N}$ con $0 \leq u \leq t$ tale che $\mathrm{ord}(f(x)) = \mathrm{ord}(g(x))p^u$.

Inoltre, poiché g è irriducibile, $\mathrm{ord}(g(x))$ non è multiplo di p e quindi il polinomio $x^{\mathrm{ord}(g(x))} - 1$ possiede solo radici semplici. Questo vuol dire che la radici di

$$x^{\mathrm{ord}(g(x))p^u} - 1 = (x^{\mathrm{ord}(g(x))} - 1)^{p^u}$$

sono le stesse di quelle del polinomio $x^{\mathrm{ord}(g(x))} - 1$ e hanno molteplicità p^u.
Siccome $g(x)^k$ divide $x^{\mathrm{ord}(g(x))p^t} - 1$, allora $p^u \geq k$ e quindi $u = t$. □

Teorema 1.15. *Siano $g_1(x), \ldots, g_k(x) \in \mathbb{F}_q[x]$ dei polinomio non nulli, a coppie relativamente primi, e sia $f(x) = \prod_{i=1}^{k} g_i(x)$. Allora*

$$\mathrm{ord}(f(x)) = \mathrm{lcm}\left(\mathrm{ord}(g_1(x)), \ldots, \mathrm{ord}(g_k(x))\right).$$

Dim. Basta che consideriamo il caso in cui $g_i(0) \neq 0$ con $i = 1, \ldots, k$ siccome l'altro caso è banale. Sia $c = \text{lcm}\,(\text{ord}(g_1(x)), \ldots, \text{ord}(g_k(x)))$. Allora siccome $g_i(x)$ divide $x^{\text{ord}(g_i(x))} - 1$ allora divide anche $x^c - 1$. Siccome i $g_i(x)$ sono relativamente primi a coppie, allora $f(x)$ divide $x^c - 1$. Per il lemma 1.12 allora $\text{ord}(f(x))$ divide c.
Viceversa, c divide $\text{ord}(f(x))$ siccome $f(x)$ divide $x^c - 1$. Segue che i $g_i(x)$ dividono $x^c - 1$, e quindi $\text{ord}(g_i(x))$ divide c. □

Teorema 1.16. *Sia $\mathbb{F}_q$ un campo di caratteristica p, e sia $f(x) \in \mathbb{F}_q[x]$ un polinomio di grado positivo e con $f(0) \neq 0$. Sia*

$$f(x) = af_1(x)^{b_1} \cdots f_k(x)^{b_k}$$

la decomposizione di $f(x)$ in polinomi monici e irriducibili.
Allora

$$\text{ord}(f(x)) = ep^t$$

con $e = \text{lcm}(\text{ord}(f_1(x)), \ldots, \text{ord}(f_k(x)))$ e t il più piccolo numero naturale tale che $p^t \geq \max(b_1, \ldots, b_k)$.

Concludendo otteniamo:

Teorema 1.17. *Sia $f(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile , tale che $f(0) \neq 0$. Allora*

$$e = \text{ord}(f(x)) \iff x^e \equiv 1 \pmod{f(x)}.$$

Osservazione. Calcolo dell'ordine di un Polinomio.
Sia $f(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile di grado m, tale che $f(0) \neq 0$. Per quanto visto si ha che $\text{ord}(f(x))$ divide $q^m - 1$. Sia

$$q^m - 1 = \prod_{j=0}^{s} p_j^{r_j}$$

la fattorizzazione di $q^m - 1$.
Per calcolare l'ordine di $f(x)$ si può procedere in modo iterativo.
Sia $1 \leq j \leq s$ e sia $n := \dfrac{q^m - 1}{p_j}$, si consideri $g \equiv x^n - 1 \pmod{f(x)}$. Si ha un delle due seguenti possibilità:

1. $g \neq 0$: in questo caso si passa al j successivo;
2. $g = 0$: in questo caso $\mathrm{ord}(f(x))$ non è multiplo di $p_j^{r_j}$ e quindi $\mathrm{ord}(f(x)) \mid \dfrac{q^m - 1}{p_j}$. Si calcola allora di nuovo g, sostituendo n con $\dfrac{n}{p_j}$, fino a quando il nuovo g soddisfa 1.

Una volta esauriti i primi della fattorizzazione di $q^m - 1$ si ottiene l'ordine cercato.

Definizione 1.18. *Un polinomio $f(x) = \sum_{i=0}^{m} f_i x^i \in \mathbb{F}_q[x]$ di grado m si dice primitivo se soddisfa le seguenti proprietà:*

1. $f_m = 1$,
2. $f_0 \neq 0$,
3. $\mathrm{ord}(f(x)) = q^m - 1$.

1.3 Codici ciclici

Questo lavoro si concentra su una particolare classe di codici dedicati al rilevamento di errori. La classe di codici trattati è quella dei codici ciclici accorciati definiti a partire dai codici ciclici.

Iniziamo col definire i codici ciclici e diamo tutte le nozioni necessarie per affrontare lo studio dei codici ciclici accorciati.

Per far ciò dedichiamo una sottosezione al teorema di MacWilliams, come riportato in [MS]. Questo teorema mette in relazione la distribuzione dei pesi del codice originale con quella del suo duale. Un'altra sottosezione affronta il calcolo della *undetected error probability*, P_{ue}, come troviamo in [CW]. Questa è una proprietà fondamentale dei codici per il rilevamento di errori.

In tutta la sezione indicheremo con p un numero primo, con $q = p^\delta$, $\delta \in \mathbb{N}_+$ e $n \in \mathbb{N}_+$. Inoltre, se $g(x) \in \mathbb{F}_q[x]$, utilizzeremo la notazione $\mathcal{R}_g^q$ per indicare l'anello $\mathbb{F}_q[x]/(g(x))$.

Definizione 1.19. *Un* codice lineare *C di lunghezza n e dimensione k è un sottospazio vettoriale di dimensione k dello spazio vettoriale $\mathbb{F}_q^n$. Gli elementi del codice lineare sono detti* parole *del codice C.*

Per definire un codice ciclico dobbiamo introdurre l'applicazione di *shift* su $\mathbb{F}_q^n$.

Definizione 1.20. *Definiamo come* shift *la mappa*

$$\pi : \mathbb{F}_q^n \to \mathbb{F}_q^n \qquad \textit{tale che} \qquad \pi\big((c_0, \ldots, c_{n-2}, c_{n-1})\big) = (c_{n-1}, c_0, \ldots, c_{n-2}).$$

Osservazione. π è una trasformazione lineare.

Definizione 1.21. *Un codice lineare $C \subset \mathbb{F}_q^n$ è detto* ciclico *se $\pi(c) \in C$ per ogni $c \in C$.*

Osservazione. L' applicazione

$$\sigma : (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n \longrightarrow c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \in \mathcal{R}_{x^n-1}^q.$$

definisce un isomorfismo fra gli spazi vettoriali $\mathbb{F}_q^n$ e $\mathcal{R}_{x^n-1}^q$.

Per semplicità di notazione indicheremo sempre con c una parola del codice; se necessario indicheremo $c \in \mathbb{F}_q^n$, con $[c]$ e con $c(x)$ la parola corrispondente in $\mathcal{R}_{x^n-1}^q$.

Osservazione. L'applicazione π corrisponde, nella rappresentazione polinomiale, all'applicazione di moltiplicazione per x:

$$\pi_x(c(x)) = x \cdot c(x) \pmod{x^n - 1}.$$

Da questa osservazione segue facilmente che:

Teorema 1.22. *Un codice lineare C in $\mathbb{F}_q^n$ è ciclico se e soltanto se C è un ideale di $\mathcal{R}_{x^n-1}^q$.*

Corollario 1.23. *Sia $g(x) \in \mathbb{F}_q[x]$, allora sono fatti equivalenti:*

- *$g(x)$ è il polinomio generatore di un codice ciclico di lunghezza n*
- *$g(x)$ divide $x^n - 1$,*

Definizione 1.24. *Dato un codice ciclico C, il polinomio monico che genera C, come ideale in $\mathcal{R}_{x^n-1}^q$, è chiamato* polinomio generatore *di C.*

Il seguente teorema connette il grado del polinomio generatore alla dimensione del codice:

Teorema 1.25. *Sia C un codice ciclico di lunghezza n e sia $g(x)$ il suo polinomio generatore. Se il grado di $g(x)$ è r, allora*

1. *C ha dimensione $n - r$,*
2. *$\{g(x), xg(x), \ldots, x^{(n-r)-1}g(x)\}$ è una base di $C \subset \mathcal{R}^q_{x^n-1}$,*

Osservazione. Per 1.8 si ha che ogni polinomio con termine noto non nullo può essere il generatore di un codice ciclico.

Introduciamo ora il codice duale di un codice lineare. Quello duale è un codice strettamente legato a quello originale.

In certi casi, lo studio delle proprietà del duale, permette di trarre conclusioni sulle caratteristiche del codice originale. Diamo quindi le seguenti definizioni per caratterizzare il codice duale.

Definizione 1.26. *Dato un codice lineare C di lunghezza n, ogni matrice le cui righe formino una base per C viene chiamata* matrice generatrice *di C.*

Osservazione. Per il teorema 1.25, se un codice ciclico C ha polinomio generatore $g(x) = x^r + \sum_{i=0}^{r-1} g_i x^i$, allora una sua matrice generatrice è la matrice $(n-r)\times(n)$

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ \vdots \\ x^{(n-r)-1}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & \cdots & g_{r-1} & 1 & & & & & \\ & g_0 & g_1 & \cdots & g_{r-1} & 1 & & & & \\ & & g_0 & g_1 & \cdots & g_{r-1} & 1 & & & \\ & & & \ddots & & & \ddots & & & \\ & & & & \ddots & & & \ddots & & \\ & & & & & g_0 & g_1 & \cdots & g_{r-1} & 1 \end{pmatrix}$$

Possiamo ora definire il codice duale:

Definizione 1.27. *Il codice duale $C^\perp$ del codice lineare C è l'insieme di tutti gli elementi $[v] = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$ tali che $\langle v, w\rangle = \sum_{i=1}^n v_i w_i = 0$ per ogni $w \in C$.*

Osservazione. Per ogni codice lineare $C \subset \mathbb{F}_q^n$ vale la seguente relazione:

$$\dim C + \dim C^\perp = n$$

Definizione 1.28. *H è una* matrice di parità *del codice lineare C se le sue colonne sono elementi della base di $C^\perp$.*

Teorema 1.29. *Se G e H sono rispettivamente una matrice generatrice e una di parità di un codice lineare C, allora $GH = 0$.*

Da questo teorema seguono due corollari utili.

Corollario 1.30. *G è una matrice generatrice di C se e soltanto se G^T è una matrice di parità di $C^\perp$.*

Corollario 1.31. *Sia H una matrice di parità del codice C, allora $c \in C$ se e soltanto se $c^T \cdot H = 0$.*

Un teorema facilmente dimostrabile ma utile in teoria dei codici è il seguente.

Teorema 1.32. *Il codice duale di un codice ciclico è ciclico.*

Teorema 1.33 (Duale di un Codice Ciclico). *Sia C un codice ciclico di lunghezza n generato dal polinomio $g(x) = g_0 + g_1 x + \cdots + g_{r-1}x^{r-1} + x^r$, allora $c \in C^\perp$ se e soltanto se le sue componenti soddisfano la relazione*

$$c_i = -g_0 c_{i-r} - \cdots - g_{r-1} c_{i-1} \qquad \forall i = r, \ldots, n-r.$$

Dim. Dai corollari 1.30 e 1.31

$$c \in C^\perp \iff G \cdot c = 0$$

dove G è una qualsiasi matrice generatrice per C.
Se come G prendiamo la matrice costruita a partire dal polinomio generatore $g(x)$,

allora

$$G \cdot c = \begin{pmatrix} g_0 & g_1 & \dots & g_{r-1} & 1 & & & & \\ & g_0 & g_1 & \dots & g_{r-1} & 1 & & & \\ & & g_0 & g_1 & \dots & g_{r-1} & 1 & & \\ & & & \ddots & & & & \ddots & \\ & & & & \ddots & & & & \ddots \\ & & & & & g_0 & g_1 & \dots & g_{r-1} & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ \vdots \\ \vdots \\ c_{n-1} \end{pmatrix} = 0$$

$$\Rightarrow \quad \begin{cases} c_0 g_0 + c_1 g_1 + \dots + c_{r-1} g_{r-1} + c_r = 0 \\ \quad \vdots \\ c_{n-r-1} g_0 + c_{n-r} g_1 + \dots + c_{n-2} g_{r-1} + c_{n-1} = 0 \end{cases}$$

da cui la tesi. □

Nello studio delle proprietà di un codice sono molto importanti le definizioni di peso di una parola e di distanza tra due di esse. Queste sono applicazioni definite su tutto $\mathbb{F}_q^n$, ma che applicate ad un codice diventano di fondamentale utilizzo in sede di rilevamento o di correzione di errori.

Definiamo ora peso e distanza per gli elementi di $\mathbb{F}_q^n$.

Definizione 1.34. *Il* peso di Hamming *di un elemento $c \in \mathbb{F}_q^n$ è uguale al numero di componenti di c diverse da zero, e cioè*

$$wt(c) = \#\{0 \leq i \leq n \mid c_i \neq 0\}.$$

Definiamo come distanza di Hamming *l'applicazione da $\mathbb{F}_q^n \times \mathbb{F}_q^n$ in $\mathbb{N}$ che a due parole di $\mathbb{F}_q^n$ associa il numero di componenti nelle quali le due parole differiscono,*

$$d(v, w) = \{0 \leq i \leq n | v_i \neq w_i,\ v, w \in \mathbb{F}_q^n\}.$$

Definiamo infine distanza minima *di $U \subset \mathbb{F}_q^n$ come il minimo delle distanze tra coppie di elementi di U*

$$d_{min}^U = \min\{d(v, w) | v, w \in U\}.$$

Osservazione. Per come è stata definita la distanza di Hamming

$$d(v,w) = wt(v-w), \qquad \forall v,w \in C.$$

Questo implica che, per i codici lineari, la distanza minima verifica la seguente relazione:

$$\begin{aligned} d^C_{min} &= \min\{d(v,w)|v,w \in C,\ v \neq w\} = \\ &= \min\{wt(v-w)|v,w \in C,\ v \neq w\} = \\ &= \min\{wt(v)|v \in C, v \neq 0\} \end{aligned}$$

dove nell'ultimo passaggio abbiamo utilizzato la linearità del codice.

Nella prossima sottosezione enunceremo e dimostreremo il teorema di MacWilliams. Dobbiamo fornire però alcune definizioni che utilizziamo sia in questo teorema, che nella sottosezione dedicata allo studio della P_{ue}.

Definizione 1.35. *Sia C un codice lineare di lunghezza n.*
Indichiamo con A_i il numero di parole del codice C di peso i con $i = 0, \ldots, n$, quindi

$$A_i = \#\{c \in C|wt(c) = 1\}.$$

Definiamo inoltre la distribuzione dei pesi *del codice C come*

$$\{A_0, A_1, \ldots, A_n\}.$$

Il weight enumerator *di C sarà il polinomio in due variabili*

$$W_C(x,y) = \sum_{i=0}^{n} A_i x^{n-i} y^i.$$

1.3.1 Teorema di MacWillams

Il Teorema di MacWilliams è molto importante in teoria dei codici perchè mette in relazione i *weight enumerator* di un codice con quello del suo duale.

Questa relazione permette, nel calcolo della distribuzione dei pesi, il passaggio da un codice al suo duale. Il teorema è molto utilizzato nel momento in cui il

numero di parole di uno dei due codici, originale o duale, risulti essere molto inferiore rispetto a quello dell'altro.

Questo teorema quindi è importante perchè permette una scelta del codice da analizzare che può rendere molto inferiore il costo computazionale di certe proprietà del codice.

Teorema 1.36 (MacWilliams). *Sia* $C \subset {}^{\mathbb{F}_q[x]}/_{(x^n-1)}$ *un codice lineare di lunghezza* n *e dimensione* k *e sia* $C^\perp$ *il suo codice duale, allora*

$$W_{C^\perp}(x,y) = \frac{1}{|C|} W_C(x+(q-1)y, x-y) \tag{1.1}$$

dove $|C| = q^k$ *è il numero di parole di* C*.*

Per semplicità ci concentriamo nel caso in cui $q = 2$, quindi in quello dei codici binari.

Per dimostrare questo teorema dobbiamo prima dimostrare un lemma importante. Sia f una mappa su $\mathbb{F}_2^n$, definiamo la *trasformata di Hadamard* $\hat{f}$ di f come

$$\hat{f}(u) = \sum_{v \in \mathbb{F}_2^n} (-1)^{\langle u,v \rangle} f(v), \qquad u \in \mathbb{F}_2^n.$$

dove con $\langle u, v \rangle$ indichiamo la somma dei prodotti componente per componente.

Lemma 1.37. *Sia* C *un codice binario di lunghezza* n *e dimensione* k*, allora*

$$\sum_{u \in C^\perp} f(u) = \frac{1}{|C|} \sum_{u \in C} \hat{f}(u).$$

Dim.

$$\sum_{u \in C} \hat{f}(u) = \sum_{u \in C} \sum_{v \in \mathbb{F}_2^n} (-1)^{\langle u,v \rangle} f(v) = \sum_{v \in \mathbb{F}_2^n} f(v) \sum_{u \in C} (-1)^{\langle u,v \rangle}.$$

Ora se $v \in C^\perp$, $\langle u, v \rangle = 0$ e quindi la somma interna è $|C|$. Se invece $v \notin C^\perp$ allora al variare di v in $\mathbb{F}_2^n$ il numero di 1 e di 0 negli esponenti sono uguali, quindi la somma interna sarà nulla. Allora, applicando i ragionamenti precedenti

$$\sum_{u \in C} \hat{f}(u) = |C| \sum_{u \in C^\perp} f(u).$$

□

Dim. [MacWilliams con $q = 2$]
Basterà applicare il lemma precedente alla mappa

$$f(u) = x^{n-wt(u)}y^{wt(u)}.$$

Infatti, otteniamo

$$\hat{f}(u) = \sum_{v\in\mathbb{F}_2^n}(-1)^{\langle u,v\rangle}x^{n-wt(v)}y^{wt(v)}.$$

Siano quindi ora $u = (u_1, \ldots, u_n)$ e $v = (v_1, \ldots, v_n)$, allora

$$\begin{aligned}\hat{f}(u) &= \sum_{v\in\mathbb{F}_2^n}(-1)^{u_1v_1+\cdots+u_nv_n}\prod_{i=1}^{n}x^{1-v_i}y^{v_i} = \\ &= \sum_{v_1=0}^{1}\cdots\sum_{v_n=0}^{1}\prod_{i=1}^{n}(-1)^{u_iv_i}x^{1-v_i}y^{v_i} = \\ &= \prod_{i=1}^{n}\sum_{w=0}^{1}(-1)^{u_iw}x^{1-w}y^{w}.\end{aligned}$$

Se $u_i = 0$, allora la somma interna sarà $x + y$, mentre se $u_i = 1$ sarà $x - y$, quindi

$$\hat{f}(u) = (x + y)^{n-wt(u)}(x - y)^{wt(u)}.$$

Riscrivendo il tutto otteniamo

$$\sum_{u\in C^\perp}x^{n-wt(u)}y^{wt(u)} = \frac{1}{|C|}\sum_{u\in C}(x + y)^{n-wt(u)}(x - y)^{wt(u)}.$$

□

Dal Teorema di MacWilliams, possiamo ricavare direttamente una relazione tra gli elementi della distribuzione dei pesi dei due codici. Dobbiamo però definire i polinomi di Krawtchouk. Daremo la definizione di questi polinomi nel caso q potenza di un primo.

Definizione 1.38. *Fissato un intero positivo n e q la potenza di un primo, definia-*

mo, per ogni $k = 0, \ldots, n$*, i* polinomi di Krawtchouk $\mathcal{P}_k(x; n)$ *come*

$$\mathcal{P}_k(x) = \mathcal{P}_k(x; n) = \sum_{j=0}^{k} (-1)^j (q-1)^{k-j} \binom{i}{j} \binom{n-x}{k-j}$$

dove x *è un'indeterminata.*

Dal teorema di MacWilliams e dai polinomi di Krawtchouk segue direttamente la seguente relazione tra le distribuzioni di pesi di C e $C^\perp$ rispettivamente.

Corollario 1.39. *Se* $\{A_i\}$ *e* $\{A'_i\}$ *sono rispettivamente le distribuzioni di pesi di un codice lineare* C *e del suo duale, allora vale la seguente relazione*

$$A'_k = \frac{1}{|C|} \sum_{i=0}^{n} A_i \mathcal{P}_k(i). \tag{1.2}$$

Siamo riusciti quindi ad ottenere una relazione tra le due distribuzioni dei pesi che risulta essere più facile da un punto di vista implementativo e computazionalmente più efficiente. Infatti possiamo evitare il passaggio attraverso i polinomi in due variabili, quali i weight enumerator. Per migliorare le prestazioni del nostro algoritmo ricaviamo una ricorrenza che calcoli i polinomi di Krawtchouk.

Definizione 1.40. *Sia* $\mathcal{R}$ *un anello e sia* $\mathfrak{s} = (s_i)_{i \in \mathbb{N}}$ *una successione contenuta in* $\mathcal{R}$*. Definiamo la* funzione generatrice *della successione come l'espressione formale*

$$S(z) = \sum_{i=0}^{\infty} s_i z^i.$$

Si può verificare con semplici calcoli che presa la successione dei polinomi di Krawtchouk $(\mathcal{P}_k(x))_{k \in \mathbb{N}}$ allora la sua funzione generatrice è

$$(1 + (q-1)z)^{n-x}(1-z)^x = \sum_{k=0}^{\infty} \mathcal{P}_k(x) z^k. \tag{1.3}$$

Teorema 1.41. *I polinomi di Krawtchouk su* $\mathbb{F}_q[x]$ *soddisfano la seguente ricorrenza*

$$(k+1)\mathcal{P}_{k+1}(x) = [(n-k)(q-1) + k - qx]\,\mathcal{P}_k(x) - (q-1)(n-k+1)\mathcal{P}_{k-1}(x).$$

per $k \in \mathbb{N}_+$*, con* $\mathcal{P}_0(x) = 1$ *e* $\mathcal{P}_1(x) = (q-1)n - qx$*.*

Dim. Nella dimostrazione assumiamo $\gamma := q - 1$.

Deriviamo rispetto a z la funzione generatrice di $(\mathcal{P}_k(x))_{k\in\mathbb{N}}$.

$$\begin{aligned}\frac{d}{dz}\left((1+\gamma z)^{n-x}(1-z)^x\right) &= \gamma(n-x)(1+\gamma z)^{n-x-1}(1-z)^x + \\ &\quad - x(1+\gamma z)^{n-x}(1-z)^{x-1} = \\ &= (1+\gamma z)^{n-x-1}(1-z)^{x-1}[\gamma n - qx - \gamma n x]\end{aligned}$$

mentre

$$\frac{d}{dz}\left(\sum_{k=0}^{\infty}\mathcal{P}_k(x)z^k\right) = \sum_{k=1}^{\infty} k\mathcal{P}_k(x)z^{k-1}.$$

Moltiplicando ora entrambe le derivate per

$$(1+\gamma z)(1-z) = 1 + (\gamma-1)z - \gamma z^2$$

otteniamo a sinistra

$$\begin{aligned}&(1+\gamma z)(1-z)\left[\frac{d}{dz}\left((1+\gamma z)^{n-x-1}(1-z)^{x-1}\right)\right] = \\ &= (1+\gamma z)^{n-x}(1-z)^x[\gamma n - qx - \gamma n x] = \\ &= \left(\sum_{k=0}^{\infty}\mathcal{P}_k(x)z^k\right)[\gamma n - qx - \gamma n z] = \\ &= \sum_{k=0}^{\infty}(\gamma n - qx)\mathcal{P}_k(x)z^k - \sum_{k=0}^{\infty}\gamma n\mathcal{P}_k(x)z^{k+1}\end{aligned}$$

mentre a destra

$$(1+\gamma z)(1-z)\left(\sum_{k=1}^{\infty} k\mathcal{P}_k(x)z^{k-1}\right) =$$
$$= (1+(\gamma-1)z-\gamma z^2)\sum_{k=0}^{\infty} k\mathcal{P}_k(x)z^{k-1} =$$
$$= \sum_{k=1}^{\infty} k\mathcal{P}_k(x)z^{k-1} + \sum_{k=1}^{\infty} k(\gamma-1)\mathcal{P}_k(x)z^{k} - \sum_{k=1}^{\infty} k\gamma\mathcal{P}_k(x)z^{k+1}.$$

Uguagliando i coefficienti di z^k otteniamo con $k \geq 1$

$$(\gamma n - qx)\mathcal{P}_k(x) - \gamma n\mathcal{P}_{k-1}(x) =$$
$$= (k+1)\mathcal{P}_{k+1}(x) + k(\gamma-1)\mathcal{P}_k(x) - (k-1)\gamma\mathcal{P}_{k-1}(x)$$

e quindi

$$(k+1)\mathcal{P}_{k+1}(x) = [\gamma(n-k)+k-qx]\,\mathcal{P}_k(x) + \gamma(n-k+1)\mathcal{P}_{k-1}(x).$$

□

1.3.2 Codici Ciclici Accorciati

Come già anticipato, in questo lavoro trattiamo i codici ciclici accorciati. Questi codici, in letteratura, vengono anche detti codici a ridondanza ciclica o "Cyclic Redundancy-Check Codes", e li indicheremo con CRC.

I CRC sono dei codici lineari derivanti direttamente dai codici ciclici. Essi sono codici particolarmente comodi perchè, nonostante non mantengano la struttura ciclica, ereditano varie proprietà dei codici ciclici da cui sono generati. Nel nostro caso sfrutteremo la caratterizzazione delle parole del codice duale tramite il polinomio generatore, come dimostriamo in seguito, per il calcolo della distribuzione dei pesi.

Vediamo più da vicino come si costruisce un codice CRC e dimostriamone le proprietà principali.

Il processo di accorciamento di un codice C in un codice $C^{\#}$ è un procedimento che consiste nel ridurre la lunghezza del codice originale per ottenere un

altro codice. Per far ciò si prendono in considerazione tutte le parole del codice C che iniziano con s zeri consecutivi e, cancellando questi zeri, si ottengono le parole che costituiranno le parole del codice $C^{\#}$. Il codice ottenuto avrà quindi lunghezza $n - s$, dove ricordiamo che n è la lunghezza del codice originale C. Chiameremo *s-accorciati* i codici ottenuti accorciando il codice di base di s componenti.

Dimostrare che la struttura di codice lineare si conserva quando si accorcia un codice lineare è immediato.

Lemma 1.42. *Sia C un codice lineare di lunghezza n allora $C^{\#}$, ottenuto accorciando il codice di s componenti, è un codice lineare.*

Osservazione. Per il teorema 1.23 il codice $C^{\#}$ ottenuto accorciando un codice ciclico C, in generale non è più un codice ciclico.

La praticità nel lavorare con i codici accorciati consiste nel fatto che la distanza minima non diminuisce.

Lemma 1.43. *Sia C un codice lineare e $C^{\#}$ un suo codice accorciato, allora $d^{C}_{min} \leq d^{C^{\#}}_{min}$*

Dim.

$$\begin{aligned} d^{C^{\#}}_{min} &= \min\{d(v^{\#}, w^{\#}) | v^{\#}, w^{\#} \in C^{\#},\ v^{\#} \neq w^{\#}\} = \\ &= \min\{d(v, w) | v, w \in C_S,\ v \neq w\} \geq \\ &\geq \min\{d(v, w) | v, w \in C,\ v \neq w\} = d^{C}_{min}. \end{aligned}$$

□

Osservazione. Se analizziamo i procedimenti adottati per accorciare un codice nel caso di rappresentazione polinomiale, allora, se $c \in C$ è la parola corrispondente alla parola $c^{\#} \in C^{\#}$, vale la seguente relazione in $\mathcal{R}^{q}_{x^n-1}$

$$c(x) \equiv x^s c^{\#}(x) \pmod{x^n - 1}.$$

Quindi, siccome $g(x)$ divide $c(x)$ e $\gcd(g(x), x) = 1$, $g(x)$ divide anche $c^{\#}(x)$. Quindi anche tutti gli elementi di $C^{\#}$ sono caratterizzati dal fatto di essere multipli di $g(x)$, da cui segue il seguente teorema.

Vediamo ora di studiare più a fondo le proprietà di questi codici. Come anticipato, non sono codici ciclici, ma ereditano delle importanti proprietà da quest'ultimi.

Anch'essi hanno un polinomio generatore che è lo stesso del codice ciclico da cui derivano. Inoltre, le parole del codice duale possono essere espresse considerando una particolare relazione dipendente dal polinomio generatore. Esplicitiamo tali osservazioni.

Teorema 1.44. *Siano C un codice ciclico di lunghezza n, $g(x)$ il polinomio generatore di grado r e $C^{\#}$ il codice s-accorciato, allora*

$$\{g(x), xg(x), \ldots, x^{n-s-r-1}g(x)\}$$

è una base del codice $C^{\#}$.

Corollario 1.45. *Sia C un codice ciclico generato da un polinomio $g(x)$ di grado r e sia $C^{\#}$ un suo codice s-accorciato. Allora una matrice generatrice del codice s-accorciato è la matrice $(n-s-r)\times(n-s)$*

$$G^{\#} = \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{(n-s-r)-1}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & \ldots & g_{r-1} & 1 & & \\ & g_0 & g_1 & \ldots & g_{r-1} & 1 & \\ & & \ddots & & & \ddots & \\ & & & g_0 & g_1 & \ldots & g_{r-1} & 1 \end{pmatrix}$$

Corollario 1.46. *Sia $C^{\#}$ un codice s-accorciato ottenuto da un codice ciclico C con generatore $g(x)$ di grado r, allora le componenti di $c \in \left(C^{\#}\right)^{\perp}$ soddisfano la relazione*

$$c_i = -g_0 c_{i-r} - \cdots - g_{r-1}c_{i-1} \qquad \forall i = r, \ldots, n-s-r$$

Dim. Si veda dimostrazione del teorema 1.33. □

1.3.3 Undetected Error Probability

Nello studio dei codici atti al rilevamento di errori una tra le proprietà importanti del codice è la *Undetected Error Probability*, P_{ue}. Questa sottosezione è

dedicata alla definizione di questa proprietà e alla formulazione di tale probabilità rispetto alla distribuzione dei pesi del codice.

Spendiamo in questa sottosezione alcune parole per capire cosa sono gli errori. Sia $C \in \mathbb{F}_q^n$ un codice e $c \in C$ una sua parola. Un *errore* è l'evento per il quale la parola ricevuta risulta differente dalla parola inviata in qualche sua componente. Se quindi indichiamo con $r \in \mathbb{F}_q^n$ la parola ricevuta, l'errore è rappresentato da $e = r - c \in \mathbb{F}_q^n$. Nel caso in cui la parola ricevuta non risulti essere una parola del codice, cioè $r \notin \mathbb{F}_q^n$, allora avremo scoperto la presenza di un errore.

Affrontiamo il caso in cui la parola ricevuta, nonostante l'avvento di alcuni errori nella trasmissione, sia contenuta nel codice. Quindi il caso in cui $r \in C$ e, siccome lavoriamo con codici lineari, in cui allora $e \in C$. Chiameremo *Undetected Error Probability* (P_{ue}) la probabilità di questo evento. Per quanto detto tale probabilità sarà uguale alla probabilità che l'errore e sia una parola del codice.

Indichiamo con ϵ la probabilità di errore limitato ad una sola componente. Quindi $\epsilon = P(e = e_i)$ dove $1 \leq i \leq n$ e $\{e_i \in \mathbb{F}_q^n | 1 \leq i \leq n\}$ è la base canonica di $\mathbb{F}_q^n$.

Vediamo come calcolare la P_{ue} attraverso la distribuzione di pesi del codice lineare e del suo codice duale. Questa parte di teoria si trova in [CW]. Nel nostro caso lavoreremo con codici lineari di lunghezza n e dimensione k.

Il mezzo attraverso il quale spediamo un messaggio è detto *canale*. Nel caso che la probabilità di errore ϵ non dipenda dalla componente della parola spedita, allora il canale sarà detto *simmetrico*. Infine il canale è detto *binario* se vengono trasmessi solamente elementi di $\mathbb{F}_2$.

Iniziamo con l'ottenere la P_{ue} per codici binari. Estenderemo quindi i risultati al caso di codici sul campo $\mathbb{F}_q$.

Siccome lavoriamo per ipotesi su un canale binario simmetrico, allora possiamo restringere la scelta di ϵ tra 0 e $1/2$.

Supponiamo che $e \in \mathbb{F}_2^n$ rappresenti un errore di peso $w_e = wt(e)$, allora la probabilità che questo evento accada è data da:

$$P(e) = \epsilon^{w_e}(1-\epsilon)^{n-w_e}, \qquad 0 \leq \epsilon \leq 1/2. \tag{1.4}$$

Sia H la matrice di parità del codice, allora per il teorema 1.31, $e \in C$ se e soltanto

se $He = 0$. Sia $E = \{e \in \mathbb{F}_2^n | He = 0\} \setminus \{e = 0\}$, cioè l'evento in cui $e \in C \setminus \{0\}$, allora $P(E)$ sarà la somma delle probabilità $P(e)$ al variare di tutte le parole del codice, quindi

$$P_{ue} = P(E) = \sum_{e \in C \setminus \{0\}} \epsilon^{w_e}(1-\epsilon)^{n-w_e} = \sum_{i=1}^{n} A_i \epsilon^i (1-\epsilon)^{n-i}.$$

L'espressione della P_{ue} dipende quindi dalla distribuzione dei pesi di un codice. Molto spesso accade però che il numero di parole di un codice sia molto più alto di quello del suo codice duale, quindi computazionalmente più costoso. Come nel nostro caso, ci si concentra quindi nella ricerca della distribuzione del codice duale. Vediamo quindi di seguito una formulazione della P_{ue} rispetto alla distribuzione del codice duale.

Per questo caso sviluppiamo un ragionamento un po' più elaborato. Sia

$$H^* \in \mathcal{M}_{\mathbb{F}_2}(2^{n-k}, n)$$

la matrice le cui righe sono tutte le parole del codice duale.

Lemma 1.47. *$H^*e = 0$ se e soltanto se $He = 0$. Se $H^*e \neq 0$ allora il vettore ottenuto avrà tanti 0 quanti 1.*

Dim. La prima parte segue direttamente dal fatto che H contiene una base del codice duale. Per quanto riguarda la seconda parte invece, se $H^*e \neq 0$, allora esiste $v \in C^\perp$ tale che $\langle e, v \rangle = 1$. Allora valgono dato $w \in C^\perp$,

- se $\langle e, w \rangle = 0$, allora $\langle e, w + v \rangle = 1$
- invece se $\langle e, w \rangle = 1$ allora $\langle e, w + v \rangle = 0$,

quindi, ad ogni parola di $C^\perp$ con prodotto nullo possiamo associarne una con prodotto 1 e viceversa. □

Definiamo ora con E_j, con $j = 0, \ldots, 2^{n-k}$, l'evento che la j-esima componente di H^*e sia uguale a 1. Per il lemma precedente allora

$$E' = \{e \in \mathbb{F}_2^n | He = 0\} = \{e \in \mathbb{F}_2^n | H^*e = 0\},$$

quindi

$$P(E') = 1 - P(E_1 \cup \cdots \cup E_{2^{n-k}}).$$

Sempre per il lemma precedente, se $H^*e \neq 0$, esattamente 2^{n-k-1} eventi di tipo E_j sono verificati, mentre nel caso in cui $H^*e = 0$ nessuno degli eventi E_j è verificato.
Rappresentando l'unione degli eventi E_j l'evento $\{e \in \mathbb{F}_2^n | H^*e \neq 0\}$, allora

$$\sum_{j=0}^{2^{n-k}} P(E_j) = 2^{n-k-1} P(E_1 \cup \cdots \cup E_{2^{n-k}})$$

quindi

$$P(E') = 1 - \frac{1}{2^{n-k-1}} \sum_{j=0}^{2^{n-k}} P(E_j).$$

Ora vediamo come calcolare $P(E_j)$. Se la j-esima riga di H^* ha peso w_j, allora $P(E_j)$ è la probabilità che il vettore e abbia un numero dispari di componenti uguali a 1 nelle w_j posizioni corrispondenti alle componenti non zero della j-esima riga di H^*, quindi

$$P(E_j) = \sum_{\substack{i=0 \\ i \ dispari}}^{w_j} \binom{w_j}{i} \epsilon^i (1-\epsilon)^{w_j - i}.$$

Vale la seguente formula

$$\sum_{\substack{i=0 \\ i \ dispari}}^{I} \binom{I}{i} a^i b^{I-i} = \frac{1}{2}[(b+a)^I - (b-a)^I],$$

otteniamo quindi

$$P(E_j) = \frac{1}{2}\big(1 - (1-2\epsilon)^{w_j}\big).$$

Combinando le relazioni otteniamo

$$\begin{aligned} P_{ue} &= P(E) = P(E' \setminus \{e = 0\}) = P(E') - P(e = 0) = \\ &= 1 - \frac{1}{2^{n-k-1}} \sum_{i=0}^{2^{n-k}} \frac{1}{2}\big(1 - (1-2\epsilon)^{w_j}\big) - (1-\epsilon)^n \\ &= \frac{1}{2^{n-k}} \sum_{i=0}^{n} B_i (1-2\epsilon)^i - (1-\epsilon)^n. \end{aligned}$$

Passiamo ora al caso in cui $\mathbb{F}_q$ sia il campo di base, con q la potenza di un primo.

Supporremo di lavorare su un canale simmetrico tale che la probabilità di errore di una singola componente sia ancora ϵ. Poichè supponiamo equiprobabili gli eventi che l'errore effettuato sia un valore in $(\mathbb{F}_q)^*$, ϵ è compreso tra 0 e $(q-1)/q$.

Inoltre si ha che la probabilità che l'errore di una singola componente sia uguale ad un elemento fissato di $(\mathbb{F}_q)^*$ è data da $\epsilon/q - 1$.

Le definizioni date precedentemente, per il caso binario, sono tuttora valide, tranne per quanto riguarda la probabilità del singolo evento $e \in \mathbb{F}_q^n$ che sarà

$$P(e) = \Big(\frac{\epsilon}{q-1}\Big)^{w_e} (1-\epsilon)^{n-w_e}.$$

Ripetendo i ragionamenti fatti per il caso $q = 2$, la probabilità sarà

$$P_{ue} = \sum_{i=1}^{n} A_i (1-\epsilon)^{n-i} \Big(\frac{\epsilon}{q-1}\Big)^i.$$

Vediamo anche nel caso dei codici in $\mathbb{F}_q$ di trovare una relazione della P_{ue} rispetto alla distribuzione del codice duale.

Con H_q^* indicheremo la matrice che contiene tutte le parole del codice duale; essa sarà quindi una matrice appartenente all'insieme $\mathcal{M}_{\mathbb{F}_q}(q^{n-k}, n)$.

Lemma 1.48. *$H_q^* e = 0$ se e soltanto se $H_q e = 0$, dove H_q è la matrice di parità del codice C. Se $H_q^* e \neq 0$, allora ogni elemento di $\mathbb{F}_q$ compare tra le componenti di $H_q^* e$ esattamente q^{n-k-1} volte.*

Dim. La dimostrazione è molto simile a quella scritta precedentemente. L'unica

nota da fare è che nel caso in cui $H_q^* e \neq 0$, allora esiste $v \in C^\perp$ tale che

$$\langle e, v \rangle \in \mathbb{F}_q^*$$

allora se $w \in C^\perp$, si ottengono due casi:

- se $\langle e, w \rangle = 0$, allora $\langle e, w + \lambda v \rangle = \lambda \langle x, v \rangle$ e al variare di λ in $\mathbb{F}_e^*$ si ottiene ogni elemento del campo diverso da zero,
- se $\langle e, w \rangle \neq 0$ allora esiste $\lambda \in \mathbb{F}_e^*$ tale che $\langle e, w + \lambda v \rangle = 0$ quindi ci riduciamo al caso precedente.

□

Indichiamo con E_j l'evento per cui la j-esima componente di $H_q^* e$ sia diversa da zero. Considerando tutti i casi la probabilità dell'evento E_j sarà

$$P(E_j) = \sum_{k=1}^{w_j} \left[\binom{w_j}{k} (1-\epsilon)^{w_j - k} \left(\frac{\epsilon}{q-1} \right)^k \cdot \sum_{i=1}^{k} (-1)^{k-i} (q-1)^i \right]$$

dove w_j è il peso della riga j-esima di H_q^*. Sviluppando otteniamo

$$P(E_j) = \frac{q-1}{q} \left[1 - \left(1 - \frac{q\epsilon}{q-1} \right)^{w_j} \right].$$

L'unione di tutti gli eventi E_j avrà probabilità

$$P\left(\bigcup_{j=1}^{q^{n-k}} E_j \right) = \frac{1}{(q-1)q^{n-k-1}} \sum_{j=1}^{q^{n-k}} P(E_j).$$

Mettendo insieme le osservazioni fatte finora otteniamo che

$$\begin{aligned} P_{ue} &= P(E \setminus \{e = 0\}) = P(E) - (1-\epsilon)^n = \\ &= q^{-(n-k)} \sum_{i=0}^{n} B_i \left(1 - \frac{q\epsilon}{q-1} \right)^i - (1-\epsilon)^n. \end{aligned}$$

Quando consideriamo dei codici per il rilevamento di errori possiamo verificare un'altra proprietà.

Definizione 1.49. *Un codice lineare $C \subset \mathbb{F}_q^n$, con q potenza di un primo p, è detto* proprio *se P_{ue} è monotona crescente per $\epsilon \in [0, \frac{q-1}{q}]$.*

Quindi i codici si dividono in due classi: propri e non propri. Siccome un codice ha prestazioni migliori al diminuire della P_{ue}, i codici propri sono considerati mogliori per quanto riguarda il ritrovamento di errori. Essi infatti non possiedono punti di massimo locale.

Scriviamo l'enunciato di un teorema che indica una condizione sufficiente affinché un codice sia proprio. Questo risultato è tratto da [Do].

Sia $C \subset \mathbb{F}_q^n$ un codice di distanza minima d_{min}^C e dimensione k. Sia $C^\perp$ il suo codice duale con distribuzione dei pesi

$$\{B_0, B_1, \ldots, B_n\}.$$

Definizione 1.50. *Denotiamo con B_l^* i numeri*

$$B_0^* = 0, \qquad B_l^* = \frac{l!}{n!} \sum_{i=1}^{l} \frac{(n-i)!}{(l-i)!} B_i, \ l = 1, \ldots, n$$

Teorema 1.51. *Sia $C \subset \mathbb{F}_q^n$ un codice lineare con duale $C^\perp$. Se vale*

$$B_{n-l}^* \geq B_{n-l+1}^* - q^{n-l-k}(q-1)$$

con $l = d_{min}^C + 1, \ldots, n$ allora C è un codice proprio.

Capitolo 2

La struttura di $\mathcal{R}_g^q$

In questo capitolo studieremo la struttura dell' anello $\mathcal{R}_g^q := \mathbb{F}_q[x]/(g(x))$, dove $g(x) \in \mathbb{F}_q[x]$, e vedremo come sia possibile costruire una sua decomposizione in componenti invarianti rispetto all'azione della moltiplicazione per x.

Nei capitoli succcessivi, applicheremo i risultati ottenuti per ottenere un metodo efficiente e rapido per calcolare i pesi delle parole di un codice ciclico accorciato (con polinomio generatore $g(x)$).

Piu' precisamente vedremo come sia possibile costruire un insieme finito di polinomi di $\mathcal{R}_g^q$, le cui orbite siano tutte disgiunte e forniscano una decomposizione di $\mathcal{R}_g^q$.

Per esplicitare tali polinomi si applica il Teorema Cinese del Resto. Se il polinomio generatore del codice $g(x)$ si fattorizza come $g(x) = \prod_{i=1}^{m} {g_i}^{e_i}$, si ha:

$$\mathcal{R}_g^q \cong \prod_{i=1}^{m} \mathcal{R}_{g_i^{e_i}}^q.$$

La decomposizione di ognuno dei fattori $\mathcal{R}_{g_i^{e_i}}^q$ con $g_i(x)$ polinomio irriducibile, viene a sua volta studiata decomponendo, come prodotto di gruppi ciclici, i gruppi moltiplicativi $(\mathcal{R}_{g_i^s}^q)^*$ con $0 \leq s \leq t$.

In tutto il capitolo indicheremo con $q = p^\delta$, con p un numero primo e $\delta \in \mathbb{N}_+$ e adotteremo le seguenti notazioni:

- con $\mathcal{R}_g^q$ indicheremo l'anello $\mathbb{F}_q[x]/(g(x))$,
- con M_g^q indicheremo il gruppo moltiplicativo $\left(\mathcal{R}_g^q\right)^*$,

2.1 Preliminari e notazioni

Definizione 2.1. *Sia $g(x) \in \mathbb{F}_q[x]$ e sia $\mu : \mathbb{F}_q[x] \longrightarrow \mathcal{R}_g^q$, data da*

$$\mu(h(x)) = h(x) \pmod{g(x)}.$$

Definiamo l'applicazione grado quoziente *su $\mathcal{R}_g^q$ come l'applicazione*

$$\deg_{qt} : \mathcal{R}_g^q \longrightarrow \mathbb{N}$$

tale che $\deg_{qt} f = \min\{\deg h(x) | h(x) \in \mathbb{F}_q[x],\ h(x) \in \mu^{-1}(f)\}$.

Osservazione. L'applicazione $\deg_{qt}$ non è in generale un'applicazione grado dato che non rispetta la regola del prodotto.

Proposizione 2.2. *Sia $f \in \mathcal{R}_g^q$. Esiste unico un polinomio $h(x) \in \mu^{-1}(f)$ tale che*

$$\deg h(x) = \deg_{qt}(f).$$

Osservazione. Grazie al teorema precedente possiamo identificare $f \in \mathcal{R}_g^q$ con il polinomio $h(x) \in \mathbb{F}_q[x]$ tale che $\mu(h(x)) = f$ e $\deg h(x) = \deg_{qt} f$.
Utilizzeremo quindi la notazione $f(x)$ sia per un elemento di $\mathcal{R}_g^q$ che per il suo rappresentante in $\mathbb{F}_q[x]$ di grado minimo.

Teorema 2.3 (Teorema Cinese del Resto). *Sia $\mathcal{R}$ un anello commutativo e*

$$\mathfrak{r}_1, \ldots, \mathfrak{r}_m \subset \mathcal{R}$$

ideali tali che $\mathfrak{r}_i + \mathfrak{r}_j = \mathcal{R}$ per ogni $i \neq j$.

$$\begin{array}{rcl} \phi : \mathcal{R} & \longrightarrow & \displaystyle\prod_{i=1}^{m} \mathcal{R}/\mathfrak{r}_i \\ a & \longmapsto & (a_1, \ldots, a_m) \end{array}$$

dove $a - a_i \in \mathfrak{r}_i$ per ogni i. Allora si ha che:

- $\ker \phi = \bigcap_{i=1}^{m} \mathfrak{r}_i$,

– *ϕ è un omomorfismo surgettivo.*

Quindi esiste un isomorfismo

$$\tilde{\phi} : \mathcal{R}/\bigcap_{i=1}^{m} \mathfrak{r}_i \longrightarrow \prod_{i=1}^{m} \mathcal{R}/\mathfrak{r}_i.$$

Dim. Si veda [La] pag. 65. □

Osservazione. Useremo questo teorema nel caso di anelli di polinomi $\mathcal{R} = \mathbb{F}_q[x]$. Quindi preso $g(x) \in \mathbb{F}_q[x]$ con $q = p^\delta$ e considerata

$$g(x) = \prod_{i=1}^{m} g_i(x)^{e_i}$$

la decomposizione di $g(x)$ in fattori irriducibili con $e_i \geq 1$, allora

$$\bigcap_{i=1}^{m} (g_i(x)^{e_i}) = \left(\prod_{i=1}^{m} g_i(x)^{e_i} \right) = (g(x))$$

siccome $\gcd(g_i(x), g_j(x)) = 1$. L'applicazione $\tilde{\phi}$ del teorema 2.3 è quindi un isomorfismo tra $\mathbb{F}_q[x]/(g(x))$ e $\prod_{i=1}^{m} \mathbb{F}_q[x]/(g_i(x)^{e_i})$.

Passiamo ora al problema fondamentale della sezione, cioè utilizzare il Teorema Cinese del Resto per ricondurci all'analisi degli anelli quozienti di potenze di irriducibili.

Sia $g(x)$ un polinomio di $\mathbb{F}_q[x]$ e

$$g(x) = \prod_{i=1}^{m} g_i(x)^{e_i}$$

la sua decomposizione in irriducibili.

Per il teorema 2.3, esiste un isomorfismo di anelli

$$\mathcal{R}_g^q \cong \prod_{i=1}^{m} \mathcal{R}_{g_i^{e_i}}^q. \tag{2.1}$$

Vediamo di esplicitare tale isomorfismo.

Proposizione 2.4. *Sia $g(x) \in \mathbb{F}_q[x]$ tale che $g(x) = \prod_{i=1}^m g_i(x)^{e_i}$ sia la sua decomposizione in irriducibili.*
Allora l'applicazione

$$\phi : \mathcal{R}_g^q \longrightarrow \prod_{i=1}^m \mathcal{R}_{g_i^{e_i}}^q$$

tale che $\phi(u(x)) = (u_1(x), \ldots, u_m(x))$ con

$$u_i(x) \equiv u(x) \pmod{g_i(x)^{e_i}}$$

è un isomorfismo con inversa

$$\phi^{-1}(u_1(x), \ldots, u_m(x)) = \sum_{i=1}^m u_i(x) v_i(x) \frac{g(x)}{g_i(x)^{e_i}} \pmod{g(x)} \tag{2.2}$$

dove $v_i(x)$ è l'elemento inverso di $g(x)/g_i(x)^{e_i}$ nel gruppo moltiplicativo $M_{g_i^{e_i}}^q$.

Dim. Dobbiamo solo verificare che la relazione (2.2) caratterizza l'applicazione inversa. Bisogna innanzitutto notare che $v_i(x)$ è ben definito. Siccome

$$\gcd\left(\frac{g(x)}{g_i(x)^{e_i}}, g_i(x)\right) = 1,$$

$g(x)/g_i(x)^{e_i}$ possiede un inverso in $\mathcal{R}_{g_i^{e_i}}^q$, quindi $v_i(x)$ è ben definito.

Per ipotesi

$$\begin{aligned} u_i(x) &\equiv u(x) \pmod{g_i(x)^{e_i}} \\ &\equiv \sum_{i=1}^m u_i(x) v_i(x) \frac{g(x)}{g_i(x)^{e_i}} \pmod{g_i(x)^{e_i}} \\ &\equiv u_i(x) v_i(x) \frac{g(x)}{g_i(x)^{e_i}} \pmod{g_i(x)^{e_i}} = u_i(x). \end{aligned}$$

Abbiamo così completato la dimostrazione. □

Grazie a questa proposizione si vede esplicitamente che per decomporre l'anello $\mathcal{R}_g^q$ possiamo separatamente studiare la struttura di ogni fattore $\mathcal{R}_{g_i^{e_i}}^q$ contenuto nel prodotto (2.1).

La prossima proposizione mette in relazione il gruppo moltiplicativo dell'anello $\mathcal{R}_g^q$ con il gruppo moltiplicativo degli anelli $\mathcal{R}_{g_i^{e_i}}^q$.

Proposizione 2.5. *Sia $g(x) \in \mathbb{F}_q[x]$ tale che $g(x) = \prod_{i=1}^m g_i(x)^{e_i}$ sia la sua decomposizione in irriducibili. Allora*

$$M_g^q \cong \prod_{i=1}^m M_{g_i^{e_i}}^q.$$

Dim. La dimostrazione segue direttamente dalla proposizione 2.4. È sufficiente infatti verificare che, dato l'isomorfismo ϕ come da proposizione, allora ad ogni $u(x) \in M_g^q$ corrisponde una m-upla $(u_1, \ldots, u_m) \in \prod_{i=1}^m M_{g_i^{e_i}}^q$ e viceversa.

La prima implicazione è facile. Dato che $u(x)$ è invertibile in M_g^q, allora

$$\gcd(u(x), g(x)) = 1.$$

Questo vuol dire che

$$\gcd(u(x), g_i(x)) = 1 \qquad \text{per} \qquad i = 1, \ldots, m.$$

Essendo tutti i $g_i(x)$ irriducibili e considerato che

$$u_i(x) \equiv u(x) \pmod{g_i(x)^{e_i}},$$

allora esiste $h_i(x) \in \mathbb{F}_q[x]$ tale che $u(x) = u_i(x) + h_i(x)g_i(x)^{e_i}$. Quindi

$$\gcd(u_i(x), g_i(x)) = \gcd(u(x) - h_i(x)g_i(x)^{e_i}, g_i(x)) = \gcd(u(x), g_i(x)) = 1.$$

Viceversa, se per ogni $i = 1, \ldots, m$ vale $\gcd(u_i(x), g_i(x)) = 1$, allora

$$\gcd\left(u_i(x)\frac{g(x)}{g_i(x)^{e_i}}v_i, g_i\right) = 1$$

essendo, per come definiti, $g(x)/g_i(x)^{e_i}$ e v_i invertibili in $\mathcal{R}_{g_i^{e_i}}^q$.
Per la proposizione 2.4

$$u(x) \equiv \sum_{i=1}^m u_i(x)v_i(x)\frac{g(x)}{g_i(x)^{e_i}} \pmod{g(x)}.$$

Se, per assurdo, $\gcd(u(x), g(x)) \neq 1$ allora esisterebbe $0 \leq j \leq m$ tale che

$$\gcd(u(x), g_j(x)) = g_j(x)$$

e quindi

$$u(x) = g_j(x)u'(x).$$

Questo vorrebbe dire che

$$g_j(x) \Big| u_j(x)v_j(x)\frac{g(x)}{g_j(x)^{e_j}}$$

che è assurdo. □

Ci siamo così ridotti al caso di anelli quozienti del tipo $\mathcal{R}_{g^t}^q$ con $g(x)$ irriducibile e $t \in \mathbb{N}_+$. Nel caso in cui t è uguale a 1, $\mathcal{R}_g^q$ è un campo finito e il suo gruppo moltiplicativo, M_g^q, per **??**, è un gruppo ciclico.

Per il Teorema dell'Elemento Primitivo 1.4 esiste allora un generatore α del gruppo moltiplicativo M_g^q e dal punto di vista puramente insiemistico otteniamo

$$\mathcal{R}_g^q = \{0\} \cup M_g^q.$$

Passiamo a considerare il caso in cui t è un intero maggiore o uguale a 2. Il problema è più delicato e richiede innanzitutto dei teoremi preliminari che trattiamo subito.

Proposizione 2.6. *Sia* $g(x) \in \mathbb{F}_q[x]$. *Fissato* $f(x) \in \mathbb{F}_q[x]$ *esistono unici dei polinomi* $f_i(x) \in \mathbb{F}_q[x]$, $i =, \ldots, m$, *con* $\deg f_i(x) < \deg g(x)$ *tali che*

$$f(x) = \sum_{k=0}^{m} f_i(x)g(x)^i.$$

Dim. Affrontiamo la dimostrazione per induzione. $\mathbb{F}_q[x]$ è un anello euclideo, possiamo quindi trovare unici due polinomi $f^{(1)}(x)$ e $f_0(x)$ con relazione tra gradi $\deg f_0(x) < \deg g(x)$, tali che

$$f(x) = f^{(1)}(x)g(x) + f_0(x).$$

Supponiamo quindi vero che per ogni $j \leq i$ valga la relazione

$$f(x) = f^{(j)}(x)g(x)^j + \sum_{k=0}^{j-1} f_k(x)g(x)^k \tag{2.3}$$

con $\deg f_k(x) < \deg g(x)$ per ogni $k \leq j-1$. Questa osservazione costituisce la base dell'induzione. Dimostriamo che possiamo trovare due polinomi $f^{(i+1)}(x)$ e $f_i(x)$ con $\deg f_i(x) < \deg g(x)$ tali che

$$f(x) = f^{(i+1)}(x)g(x)^{i+1} + \sum_{k=0}^{i} f_k(x)g(x)^k.$$

Per dimostrarlo basta applicare la divisione euclidea a $f^{(i)}$ e inserirla nella relazione (2.3).

Verifichiamo che i polinomi $f^{(i)}(x)$ sono un numero finito. Questa è una semplice conseguenza dei gradi di questi ultimi che costituiscono una successione strettamente decrescente in $\mathbb{N}$ e quindi definitivamente nulla.

L'unicità di una tale scrittura segue direttamente dalla condizione imposta sui gradi dei polinomi $f_i(x)$. □

Osservazione. Usando l'applicazione grado quoziente, la condizione sui gradi dei polinomi $f_i(x)$ nella precedente proposizione ci permette di esprimere, in modo unico, ogni polinomio $f(x) \in \mathcal{R}_{g^t}^q$ in termini di classi di resto, così da ottenere:

$$f(x) = \sum_{i=0}^{t-1} f_i(x)g(x)^i$$

dove, per semplicità, indichiamo con $f_i(x) \in \mathcal{R}_{g^t}^q$, $i = 0, \ldots, t-1$, i polinomi tali che $\deg_{qt}(f_i) < \deg(g)$.

Passiamo ora ad un teorema utile per quelli che affronteremo in seguito in questa sottosezione.

Teorema 2.7. *Sia $g(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile e $t \geq 2$. Sia inoltre*

$f(x) \in \mathcal{R}_{g^t}^q$, *con*

$$f(x) = \sum_{i=0}^{t-1} f_i(x) g(x)^i,$$

dove $f_i(x) \in \mathcal{R}_{g^t}^q$ *con* $i = 0, \ldots, t-1$.
Allora $f(x) \in M_{g^t}^q$ *se e soltanto se* $f_0(x) \neq 0$.

Dim. Se $f(x) \in M_{g^t}^q$ allora $f(x)$ è invertibile in $\mathcal{R}_{g^t}^q$. Questo vuol dire che esiste $h(x) \in \mathcal{R}_{g^t}^q$ tale che $f(x)h(x) = 1$.

Riscrivendo la formula tramite un'equivalenza otteniamo:

$$f(x)h(x) \equiv 1 \pmod{g(x)^t},$$

esiste quindi un $s(x) \in \mathbb{F}_q[x]$ tale che, in $\mathbb{F}_q[x]$,

$$f(x)h(x) + s(x)g(x)^t = 1.$$

Per l'identità di Bèzout otteniamo che $\gcd(f(x), g(x)^t) = 1$, ma essendo $g(x)$ un polinomio irriducibile, questo equivale a dire che $\gcd(f(x), g(x)) = 1$ e quindi che

$$g(x) \nmid f(x). \tag{2.4}$$

Per la dimostrazione precedente, data

$$f(x) = f_0(x) + f_1(x)g(x) + \cdots + f_{t-1}(x)g(x)^{t-1} \in \mathcal{R}_{g^t}^q$$

la rappresentazione di $f(x)$ in base al polinomio $g(x)$, la relazione (2.4) è equivalente a

$$f_0(x) \neq 0.$$

Per dimostrare l'altra implicazione, basta ripercorrere i passaggi in senso inverso. □

Grazie a questo teorema possiamo affrontare il successivo. Scopo di questo teorema è mostrare una possibile, e per noi utile, decomposizione di $\mathcal{R}_{g^t}^q$ come unione di insiemi disgiunti.

Teorema 2.8. *Sia* $g(x) \in \mathbb{F}_q[x]$ *un polinomio irriducibile e* $t \geq 2$*. Allora*

$$\mathcal{R}^q_{g^t} = \{0\} \cup \bigsqcup_{i=0}^{t-1} g(x)^i \cdot M^q_{g^{t-i}}.$$

Dim. Fissato $f(x)$ in $\mathcal{R}^q_{g^t}$ esiste $k \in \mathbb{N}$ tale che

$$k = \max\left\{ i \in \{0, \ldots, t-1\} \;\middle|\; g(x)^i \mid f(x) \right\}.$$

Quindi possiamo rappresentare un elemento $f(x) \in \mathcal{R}^q_{g^t}$ nella forma

$$f(x) = g(x)^k h(x), \tag{2.5}$$

dove $\gcd(h(x), g(x)) = 1$.

Ritornando alla scomposizione $f(x) = \sum_{i=0}^{t-1} f_i(x) g(x)^i$, la relazione (2.5) è equivalente a:

$$f_i(x) = 0, \ \forall\, i < k \quad \text{e} \quad f_k(x) \neq 0, \quad \text{con } k \leq t-1.$$

Procedendo quindi passo passo, grazie al teorema 2.4, otteniamo la seguente scomposizione:

- se $f_0(x) \neq 0 \ \Rightarrow \ f(x) \in M^q_{g^t}$
- se $f_0(x) = 0 \; e \; f_1(x) \neq 0 \ \Rightarrow \ f(x) \in g(x) \cdot M^q_{g^{t-1}}$
- per $i \leq t-1$, se $f_j(x) = 0 \; \forall j < i \; e \; f_i(x) \neq 0 \ \Rightarrow \ f(x) \in g^i(x) \cdot M^q_{g^{t-i}}$
- se $f_i(x) = 0, \ \forall i \leq t-1 \ \Rightarrow \ f(x) = 0.$

Dato che l'altra inclusione è banale, la tesi è dimostrata. □

2.2 Ricerca dei Generatori di $M_{g^s}^q$

Nella sezione precedente abbiamo ottenuto una decomposizione dell'anello $\mathcal{R}_{g^t}^q$ come unione disgiunta di sottoinsiemi del tipo

$$g(x)^i \cdot M_{g^{t-i}}^q.$$

Il nostro scopo ora è trovare la scomposizione dei gruppi moltiplicativi $M_{g^t}^q$ come prodotto di gruppi ciclici, in accordo con il Teorema di Classificazione dei Gruppi Abeliani Finiti 1.2.

Teorema 2.9. *Sia $g(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile di grado r e $t \geq 2$, allora l'ordine del gruppo $M_{g^t}^q$ è $q^{(t-1)r}(q^r - 1)$.*
Inoltre $M_{g^t}^q = M_g^q \times S_p$ dove S_p è il p-sottogruppo di Sylow di $M_{g^t}^q$.

Dim. Verifichiamo prima la tesi sull'ordine. Per il teorema 2.7

$$\begin{aligned} M_{g^t}^q &= \left\{ f(x) \in \mathcal{R}_{g^t}^q \mid f_0(x) \neq 0 \right\} = \\ &= \left(\mathcal{R}_{g^t}^q \right) \setminus \left\{ f(x) \in \mathcal{R}_{g^t}^q \;\middle|\; g(x) \mid f(x) \right\} \end{aligned}$$

quindi

$$\begin{aligned} \#M_{g^t}^q &= \#\left(\left(\mathcal{R}_{g^t}^q \right) \setminus \left\{ f(x) \in \mathcal{R}_{g^t}^q \;\middle|\; g(x) \mid f(x) \right\} \right) \\ &= \#\left(\mathcal{R}_{g^t}^q \right) - \#\left\{ f(x) \in \mathcal{R}_{g^t}^q \;\middle|\; g(x) \mid f(x) \right\} \\ &= q^{tr} - q^{(t-1)r} = q^{(t-1)r}(q^r - 1) \end{aligned}$$

dato che $\left\{ f(x) \in \mathcal{R}_{g^t}^q \;\middle|\; g(x) \mid f(x) \right\} = g(x) \cdot \mathcal{R}_{g^{t-1}}^q$.
Sia $f \in M_{g^t}^q$. Per il teorema 2.7 possiamo scrivere

$$f(x) = \sum_{i=0}^{t-1} f_i(x) g^i(x) \qquad \text{con} \qquad f_0(x) \neq 0.$$

Siccome $f_0(x) \neq 0$ e $\deg_{qt} f_0(x) < r$, allora $f_0(x)$ è un elemento invertibile in $\mathcal{R}_{g^t}^q$, ne segue che

$$\exists\, f_0'(x) \in \mathcal{R}_{g^t}^q \qquad \text{tale che} \qquad f_0(x) f_0'(x) = 1$$

quindi

$$f(x) = f_0(x)(1 + \sum_{i=1}^{t-1} f_i(x) f_0'(x) g(x)^i) = f_0(x)(1 + \sum_{i=1}^{t-1} f_i'(x) g(x)^i),$$

dove nell'ultima uguaglianza abbiamo applicato il teorema 2.6.

Per le osservazioni fatte in precedenza, al variare di $f(x) \in \mathcal{R}_{g^t}^q$, gli elementi $f_i'(x)$ sono caratterizzati dal fatto che hanno grado minore di r, quindi gli $f_i'(x)$ possono esser considerati come elementi di M_g^q.

Dimostrando il prossimo teorema concluderemo anche questa dimostrazione. □

Teorema 2.10. *Sia $f(x) \in M_{g^t}^q$ con $g(x)$ irriducibile e $t \geq 2$. Allora l'ordine moltiplicativo di $f(x)$ è p^k, con $k \in \mathbb{N}_+$ se e soltanto se esiste $m(x) \in \mathbb{F}_q[x]$ tale che*

$$f(x) = 1 + m(x)g(x).$$

Dim. Per quanto dimostrato all'inizio della di questa sottosezione, esitono $f_0(x)$ e $f^{(1)}(x)$, come da dimostrazione del teorema 2.6, due polinomi di $\mathbb{F}_q[x]$ con $\deg f_0(x) < \deg g(x)$ tali che $f(x) = f_0(x) + f^{(1)}(x)g(x)$.
Allora, preso $k \in \mathbb{N}_+$ tale che $p^k > t$,

$$f(x)^{p^k} = \left(f_0(x) + f^{(1)}(x)g(x)\right)^{p^k} = f_0(x)^{p^k} + \left(f^{(1)}g(x)\right)^{p^k}.$$

In $\mathcal{R}_{g^t}^q$ la relazione diventa

$$f(x)^{p^k} = f_0(x)^{p^k},$$

e, sapendo che il grado di $f_0(x)$ è minore del grado di $g(x)$, otteniamo, per il criterio di uguaglianza tra polinomi,

$$f_0(x) = 1.$$

Viceversa, se esiste $m(x) \in \mathbb{F}_q[x]$ tale che $f(x) = 1 + m(x)g(x)$, allora per $k \in \mathbb{N}_+$ tale che $p^k > t$

$$f(x)^{p^k} = (1 + m(x)g(x))^{p^k} = 1 + (m(x)g(x))^{p^k} = 1$$

in $\mathcal{R}_{g^t}^q$. L'ordine non potrà che essere un divisore di p^k, quindi sarà sempre una potenza di p. □

Quello che ci manca ora per completare la ricerca dei generatori di $M_{g^t}^q$ è caratterizzare i generatori del p-Sylow, che è un gruppo abeliano finito quindi esprimibile come prodotto di gruppi ciclici.

Osservazione. Da questo punto in avanti, indichiamo con $\alpha \in \mathbb{F}_q$ un elemento algebrico di $\mathbb{F}_p$ di grado δ. Quindi otteniamo che $\mathbb{F}_q = \mathbb{F}_p[\alpha]$. Questo vuol dire che ogni elemento di $\mathbb{F}_q$ è esprimibile come combinazione lineare di $1, \alpha, \ldots, \alpha^{\delta-1}$ a coefficienti in $\mathbb{F}_p$.

Teorema 2.11. *Sia $f(x) \in S_p$, e $f(x) = 1 + f_1(x)g(x) + \cdots + f_{t-1}g^{t-1}$ la sua scrittura in base $g(x)$ come da proposizione 2.6.*
Allora esistono $c_{(ij)}^{(k)} \in \mathbb{N}$ tali che

$$f(x) = \prod_{i,j,k} a_{i,j,k}(x)^{c_{(ij)}^{(k)}},$$

dove $a_{i,j,k}(x) := 1 + \alpha^i x^j g(x)^k \in S_p$, e:

- $0 \leq i < \delta$;
- $0 \leq j < r$;
- $1 \leq k \leq t$ *con* $p \nmid k$.

La dimostrazione di questo teorema è conseguenza del successivo lemma.

Lemma 2.12. *Sia $f(x) = 1 + f_k(x)g(x)^h + m(x)g(x)^{h+1} \in \mathbb{F}_q[x]$ con $h \geq 1$ e $\deg f_h(x) < \deg g(x)$. Allora esistono $c_{(ij)} \in \{0, \ldots, p-1\}$ tali che*

$$\prod_{i,j}(1 + \alpha^i x^j g(x)^k)^{c_{(ij)}} \equiv f(x) \pmod{g(x)^{h+1}},$$

con $0 \leq i < \delta$, $0 \leq j < r-1$ e $0 \leq k \leq h$.

Dim. Dividiamo la dimostrazione in due parti: la prima in cui $p \nmid h$ e la seconda in cui $p \mid h$.

Affrontiamo quindi il caso in cui $p \nmid h$. Per come definito

$$f_h(x) = \sum_{i,j} c_{(ij)} \alpha^i x^j$$

con $c_{(ij)} \in \mathbb{F}_p$. Quindi

$$\begin{aligned} \prod_{i,j} (1 + \alpha^i x^j g(x)^h)^{c_{(ij)}} &= 1 + \sum_{i,j} c_{(ij)} \alpha^i x^j g(x)^h + m'(x) g(x)^{h+1} \equiv \\ &\equiv f(x) \pmod{g(x)^{h+1}}. \end{aligned}$$

Ora analizziamo il caso in cui invece $p \mid h$, quindi $h = ph'$. Per ipotesi $f(x) = 1 + f_h(x) g(x)^{ph'} + m(x) g(x)^{h+1}$.
Siccome $g(x)$ è un polinomio irriducibile di $\mathbb{F}_q[x]$, il campo da lui generato è perfetto. Sia quindi $\bar{f}_h(x) \in \mathcal{R}_g^q$ la proiezione naturale di $f_h(x)$ in $\mathcal{R}_g^q$. Il campo perfetto ci dice che

$$\exists\, \bar{l}(x) \in \mathcal{R}_g^q \quad \text{tale che} \quad \bar{l}(x)^p = \bar{f}_h(x),$$

ovvero che esiste $l(x) \in \mathbb{F}_q[x]$ con $\deg l(x) < \deg g(x)$ tale che

$$l(x)^p \equiv f_h(x) \pmod{g(x)}.$$

Scriviamo quindi

$$l(x) = \sum_{i,j} c_{i,j} \alpha^j x^i \in \mathbb{F}_q[x].$$

Possiamo quindi concludere verificando che

$$\begin{aligned} \left(\prod_{i,j} (1 + \alpha^i x^j g(x)^{h'})^{c_{(ij)}} \right)^p &= \left(1 + l(x) g(x)^{h'} + \tilde{l}(x) g(x)^{h'+1} \right)^p = \\ &= 1 + l(x)^p g(x)^{ph'} + m'(x) g(x)^{p(h'+1)} \equiv \\ &\equiv 1 + f_h(x) g(x)^h \pmod{g(x)^{h+1}} \equiv \\ &\equiv f(x) \pmod{g(x)^{h+1}} \end{aligned}$$

□

Osservazione. Grazie alla seconda parte della dimostrazione, possiamo evitare di prendere in considerazione nell'insiemi di generatori di S_p gli elementi del tipo

$$1 + \alpha^i x^j g(x)^k$$

con k multiplo di p.

Dim. [del teorema 2.11]
Per ipotesi il polinomio $f_1(x)$ è della forma

$$f_1(x) = \sum_{i,j} c_{(ij)}^{(1)} \alpha^i x^j,$$

quindi per il lemma precedente

$$\prod_{i,j} (1 + \alpha^i x^j g(x))^{c_{(ij)}^{(1)}} = 1 + f_1(x)g(x) + m(x)g(x)^2. \tag{2.6}$$

Consideriamo $\tilde{f}_2(x) \equiv f_2(x) - m(x) \pmod{g(x)}$, allora

$$\tilde{f}_2(x) = \sum_{i,j} c_{i,j}^{(2)} \alpha^i x^j.$$

Sempre per il lemma precedente otteniamo che

$$\prod_{i,j} (1 + \alpha^i x^j g(x)^2)^{c_{(ij)}^{(2)}} = 1 + \tilde{f}_2(x)g(x)^2 + \tilde{m}(x)g(x)^2. \tag{2.7}$$

Moltiplicando i polinomi ottenuti tramite i prodotti in (2.6) e (2.7), otteniamo

$$\begin{aligned}
&(1 + f_1(x)g(x) + m(x)g(x)^2)(1 + \tilde{f}_2(x)g(x)^2 + \tilde{m}(x)g(x)^2) = \\
&= 1 + f_1(x)g(x) + m(x)g(x)^2 + \tilde{f}_2(x)g(x)^2 + \hat{m}(x)g(x)^3 = \\
&= 1 + f_1(x)g(x) + f_2(x)g(x)^2 + \hat{m}(x)g(x)^3.
\end{aligned}$$

Continuando iterativamente, fino al t-esimo passo, otteniamo la tesi. □

A questo punto consideriamo l'omomorfismo di gruppi

$$\eta : S_p \longrightarrow \prod_{i,j,k} (a_{i,j,k}(x)) \tag{2.8}$$

dove i, j e k soddisfano le stesse condizioni delle ipotesi del teorema 2.11. L'insieme

$$A = \{a_{i,j,k} \in \mathcal{R}_{g^t}^q\}$$

è un insieme di generatori per S_p, allora dimostrando l'iniettività dell'omomorfismo (2.8) otterremo un isomorfismo tra i due gruppi.

Rispettando l'omomorfismo la struttura di gruppo, basterà fare la seguente considerazione per verificare l'iniettività.

Teorema 2.13. *Siano $g(x)$, $a_{i,j,k}$, e t come da teorema 2.11. Allora*

$$\prod_{i,j,k} a_{i,j,k}^{c_{(ijk)}} \equiv 1 \pmod{g(x)^t} \iff c_{(ijk)} \equiv 0 \pmod{\mathrm{ord}(a_{ijk})}.$$

Dim. Sviluppiamo la potenza dei singoli $a_{i,j,k} \in S_p$.
Mettiamo in evidenza la massima potenza di p che divide $c_{(ijk)}$, quindi

$$c_{(ijk)} = p^{s_{(ijk)}} c'_{(ijk)} \text{ con } p \nmid c'_{(ijk)},$$

allora,

$$\begin{aligned}
(a_{i,j,k})^{c_{(ijk)}} &= \left(1 + \alpha^i x^j g(x)^k\right)^{c_{(ijk)}} = \\
&= \left(1 + \alpha^{ip^{s_{(ijk)}}} x^{jp^{s_{(ijk)}}} g(x)^{kp^{s_{(ijk)}}}\right)^{c'_{(ijk)}} = \\
&= 1 + \sum_{h=0}^{c'_{(ijk)}} \binom{c'_{(ijk)}}{h} \left(\alpha^{ip^{s_{(ijk)}}} x^{jp^{s_{(ijk)}}} g(x)^{kp^{s_{(ijk)}}}\right)^h.
\end{aligned}$$

Sia $\boldsymbol{kp^{s_{(ijk)}}} = \min\{kp^{s_{(ijk)}}\}$, otteniamo che $\boldsymbol{kp^{s_{(ijk)}}} \geq t$.
Considerando tutti gli elementi del prodotto otteniamo che

$$(a_{i,j,k})^{c_{(ijk)}} \equiv 1 \pmod{g(x)^t}$$

quindi

$$c_{(ijk)} \equiv 0 \pmod{\mathrm{ord}(a_{i,j,k})}$$

per definizione di ordine. □

Capitolo 3

L'anello $\mathcal{R}_g^q$ e il Codice Duale

In questo capitolo studiamo le strutture di orbite nell'anello $\mathcal{R}_g^q$.

Iniziamo col dare uno sguardo alla teoria delle successioni a ricorrenza lineare. Ne diamo definizione e proprietà fondamentali.

Colleghiamo quindi la teoria sulle successioni a ricorrenza lineare con lo sviluppo in serie formali della divisione tra i polinomi $u(x) \in \mathcal{R}_g^q$ e $g(x)$. Infatti lo sviluppo in serie di potenze formali genera una successione a ricorrenza lineare con ricorrenza dipendente dal polinomio $g(x)$. Grazie alla proprietà dei codici ciclici accorciati per la quale le parole del codice duale possono essere costruite tramite la stessa ricorrenza lineare, quello che otteniamo è una corrispondenze tra gli elementi $u(x) \in \mathcal{R}_g^q$ e le parole del codice duale.

Arriviamo a lavorare sulla scomposizione in orbite dell'anello $\mathcal{R}_g^q$. Le orbite sono molto importanti nel nostro lavoro in quanto scompongono, grazie alla corrispondenza, data dalle successioni, il codice duale in classi per le quali il calcolo del peso delle parole in esse contenute è notevolmente semplificato.

Il nostro lavoro è quindi centrato nella ricerca di tali orbite in $\mathcal{R}_g^q$. A questo punto dello scritto facciamo alcune considerazioni sull'azione di un gruppo ciclico su un prodotto cartesiano. Vediamo quindi che la nostra ricerca si riconduce, grazie all'azione di gruppo e al Teorema Cinese del Resto, alla ricerca dei rappresentanti delle orbite in $\mathcal{R}_{g^t}^q$ con $g(x)$ irriducibile.

Concludiamo quindi con uno studio approfondito delle orbite di $\mathcal{R}_{g^t}^q$ con $g(x)$ irriducibile, dove forniamo un elemento per ogni orbita in esso contenuta.

3.1 Successioni a Ricorrenza Lineare

Diamo le nozioni sulle successioni a ricorrenza lineare che ci serviranno in questo capitolo. Esse saranno fondamentali in seguito per costruire una corrispondenza biunivoca tra il codice duale di codici ciclici accorciati e l'anello $\mathcal{R}_g^q$.

Definizione 3.1. *La successione* $\mathfrak{s} = (s_i)_{i\in\mathbb{N}} \subset \mathbb{F}_q$ *si dice* a ricorrenza lineare di ordine k *se esistono* k *un intero positivo e* $a, a_0, \ldots, a_k$ *elementi di un campo finito* $\mathbb{F}_q$ *tali che*

$$s_{n+k} = a_{k-1}s_{n+k-1} + a_{k-2}s_{n+k_2} + \cdots + a_0 s_n + a \tag{3.1}$$

per ogni $n \in \mathbb{N}$. *Diremo che la successione è* omogenea *se* $a = 0$.

Osservazione. I valori iniziali $s_0, \ldots, s_{k-1}$ determinano univocamente la successione $\mathfrak{s}$.

Definizione 3.2. *Sia* S *un insieme non vuoto, e sia* $\mathfrak{s} = (s_i)_{i\in\mathbb{N}} \subset S$ *una sua successione. Se esistono* $r > 0$ *e* $n' \geq 0$ *tale che* $s_{n+r} = s_n$ *per ogni* $n \geq n'$ *allora la successione è detta* definitivamente periodica *e* r *è detto* periodo *della successione. Il più piccolo tra tutti i possibili periodi di una successione definitivamente periodica è detto* periodo minimo.

Lemma 3.3. *Ogni periodo di una successione definitivamente periodica è divisibile per il periodo minimo.*

Definizione 3.4. *Una successione si dice* periodica *se esiste* $r > 0$ *tale che per ogni* $n \geq 0$ *vale* $s_{n+r} = s_n$.

Teorema 3.5. *Sia* $\mathbb{F}_q$ *un campo finito e* k *un intero positivo. Ogni successione a ricorrenza lineare di ordine* k *in* $\mathbb{F}_q$ *è definitivamente periodica con periodo minimo* r *tale che* $r \leq q^k$, *e* $r \leq q^k - 1$ *se è omogenea. Inoltre se il coefficiente* a_0 *della relazione* (3.1) *è diverso da 0 allora la successione è periodica.*

Dim. Si veda [LN] pagg. 399-400. □

Definizione 3.6. *Se $\mathfrak{s}$ è una successione a ricorrenza lineare di ordine k che soddisfa la relazione* (3.1)*, allora il polinomio*

$$f(x) = x^k - a_{k-1}x^{k-1} - a_{k-2}x^{k-2} - \cdots - a_0 \in \mathbb{F}_q[x]$$

è detto polinomio caratteristico *della successione.*

Teorema 3.7. *Sia $\mathfrak{s}$ una successione a ricorrenza lineare omogenea di $\mathbb{F}_q$ con valori iniziali non nulli. Supponendo che il polinomio caratteristico $f(x) \in \mathbb{F}_q[x]$ sia irriducibile e soddisfi $f(0) \neq 0$, allora la successione è periodica di periodo minimo uguale a $\mathrm{ord}(f(x))$. In generale se $f(x)$ non è irriducibile il periodo della successione divide $\mathrm{ord}(f(x))$.*

Dim. Si veda [LN] pagg. 408-409. □

Come le successioni a ricorrenza lineare, anche la parole del codice duale di un codice ciclico accorciato soddisfano una ricorrenza. Di seguito vediamo un metodo per il calcolo delle successioni a ricorrenza lineari i quali coefficienti, considerati a gruppi di n consecutivi, formano le parole del codice duale.

3.2 $\mathcal{R}_g^q$ e Il Duale di un Codice CRC

Abbiamo già fatto vedere nel primo capitolo come caratterizzare le parole del codice duale a partire dal polinomio generatore del codice ciclico accorciato. Abbiamo mostrato due metodi. Il primo in cui le caratterizzavamo tramite quelle parole che annullano una matrice generatrice del codice ciclico accorciato, e il secondo, equivalente, mostrando che le componenti consecutive di una parola del codice duale soddisfano una ricorrenza lineare basata sul polinomio generatore $g(x)$, come da lemma 1.46.

Teorema 3.8. *Sia $g(x) = \sum_{j=0}^{r} g_j x^j$ un polinomio monico di $F_q[x]$ tale che*

$$\gcd(x, g(x)) = 1$$

e sia $u(x) \in F_q[x]$, tale che $\deg u(x) < \deg g(x)$. Allora esiste unica una

successione $(c_i)_{i\in\mathbb{N}} \subset \mathbb{F}_q$ *tale che*

$$\frac{u(x)}{g(x)} = \sum_{i=0}^{\infty} \frac{c_i}{x^{i+1}} = c(1/x).$$

Inoltre la successione $(c_i)_{i\in\mathbb{N}}$ *soddisfa la relazione*

$$c_i = -g_0 c_{i-r} - \cdots - g_{r-1} c_{i-1} \qquad \forall i \geq r. \tag{3.2}$$

Dim. Siano $g(x) = \sum_{j=0}^{r} g_j x^j \in \mathbb{F}_q[x]$ e $u(x) = \sum_{j=0}^{r-1} u_j x^j \in \mathbb{F}_q[x]$. Per le ipotesi su $g(x)$, $g_r = 1$ e $g_0 \neq 0$. Dimostreremo l'esistenza e l'unicità costruendo i primi r termini della successione tramite i polinomi $g(x)$ e $u(x)$ e il resto della successione basandoci sulla relazione (3.2).

Otterremo i primi r termini risolvendo il seguente sistema lineare

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{r-3} \\ c_{r-2} \\ c_{r-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & & \dots & 0 \\ g_{r-1} & 1 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & & \\ g_3 & \dots & g_{r-1} & 1 & \ddots & \vdots \\ g_2 & g_3 & \dots & g_{r-1} & 1 & 0 \\ g_1 & g_2 & g_3 & \dots & g_{r-1} & 1 \end{pmatrix}^{-1} \begin{pmatrix} u_{r-1} \\ u_{r-2} \\ \vdots \\ u_2 \\ u_1 \\ u_0 \end{pmatrix} \tag{3.3}$$

L'invertibilità della matrice è garantita dal fatto che è triangolare con diagonale costituita da tutti 1. Costruiremo infine i successivi elementi applicando la relazione (3.2).

Quello che faremo ora è verificare che la successione scelta soddisfa la condizione

$$\frac{u(x)}{g(x)} = \sum_{i=0}^{\infty} \frac{c_i}{x^{i+1}}.$$

$$\begin{aligned} c(1/x)g(x) &= \Big(\sum_{i=0}^{\infty} \frac{c_i}{x^{i+1}}\Big)\Big(\sum_{k=0}^{r} g_k x^k\Big) = \\ &= \sum_{i=1}^{\infty} \frac{1}{x^i}\Big(\sum_{k=0}^{r} c_{k+i-1} g_k\Big) + \sum_{i=0}^{r-1} x^i \Big(\sum_{k=i+1}^{r} c_{k-i+1} g_k\Big) \end{aligned}$$

dove $g_r = 1$.

Grazie alla relazione (3.2) segue che

$$\sum_{k=0}^{r} c_{k+i-1} g_k = 0 \qquad \forall i \in \mathbb{N}^*$$

e quindi la prima parte della somma si annulla, mentre l'altra sarà $u(x)$ grazie al sistema (3.3).

L'esistenza è quindi dimostrata, rimane solo da verificare l'unicità. Essa si ottiene semplicemente notando che la soluzione del sistema (3.3) è unica, inducendo l'unicità della nostra successione. □

Abbiamo dimostrato con il precedente teorema l'esistenza di una successione in $\mathbb{F}_q$ che soddisfa la condizione (3.2). Per quanto dimostrato quindi nel primo capitolo, presi n elementi consecutivi della successione otteniamo una parola del codice duale.

Il procedimento di estrazione delle parole del codice è molto semplice. Una volta calcolata la prima parola del codice, costituita dai primi n coefficienti della successione $c(1/x)$, costruiamo la parola successiva semplicemento perdendo la componente di testa della parola del codice (c_0) e acquisendo come ultima componente l'$(n+i)$-esima estratta dalla successione, cioè c_n. Tutte le parole successive quindi saranno formate dalla loro parola precedente perdento la componente di testa e acquisendo, come ultima componente, il primo coefficiente della successione non ancora utilizzato.

Non ci rimane quindi che vedere come costruire la successione in modo iterativo, così da poter generare le parole del codice duale. La costruzione seguirà il seguente procedimento: siano $g(x), u(x) \in \mathbb{F}_q$, due polinomi che soddisfano le ipotesi del teorema 3.8, allora:

$$\begin{aligned}\frac{u(x)}{g(x)} &= \frac{u_{r-1}x^{r-1}+\cdots+u_1x+u_0}{x^r+g_{r-1}x^{r-1}+\cdots+g_1x+g_0} = \\ &= \frac{1}{x}\frac{u_{r-1}x^r+\cdots+u_1x^2+u_0x}{x^r+g_{r-1}x^{r-1}+\cdots+g_1x+g_0} = \\ &= \frac{1}{x}\frac{u_{r-1}g(x)+xu(x)-u_{r-1}g(x)}{g(x)} = \\ &= \frac{u_{r-1}}{x}+\frac{xu(x) \pmod{g(x)}}{xg(x)}, \end{aligned} \tag{3.4}$$

dove il polinomio $xu(x) \pmod{g(x)}$ soddisfa le stesse ipotesi di $u(x)$. Procedendo con le potenze successive di x ed estraendo da ogni $x^iu(x) \pmod{g(x)}$, con $i = 0, 1, \ldots$, il coefficiente $(r-1)$-esimo, costruiamo una succesione $(c_i)_{i\in\mathbb{N}}$. Dobbiamo dimostrare che quest'ultima soddisfa le ipotesi del teorema 3.8, e quindi che è una successione a ricorrenza lineare.

Lemma 3.9. *Sia $(c_i)_{i\in\mathbb{N}}$ la successione definita tramite la formula* (3.4)*. Definita*

$$c(1/x) = \sum_{i=0}^{\infty} \frac{c_i}{x^{i+1}}$$

la serie di potenze formale ad essa associata allora la successione soddisfa la seguente relazione

$$c_i = -g_0c_{i-r} - \cdots - g_{r-1}c_{i-1} \qquad \forall i \geq r.$$

Dim. Per come è stata definita la successione per ogni $\eta \in \mathbb{N}$ vale

$$\frac{u(x)}{g(x)} = \sum_{i=0}^{\eta} \frac{c_i}{x^{i+1}} + \frac{x^{\eta+1}u(x) \pmod{g(x)}}{x^{\eta+1}g(x)}$$

Quindi

$$u(x) = \sum_{i=0}^{\eta} \frac{c_i}{x^{i+1}}g(x) + \frac{x^{\eta+1}u(x) \pmod{g(x)}}{x^{\eta+1}}, \tag{3.5}$$

fissato quindi $\nu \in \mathbb{N}$ maggiore di r, e preso un η abbastanza grande, allora il

coefficiente corrispondente a $x^{-\nu}$ della somma di sinistra di (3.5) sarà

$$g_0 c_\nu + \cdots + g_{r-1} c_{\nu+r-1} + c_{\nu+r}.$$

Per l'uguaglianza tra polinomi la relazione (3.5) deve essere nulla quindi

$$c_{\nu+r} = -g_0 c_\nu - \cdots - g_{r-1} c_{\nu+r-1}$$

a data l'arbitrarietà di η e ν il teorema è dimostrato. □

Siccome il duale è anch'esso uno spazio vettoriale su $\mathbb{F}_q$, allora il numero dei suoi elementi è q^r.

Vogliamo ora dimostrare di riuscire a catalogare tutte le parole del codice duale grazie al metodo descritto prima.

Osservazione. D'ora in poi indichiamo con $g(x) \in \mathbb{F}_q[x]$ il polinomio monico di grado r generatore del codice ciclico da cui otteniamo il codice ciclico accorciato. Con $u(x)$ intendiamo un polinomio di $\mathbb{F}_q[x]$ con $\deg u(x) < \deg g(x)$.
La successione $(c_i) := (c_i)_{i\in\mathbb{N}}$ è quella ottenuta sviluppando il rapporto tra $u(x)$ e $g(x)$. Con S_u rappresentiamo l'insieme delle parole del codice duale ottenute tramite la successione (c_i).

Dimostriamo ora i seguenti due teoremi.

Teorema 3.10. *Il numero di elementi di $S_u \subset C^\perp$ è*

$$M = \text{ord}\left(\frac{g(x)}{\gcd(g(x), u(x))}\right).$$

Dim. Per la teoria delle successioni a ricorrenza lineare (c_i) è una successione periodica. Quello che dobbiamo calcolare è il periodo che è anche il numero di elementi di S_u.

Quindi cerchiamo $M = \min_{m\in\mathbb{N}}\{u(x) \equiv x^m u(x) \pmod{g(x)}\}$.

La relazione $u(x) \equiv x^m u(x) \pmod{g(x)}$ è verificata se il polinomio $g(x)$ divide $u(x) - x^m u(x) = u(x)(1 - x^m)$. Questa osservazione equivale a dire che esiste $h(x) \in \mathbb{F}_q[x]$ tale che

$$u(x)(1 - x^m) = h(x)g(x) \qquad \Longleftrightarrow \qquad 1 - x^m = h'(x)\frac{g(x)}{\gcd(g(x), u(x))}$$

dove $h'(x) = h(x)/u'(x)$ con $u(x) = u'(x)/\gcd(g(x), u(x))$.

Riscrivendo la relazione di destra

$$x^m \equiv 1 \pmod{\frac{g(x)}{\gcd(g(x), u(x))}}$$

per la definizione di ordine di un polinomio, otteniamo:

$$M = \text{ord}\left(\frac{g(x)}{\gcd(g(x), u(x))}\right).$$

□

Ora abbiamo visto che con un unico polinomio $u(x)$ riusciamo ad elencare più parole. Con il seguente teorema vedremo invece come evitare di riconsiderare inavvertitamente le stesse parole.

Teorema 3.11. *Siano $u_1(x), u_2(x) \in \mathbb{F}_q[x]$ due polinomi di grado minore del grado di $g(x) \in \mathbb{F}_q[x]$. Allora esiste $j \in \mathbb{N}$ tale che*

$$u_2(x) \equiv x^j u_1(x) \pmod{g(x)}$$

se e soltanto se $S_{u_1} = S_{u_2}$.

Dim. Se esiste j tale che $u_2(x) \equiv x^j u_1(x) \pmod{g(x)}$ allora la successione per ricorrenza di $u_2(x)$ non è altro che quella di $u_1(x)$ iniziata però dal j-esimo termine quindi $S_{u_2} \subset S_{u_1}$. Dimostrando ora che i due insiemi hanno la stessa cardinalità otterremo la tesi. Sappiamo che $\gcd(x, g(x)) = 1$ e grazie a questo otteniamo

$$\begin{aligned} \#S_{u_2} &= \text{ord}\left(\frac{g(x)}{\gcd(g(x), u_2(x))}\right) = \text{ord}\left(\frac{g(x)}{\gcd(g(x), x^j u_1(x))}\right) \\ &= \text{ord}\left(\frac{g(x)}{\gcd(g(x), u_1(x))}\right) = \#S_{u_1} \end{aligned}$$

Viceversa se non esistesse nessun $j \in \mathbb{N}$ tale che $u_2 \equiv x^j u_1 \pmod{g(x)}$ allora vorrebbe dire che per nessun $j \in \mathbb{N}$ la successione definita per ricorrenza ottenuta da u_2 risulterà uguale a quella ottenuta da u_1 quindi per l'uguaglianza tra successioni $S_{u_1} \neq S_{u_2}$. □

Grazie a questo teorema notiamo che il nostro scopo è di decomporre l'anello $\mathcal{R}_g^q$ come unione delle orbite determinate dall'azione della moltiplicazione per x. Le orbite in $\mathcal{R}_g^q$ sono delle classi d'equivalenza, quindi, determinano una partizione di $\mathcal{R}_g^q$.

Considerando la corrispondenza che emerge da 3.4 e grazie alla teoria delle successioni definite per ricorrenza, anche gli insiemi del duale di un codice ciclico accorciato corrispondenti alle orbite di $\mathcal{R}_g^q$ costituiscono delle classi d'equivalenza.

3.3 x-orbite in $\mathcal{R}_g^q$

In questa sezione ci occupiamo della ricerca di rappresentanti per le orbite in $\mathcal{R}_g^q$.

Nella prima sottosezione illustriamo una parte di teoria sull'azione di gruppi. Questa ci serve per la definizione delle x-orbite in $\mathcal{R}_g^q$ e per lo studio di quest'ultime. Ci dedichiamo specialmente all'azione di un gruppo ciclico su un prodotto cartesiano, siccome, grazie ad esso e al Teorema Cinese del Resto, possiamo impiegare i nostri sforzi nella ricerca di rappresentanti di orbite in $\mathcal{R}_{g^t}^q$ dove $g(x)$ è un polinomio irriducibile.

Utilizziamo quindi questi risultati nella seconda sottosezione. Lo scopo della sottosezione è dare una forma esplicita di un rappresentante per ogni classe in $\mathcal{R}_{g^t}^q$ basata sui generatori trovati nel capitolo precedente. Per farlo, come nella ricerca dei generatori di $\mathcal{R}_{g^t}^q$, studieremo prima il caso in cui t è uguale a 1 e quindi tutti gli altri casi, cioè in cui t è maggiore o uguale a 2.

3.3.1 Azione di Gruppi Ciclici

Vediamo ora di trovare un elemento in ognuna delle orbite determinate dall'azione della moltiplicazione per x. Le orbite in $\mathcal{R}_g^q$ sono delle classi di $\mathcal{R}_g^q$. Sia

$$g(x) = \prod_{i=1}^{m} g_i(x)^{e_i}$$

la decomposizione in irriducibili di $g(x) \in \mathbb{F}_q[x]$. Per sviluppare al meglio questo argomento dovremo prima affrontare una parte di teoria riguardante l'azione

di un gruppo ciclico su un prodotto cartesiano. Troveremo poi i rappresentanti sfruttando il Teorema Cinese del Resto.

Sia quindi $G = (c)$ un gruppo ciclico, e sia E un insieme finito qualsiasi su cui G agisce.

Definizione 3.12. *Sia un punto $\xi \in E$, chiamiamo* orbita del punto ξ rispetto all'azione di $\mathcal{G}$ *il sottoinsieme*

$$\mathcal{G}(x) = \{g\xi | g \in G\}.$$

Sia ora E un insieme finito e siano T_1, T_2 due orbite di cardinalità rispettivamente d_1, d_2. Quello che vogliamo fare è caratterizzare tutte le possibili orbite che si ottengono tramite l'azione del gruppo ciclico G sul prodotto cartesiano

$$T = T_1 \times T_2.$$

Osservazione. Indicheremo con $\xi^{(k)}$ l'azione di $c^k \in G$ sull'elemento $\xi \in E$.

Teorema 3.13. *Siano T_1, T_2 due orbite ottenute dall'azione del gruppo $G = (c)$ sull'insieme finito E. Siano d_1, d_2 il numero di elementi rispettivamente dei due insiemi T_1, T_2. Se $t_1 \in T_1$ e $t_2 \in T_2$ sono, allora i*

$$(t_1, t_2^{(k)}) \in T$$

dove $0 \leq k \leq \gcd(d_1, d_2) - 1$, sono rappresentanti delle orbite di G su T, e le orbite contengono tutte $\text{lcm}(d_1, d_2)$ *elementi.*

Dim. Iniziamo col dimostrare che le orbite degli elementi $(t_1, t_2^{(k)})$ contengono esattamente $\text{lcm}(d_1, d_2)$ elementi. Dato che, per $i = 1, 2$, T_i ha cardinalità d_i, allora

$$t_i^{(l)} = t_i \iff d_i | l \qquad \text{con } i = 1, 2.$$

da questo segue che

$$t_i^{(l)} = t_i \qquad \text{con } i = 1, 2 \iff \text{lcm}(d_1, d_2) | l.$$

Con questo abbiamo dimostrato che le orbite hanno $\text{lcm}(d_1, d_2)$ elementi.

Le orbite sono tutte disgiunte formando classi d'equivalenza. Siccome il numero di elementi di T è $d_1 \cdot d_2$, allora il numero di orbite sarà $\gcd(d_1, d_2)$. Quello che ci rimane da dimostrare è che a differenti elementi $\left(t_1, t_2^{(k)}\right)$ corrispondono differenti orbite. Se così non fosse esisterebbero $k_1, k_2 \in \mathbb{N}$ con

$$k_1 - k_2 \not\equiv 0 \pmod{\gcd(d_1, d_2)}$$

tali che $\left(t_1, t_2^{(k_2)}\right) \in \mathcal{G}\left(\left(t_1, t_2^{(k_1)}\right)\right)$. Esisterebbe quindi $k_3 \in \mathbb{N}$ tale che

$$\left(t_1, t_2^{(k_2)}\right) = \left(t_1, t_2^{(k_1)}\right)^{(k_3)} = \left(t_1^{(k_3)}, t_2^{(k_1+k_3)}\right)$$

allora $d_1 | k_3$ e di conseguenza

$$k_2 = k_1 + k_3 = k_1 + h \cdot d_1 = k_1 + h' \gcd(d_1, d_2)$$

il che è assurdo. □

Quello che vogliamo fare ora è utilizzare il risultato precedente nel nostro caso. Per farlo quindi consideriamo il gruppo ciclico

$$(\mathrm{x}) = \{x^k | k \in \mathbb{N}\} \subset M_g^q.$$

Il prodotto cartesiano su cui facciamo agire il nostro gruppo è quello ottenuto grazie al Teorema Cinese del Resto, quindi

$$\mathcal{R}_g^q \cong \prod_{i=1}^{m} \mathcal{R}_{g_i^{e_i}}^q.$$

Vediamo come, grazie a questi risultati, ridurre il problema della ricerca di rappresentanti di orbite in $\mathcal{R}_g^q$ a quello dei rappresentanti di orbite nelle sue componenti $\mathcal{R}_{g_i^{e_i}}^q$.

Se consideriamo le proiezioni degli elementi che appartengono ad un'orbita dell'azione della moltiplicazione per x su $\mathcal{R}_g^q$, per il lemma 2.4, esse stesse genereranno un'orbita in $\mathcal{R}_{g_i^{e_i}}^q$, infatti

$$x^l u_i(x) \pmod{g_i(x)^{e_i}} = x^l u(x) \pmod{g_i(x)^{e_i}}$$

D'ora in avanti considereremo la seguente notazione:

$$\mathfrak{C}_u = \{x^k u(x) | u(x) \in \mathcal{R}_g^q,\ k \in \mathbb{N}\} \subset \mathcal{R}_g^q$$

l'orbita del polinomio $u(x) \in \mathcal{R}_g^q$.

Come corollario del teorema precedente faremo vedere quali orbite in $\mathcal{R}_g^q$ si possanno ottenere a partire dalle orbite $\mathfrak{C}_{u_i} \in \mathcal{R}_{g_i^{e_i}}^q$, $i = 1, \ldots, m$.

Corollario 3.14. *Sia $g(x) \in \mathbb{F}_q[x]$ con $g(x) = \prod_{i=1}^m g_i(x)^{e_i}$ la sua decomposizione in irriducibili. Siano $\mathfrak{C}_{u_i}$ orbite negli anelli $\mathcal{R}_{g_i^{e_i}}^q$, con $i = 1, \ldots, m$, di cardinalità d_i e di rappresentanti u_i. Allora i rappresentanti di tutte le orbite in $\mathcal{R}_g^q$ ottenibili tramite i rappresentanti precedenti sono*

$$\left(u_1, u_2^{(k_2)}, \ldots, u_m^{(k_m)}\right) \in \prod_{i=1}^m \mathcal{R}_{g_i^{e_i}}^q$$

dove $0 \leq k_i < \mathcal{K}_i$ con $i = 2, \ldots, m$ e

$$\mathcal{K}_i = \gcd(d_i, \mathit{lcm}(d_1, d_2, \ldots, d_{i-1})).$$

Dim. Per dimostrare questo corollario basta ricondurci alle ipotesi del teorema precedente. Il gruppo ciclico che agisce su $\mathcal{R}_g^q$ sarà come già detto (x).

Per il corollario al Teorema Cinese del Resto **??** sappiamo che $\mathcal{R}_g^q$ è isomorfo a $\prod_{i=1}^m \mathcal{R}_{g_i^{e_i}}^q$. Siano quindi, come da ipotesi, $\mathfrak{C}_{u_i} \subset \mathcal{R}_{g_i^{e_i}}^q$ delle orbite di rappresentanti u_i, dimostreremo quindi il teorema per induzione su i.

Se $i = 2$ la tesi corrisponde al teorema 3.13, quindi la base dell'induzione è dimostrata.

Sia quindi $s < m$, e supponiamo vero che gli elementi

$$(u_1, u_2^{(k_s)}, \ldots, u_s^{(k_s)})$$

siano rappresentanti di $\prod_{i=1}^{s} \mathcal{R}_{g_i^{e_i}}^q$ con k_i come da ipotesi. Dimostriamo che

$$(u_1, u_2^{(k_s)}, \ldots, u_s^{(k_s)}, u_{s+1}^{(k_{s+1})})$$

costituiscono l'insieme dei rappresentanti per $\prod_{i=1}^{s+1} \mathcal{R}_{g_i^{e_i}}^q$.

Consideriamo quindi $\mathfrak{u}$ un rappresentante di $\prod_{i=1}^{s} \mathcal{R}_{g_i^{e_i}}^q$ e sia $\mathfrak{C}_\mathfrak{u}$ l'orbita corrispondente. Quello che faremo è applicare il teorema 3.13 al prodotto

$$\mathfrak{C} = \mathfrak{C}_\mathfrak{u} \times \mathfrak{C}_{u_{s+1}}.$$

Grazie alle ipotesi del teorema e seguendo il ragionamento fatto all'interno della dimostrazione del teorema 3.13, la cardinalità di $\mathfrak{C}_\mathfrak{u}$ è

$$\text{lcm}(d_1, \ldots, d_s).$$

Quindi il numero di elementi del prodotto $\mathfrak{C}$ è $d_{s+1} \cdot \text{lcm}(d_1, \ldots, d_s)$. Il numero di elementi di un'orbita sarà

$$\text{lcm}(d_{s+1}, \text{lcm}(d_1, \ldots, d_s)) = \text{lcm}(d_1, \ldots, d_{s+1})$$

dove l'uguaglianza si dimostra banalmente.

Il numero quindi di orbite di $\mathfrak{C}$ sarà quindi

$$\#\mathfrak{C} = \frac{d_{s+1} \cdot \text{lcm}(d_1, \ldots, d_s)}{\text{lcm}(d_1, \ldots, d_{s+1})} = \gcd(d_{s+1}, \text{lcm}(d_1, \ldots, d_s)) = \mathcal{K}_{s+1}.$$

Quello che ci rimane da dimostrare è che dati $0 \leq k_{s+1} < k'_{s+1} < \mathcal{K}_{s+1}$ allora

$$\left(\mathfrak{u}, u_{s+1}^{(k'_{s+1})}\right) \notin \mathfrak{C}_{\left(\mathfrak{u}, u_{s+1}^{(k_{s+1})}\right)}.$$

Se così non fosse allora esisterebbe $r \in \mathbb{N}$ tale che

$$\left(\mathfrak{u}, u_{s+1}^{(k'_{s+1})}\right) = \left(\mathfrak{u}, u_{s+1}^{(k_{s+1})}\right)^{(r)} = \left(\mathfrak{u}^{(r)}, u_{s+1}^{(k_{s+1}+r)}\right),$$

allora avremmo che $d_i|r$ per $i = 1, \ldots, s$ e quindi $\text{lcm}(d_1, \ldots, d_s)|r$, ma allora

$$\begin{aligned} k'_{s+1} &= k_{s+1} + r = k_{s+1} + h \cdot \text{lcm}(d_1, \ldots, d_s) = \\ &= k_{s+1} + h' \cdot \gcd(d_{s+1}, \text{lcm}(d_1, \ldots, d_s)) \end{aligned}$$

che è assurdo. □

Il seguente corollario è conseguenza del precedente e del fatto che le orbite, per come definite, sono delle classi d'equivalenza.

Corollario 3.15. *Il numero di orbite di* $\prod_{i=1}^{m} \mathfrak{C}_{u_i}$ *è*

$$\mathcal{K} = \prod_{i=1}^{m} \mathcal{K}_i = \prod_{i=1}^{m} \frac{d_i \cdot \mathit{lcm}(d_1, \ldots, d_{i-1})}{\mathit{lcm}(d_1, \ldots, d_i)} = \frac{d_1 \cdots d_m}{\mathit{lcm}(d_1, \ldots, d_m)}$$

3.3.2 Le Orbite in $\mathcal{R}_{g^t}^q$

Per concludere la ricerca dei rappresentanti, ci è sufficiente caratterizzare i rappresentanti delle orbite in $\mathcal{R}_{g^t}^q$ con $g(x) \in \mathbb{F}_q[x]$ irriducibile e $t \geq 1$.

Analizziamo inizialmente il caso $t = 1$. Questo caso si divide in due: il caso in cui $g(x)$ è primitivo e il caso in cui non lo sia.

Il primo è il caso più semplice. Sia $u(x) \in \mathbb{F}_q[x]$, allora

$$\gcd(u(x), g(x)) = 1.$$

Siccome primitivo, $g(x)$ ha ordine $\text{ord}(g(x)) = q^r - 1$. Quindi il numero di parole che troviamo a partire da un qualsiasi polinomio non nullo $u(x) \in \mathbb{F}_q[x]$ saranno

$$M = \text{ord}\left(\frac{g(x)}{\gcd(g(x), u(x))}\right) = \text{ord}(g(x)) = q^r - 1.$$

Questo numero corrisponde all'ordine del gruppo moltiplicativo di $\mathcal{R}_g^q$. Nel caso di un polinomio primitivo quindi potremo prendere come rappresentanti un qualsiasi polinomio non nullo $u(x) \in \mathcal{R}_g^q$ e il polinomio zero.

Il secondo caso è quello in cui $g(x)$ non è primitivo. Per comodità di calcolo indicheremo con o_g l'ordine di $g(x)$. Per il teorema 1.17 sappiamo che o_g divide $q^r - 1$.

Per il teorema 1.4 esiste un generatore del gruppo moltiplicativo di $\mathcal{R}_g^q$, essendo quest'ultimo un campo. Sia $h(x)$ la rappresentazione di un elemento primitivo in $\mathcal{R}_g^q$ tale

$$x = h(x)^m \pmod{g(x)} \quad \text{con} \quad m = \frac{q^r - 1}{o_g}.$$

Si puo' scrivere $u(x)$ usando $h(x)$, cioè $u(x) = h(x)^i \pmod{g(x)}$, così l'orbita $\mathfrak{C}_u$ diventa

$$\mathfrak{C}_u = \{h(x)^k \pmod{g(x)} | \, k = i + sm, \; s \in \mathbb{N}\}.$$

Con la precedente relazione notiamo quindi che gli elementi di $\mathfrak{C}_u$, visti come potenze di $h(x)$, altro non sono che tutti quelli i cui esponenti sono congrui a i modulo m. Da questo otteniamo che l'insieme dei rappresentanti è

$$\{c_i(x) \in \mathcal{R}_g^q \mid c_i(x) = h(x)^i \pmod{g(x)} \text{ con } i = 0, 1, \ldots, m-1\} \cup \{0\}.$$

Analizziamo ora il caso in cui $t > 1$.
Sfruttiamo i generatori trovati per $\mathcal{R}_{g^t}^q$ nel capitolo precedente, per esplicitare i rappresentanti delle orbite.

Dimostriamo il seguente teorema.

Teorema 3.16. *Sia $u(x) \in \mathcal{R}_{g^t}^q$ e $\mathfrak{C}_u$ l'orbita da lui generata. Sia $s \in \mathbb{N}$ tale che*

$$u(x) = g(x)^s \bar{u}(x)$$

con $g(x)$ che non divide $\bar{u}(x)$, allora preso $u'(x) \in \mathcal{R}_{g^t}^q$,

$$u'(x) \in \mathfrak{C}_u \iff \begin{cases} g(x)^s \mid u'(x), \\ g^{s+1} \nmid u'(x), \\ u'(x)/g(x)^s \in \mathfrak{C}_{\bar{u}} \subset \mathcal{R}_{g^{t-s}}^q. \end{cases}$$

Dim. Se $u'(x) \in \mathfrak{C}_u$ allora vuol dire che esiste $l \in \mathbb{N}$ tale che

$$\begin{aligned} u'(x) &\equiv x^l u(x) \pmod{g(x)^t} \equiv x^l g(x)^s \bar{u}(x) \pmod{g(x)^t} \equiv \\ &\equiv \big(x^l \bar{u}(x) \pmod{g(x)^{t-s}}\big) g(x)^s. \end{aligned}$$

Quindi, siccome $\gcd\big(x, g(x)\big) = 1$ e che $\gcd\big(\bar{u}(x), g(x)\big) = 1$,

$$\gcd\big(x^l\bar{u}(x), g(x)\big) = \gcd\big(x^l\bar{u}(x) \pmod{g(x)^{t-s}}, g(x)\big) = 1$$

Viceversa se $s \in \mathbb{N}$ rappresenta il massimo esponente per il quale $g(x)^s \mid u'(x)$, vuol dire che possiamo trovare $\bar{u}'(x)$ tale che

$$u'(x) = g(x)^s\bar{u}'(x)$$

quindi, per ipotesi, esiste $l \in \mathbb{N}$ tale che

$$\begin{aligned} \bar{u}'(x) &\equiv x^l\bar{u}(x) \pmod{g(x)^{t-s}} \\ u'(x) = g(x)^s\bar{u}'(x) &\equiv \big(x^l\bar{u}(x) \pmod{g(x)^{t-s}}\big)g(x)^s \equiv \\ &\equiv x^l g(x)^s\bar{u}(x) \pmod{g(x)^t} = x^l u(x) \pmod{g(x)^t} \end{aligned}$$

□

Osservazione. Dal teorema si nota che i criteri per caratterizzare le orbite sono due

- se h è l'esponente della massima potenza di $g(x)$ che divide $u(x)$, allora h è l'esponente della massima potenza di $g(x)$ che divide tutti gli elementi contenuti nell'orbita $\mathfrak{C}_u$;
- la scelta dei rappresentanti di tutte le possibili orbite cade tra tutti gli elementi di $\mathcal{R}_{g^t}^q$ che sono primi con $g(x)$.

Grazie a questa osservazione giustifichiamo il lavoro fatto nel secondo capitolo, in cui abbiamo suddiviso l'anello $\mathcal{R}_{g^t}^q$ nel seguente modo

$$\mathcal{R}_{g^t}^q = \{0\} \cup \bigsqcup_{i=0}^{t-1} g(x)^i \cdot M_{g^{t-i}}^q.$$

Vediamo di precisare il tipo di azione che svolge la moltiplicazione per x in $\mathcal{R}_{g^t}^q$.

Il primo teorema che dimostreremo riguarderà l'ordine degli elementi $a_{i,j,k}(x)$, con $0 \leq i \leq \delta - 1$, $0 \leq j \leq r - 1$ e $1 \leq k \leq s - 1$ con $p \nmid k$, che generano il gruppo $M_{g^s}^p$ con $s = 0, 1, \ldots, t$.

Teorema 3.17. *Sia $s \in \mathbb{N}$ e sia $M_{g^s}^q$, allora gli elementi*

$$a_{j,k}(x) = 1 + \alpha^i x^j g(x)^k$$

con $0 \leq i \leq \delta - 1$, $0 \leq j \leq r - 1$ e $1 \leq k \leq s - 1$ con $p \nmid k$ hanno ordine

$$\mathrm{ord}(a_{i,j,k}(x)) = p^{\lceil \log_p \frac{s}{k} \rceil}$$

Dim. Per il teorema 2.10 sappiamo che gli elementi del tipo $a(x) = 1 + f(x)g(x)$ hanno ordine una potenza di p, Quindi anche i nostri elementi $a_{i,j,k}(x)$, per come definiti, hanno ordine una potenza di p. Tutto sta nel cercare la minima potenza che renda uguale a 1 il nostro polinomio.

Sia $m \in \mathbb{N}$, allora

$$a_{i,j,k}(x)^{p^m} = (1 + \alpha^i x^j g(x)^k)^{p^m} \equiv 1 + (\alpha^i x^j g(x)^k)^{p^m} \pmod{g(x)^s}.$$

Siccome $\gcd(\alpha^i x^j, g(x)) = 1$, allora

$$\begin{aligned} (\alpha^i x^j g(x)^k)^{p^m} \equiv 0 \pmod{g(x)^s} \quad &\iff \quad g(x)^{k \cdot p^m} \equiv 0 \pmod{g(x)^s} \\ &\iff \quad k \cdot p^m \geq s. \end{aligned}$$

A questo punto possiamo affermare che $i = \lceil \log_p \frac{s}{k} \rceil$, essendo

$$p^i = p^{\lceil \log_p \frac{s}{k} \rceil} > \frac{s}{k}.$$

Dato che a questo punto l'ordine di $a_{i,j,k}(x)$ deve dividere $p^{\lceil \log_p \frac{s}{k} \rceil}$ e dato che $\lceil \log_p \frac{s}{k} \rceil$, per la scelta fatta, corrisponde alla potenza minima, allora

$$\mathrm{ord}(a_{i,j,k}(x)) = p^{\lceil \log_p \frac{s}{k} \rceil}.$$

□

Ora passiamo all'ordine del polinomio x come elemento di $M_{g^s}^p$, $0 \leq s \leq t$.

Teorema 3.18. *Siano $s \in \mathbb{N}$, $M_{g^s}^q$ come da teorema precedente, allora l'ordine del polinomio x in $M_{g^s}^q$ è*

$$\mathrm{ord}(x) = \mathrm{ord}(g(x)) \cdot p^{\lceil \log_p s \rceil}.$$

Dim. Per definizione di ordine in $M_{g^s}^q$

$$x^{\mathrm{ord}(x)} \equiv 1 \pmod{g(x)^s} \;\Rightarrow\; g(x)^s \mid x^{\mathrm{ord}(x)} - 1. \tag{3.6}$$

Sia ora $\mathrm{ord}(x) = m = p^l \bar{m}$, con $p \nmid \bar{m}$, allora

$$x^m - 1 = x^{p^l \bar{m}} - 1 = (x^{\bar{m}} - 1)^{p^l}.$$

Dato che stiamo lavorando in caratteristica p, ogni fattore irriducibile di $x^m - 1$ ha molteplicità p^l. Per la relazione 3.6, otteniamo che la molteplicità di $g(x)$ deve essere almeno s, quindi $p^l \geq s$ il che vuol dire che $l \geq \lceil log_p s \rceil$, e quindi $l = \lceil log_p s \rceil$ siccome p^s è il massimo ordine possibile per un elemento di $M_{g^s}^q$.

Sappiamo inoltre che

$$g(x) \mid x^m - 1 \iff \mathrm{ord}(g(x)) \mid m.$$

Quindi l'ordine di x è il minimo comune multiplo tra $p^{\lceil \log_p s \rceil}$ e $\mathrm{ord}(g(x))$ che, essendo primi tra loro, è

$$\mathrm{ord}(g(x)) \cdot p^{\lceil \log_p s \rceil}.$$

□

Essendo l'ordine di x uguale al numero di parole contenute nel sottogruppo di $M_{g^s}^q$ generato da x stesso, che indicheremo con (x), ed essendo

$$\gcd(\mathrm{ord}(g(x)), p^{\lceil \log_p s \rceil}) = 1,$$

allora, una volta definiti

$$\mathrm{x}_p(x) := x^{\mathrm{ord}(g(x))} \text{ e } \mathrm{x}_{o_g}(x) := x^{p^{\lceil \log_p s \rceil}},$$

otteniamo che

$$(\mathrm{x}) = \big(\mathrm{x}_p(x)\big) \times \big(\mathrm{x}_{o_g}(x)\big),$$

Essenzialmente però gli elementi $\mathrm{x}_p(x)$ e $\mathrm{x}_{o_g}(x)$ sono degli elementi di $M_{g^s}^q$, quindi per i teoremi dimostrati nel secondo capitolo, e specialmente per i teoremi 2.9 e 2.11,

$$\mathrm{x}_p(x) = \prod_{i,j,k} a_{i,j,k}(x)^{c_{(ij)}^{(k)}} \qquad \text{e} \qquad \mathrm{x}_{o_g}(x) = a(x)^{\frac{q^r-1}{o_g}}$$

dove $M_{g^s}^q = M_g^q \times S_p$, $a(x)$ è un generatore di M_g^q, gli $a_{i,j,k}(x)$ sono i generatori del p-Sylow e con o_g indichiamo $\mathrm{ord}(g(x))$.

Dimostriamo il seguente teorema.

Teorema 3.19. *Sia* $\mathrm{x}_p(x)$ *un elemento di* $S_p \subset M_{g^s}^q$ *di ordine* $p^{\lceil \log_p s \rceil}$*, allora esistono* $0 \le i_0 < \delta - 1$, $0 \le j_0 < \deg g(x)$ *tali che l'insieme*

$$\{\mathrm{x}_p(x)\} \cup \{a_{i,j,k} | (i,j,k) \neq (i_0, j_0, 1)\}$$

è un insieme di generatori per S_p*.*

Dim. Il polinomio $\mathrm{x}_p(x)$ è un elemento di S_p e inoltre possiede ordine massimo in questo gruppo. Siccome l'insieme

$$\left\{a_{i,j,k}(x) \mid 0 \le i \le \delta - 1, 0 \le j \le \deg g(x) - 1,\ 1 \le k \le s,\ p \nmid k\right\},$$

dove con $\alpha \in \mathbb{F}_q$ indichiamo un elemento algebrico di ordine δ in $\mathbb{F}_p$, è un insieme di generatori di S_p per il teorema 2.11, allora esistono $c_{(ij)}^{(k)} \in \mathbb{N}$ tali che

$$\mathrm{x}_p(x) = \prod_{i,j,k} a_{i,j,k}^{c_{(ij)}^{(k)}}.$$

Avendo $\mathrm{x}_p(x)$ ordine massimo in S_p, dato che l'ordine dei polinomi $a_{i,j,k} \in S_p$ è $p^{\lceil \log_p \frac{s}{k} \rceil}$, allora esistono $0 \le i_0 < \delta - 1$ e $0 \le j_0 \le \deg g(x) - 1$ con tali che

$$\gcd(c_{(i_0 j_0)}^{(1)}, p) = 1.$$

Preso quindi un qualsiasi elemento $f(x) \in S_p$, esistono $e_{(ij)}^{(k)} \in \mathbb{N}$ tali che

$$f(x) = \prod_{i,j,k} a_{i,j,k}^{e_{(ij)}^{(k)}} = \mathrm{x}_p(x)^{\tilde{e}_p} \cdot \prod_{\substack{i,j,k \\ (i,j,k)\neq(i_0,j_0,1)}} a_{i,j,k}^{\tilde{e}_{(ij)}^{(k)}}$$

dove

$$\begin{aligned} \tilde{e}_p &\equiv e_{(i_0 j_0 1)} \cdot (c_{(i_0 j_0)}^{(1)})^{-1} \pmod{p^{\lceil \log_p s \rceil}} \\ \tilde{e}_{(ij)}^{(k)} &\equiv e_{(ij)}^{(k)} - c_{(ij)}^{(k)} \tilde{e}_p \pmod{p^{\lceil \log_p \frac{s}{k} \rceil}}. \end{aligned}$$

□

Vediamo ora un metodo per esplicitare i coefficienti i_0 e j_0. Per fare ciò lavoriamo nell'anello delle classi di resto di $g(x)^2$. Quindi partendo dalla espressione di $\mathrm{x}_p(x)$ come prodotto degli $a_{i,j,k}(x)$ e lavorando modulo $g(x)^2$ otteniamo

$$\begin{aligned} \mathrm{x}_p(x) &= \prod_{i,j,k} a_{i,j,k}^{c_{(ij)}^{(k)}} \equiv \\ &\equiv \prod_{i,j} a_{i,j,1}^{c_{(ij)}^{(1)}} \pmod{g(x)^2} \equiv \\ &\equiv \prod_{i,j} a_{i,j,1}^{c_{(ij)}^{(1)} \pmod{p}} \pmod{g(x)^2}. \end{aligned}$$

Analizziamo da vicino la moltiplicazione modulo $g(x)^2$ per quanto riguarda gli elementi di S_p. Dato che presi due elementi $s_1, s_2 \in S_p$, essi possono esser rappresentati come

$$s_i = 1 + \sigma_i(x) g(x) \qquad i = 1, 2$$

allora $s_1 \cdot s_2 \equiv 1 + (\sigma_1 + \sigma_2) g(x) \pmod{g(x)^2}$, quindi la moltiplicazione di due elementi di S_p corrisponde alla somma dei polinomi che li formano.

Se definiamo $\tilde{c}_{(ij)}^{(1)} = c_{(ij)}^{(1)} \pmod{p}$, allora

$$a_{(i,j,1)}^{\tilde{c}_{(ij)}^{(1)}} = (1 + \alpha^i x^j g(x))^{\tilde{c}_{(ij)}^{(1)}} \equiv 1 + \tilde{c}_{(ij)}^{(1)} \alpha^i x^j g(x) \pmod{g(x)^2}$$

a questo punto

$$\begin{aligned} \mathrm{x}_p(x) &\equiv \prod_{i,j} a_{i,j,1}^{\tilde{c}_{(ij)}^{(1)}} \pmod{g(x)^2} \\ &\equiv \prod_{i,j} (1 + \tilde{c}_{(ij)}^{(1)} \alpha^i x^j g(x)) \pmod{g(x)^2} \\ &\equiv 1 + \sigma(x) g(x) \pmod{g(x)^2} \end{aligned}$$

dove $\sigma(x) = \sum_j \tilde{c}_{(ij)}^{(0)} \alpha^i x^j$. I nostri i_0, j_0 potranno esser scelti tra tutti gli esponenti di $\sigma(x)$ con coefficiente diverso da zero.

Vediamo quindi di trovare, in virtù di quanto detto prima, i rappresentanti delle orbite di $\mathcal{R}_{g^t}^q$ con $t > 1$.

Grazie a quanto scritto finora possiamo formulare la condizione

$$u'(x) \in \mathfrak{C}_{u(x)} \iff \exists \nu \in \mathbb{N} : \; u'(x) \equiv x^\nu u(x) \pmod{g(x)^s}$$

nel seguente modo

$$u'(x) \in \mathfrak{C}_{u(x)} \iff \exists \nu, \mu \in \mathbb{N} : \; u'(x) \equiv \mathrm{x}_p(x)^\nu \cdot \mathrm{x}_{o_g}(x)^\mu \cdot u(x) \pmod{g(x)^s}.$$

Sfrutteremo ora il fatto che $\mathrm{x}_p(x)$ può sostituire $a_{i_0,j_0,1}$ nell'insieme dei generatori di $M_{g^s}^p$ per ottenere la formulazione definitiva dei rappresentanti.

Teorema 3.20. *Sia $g(x) \in \mathbb{F}_q[x]$ un polinomio irriducibile e di grado r e $t \geq 2$. Dei rappresentanti delle orbite in $\mathcal{R}_{g^t}^q$ sono gli elementi*

$$u_{i,j,k,l} \equiv a(x)^l \prod_{\substack{i,j,k \\ (i,j,k) \neq (i_0,j_0,1)}} (1 + \alpha^i x^j g(x)^k)^{c_{(ij)}^{(k)}} \pmod{g(x)^s}$$

con

- $0 \leq l \leq \frac{q^r - 1}{o_g} - 1$,
- $0 \leq i < \delta$,
- $0 \leq j < r$,

- $1 \leq k \leq t$, *con* $p \nmid k$ *e*
- $0 \leq c_{(ij)}^{(k)} \leq p^{\lceil \log_p \frac{s}{k} \rceil}$.

Capitolo 4

Struttura dell'Algoritmo

L'unico metodo, indipendente dalla struttura del codice scelto, per il calcolo della distribuzione dei pesi di un codice, passa attraverso il calcolo del peso di ogni parola del codice. Questo metodo è computazionalmente troppo costoso.

Esibiamo, in questo capitolo, l'algoritmo per il calcolo della distribuzione dei pesi del duale di un codice ciclico accorciato.

La caratterizzazione, tramite una relazione lineare, delle parole del codice duale di un codice ciclico accorciato, collega quest'ultime alle successioni a ricorrenza lineare. Infatti da una qualsiasi successione di ricorrenza uguale a quella che caratterizza le parole del codice duale, possiamo estrarre una parola del duale. Viceversa da una parola del duale, applicando continuamente la relazione, possiamo costruire una successione a ricorrenza lineare.

In realtà più di una parola può essere estratta dalla stessa successione. Infatti una qualsiasi stringa di n coefficienti consecutivi della successione, dove n è la lunghezza del codice, costituiscono una parola del codice duale.

Abbiamo dimostrato pure l'esistenza di una corrispondenza biunivoca tra le successioni a ricorrenza lineare e gli elementi di $\mathcal{R}_g^q$.

Passiamo ora a caratterizzare il passo decisivo dell'algoritmo, cioè quello utilizzato per calcolare la distribuzione di pesi del duale di un codice ciclico accorciato. Lo analizziamo nel caso più semplice possibile, cioè nel caso in cui il polinomio generatore sia irriducibile e primitivo.

La scelta fatta, cioè di analizzare inizialmente il caso di un polinomio primitivo, non è casuale. Infatti questo è il caso affrontato nell'articolo di Fujiwara e che

ha ispirato quello di Castagnoli.

Con questo esempio vogliamo far capire il funzionamento dell'algoritmo.

Sia quindi $g(x)$ un polinomio primitivo con $\deg g(x) = r$. Sia $(c_i)_{i\in\mathbb{N}}$ una successione a ricorrenza lineare di polinomio caratteristico $g(x)$. Per la teoria delle successioni a ricorrenza lineare il periodo di $(c_i)_{i\in\mathbb{N}}$ è $q^r - 1$.

Il numero $q^r - 1$ è pure il numero di parole del duale del codice ciclico accorciato di polinomio generatore $g(x)$. Vediamo quindi come calcolare il peso di tutte le parole del codice duale ottenibili tramite la successione $(c_i)_{i\in\mathbb{N}}$.

La prima parola del codice viene formata considerando le prime n componenti della successione $(c_i)_{i\in\mathbb{N}}$. Calcoliamo il peso di questa parola. Il peso delle successive parole che otteniamo tramite la successione, che sono in numero uguale a quello della cardinalità dell'orbita di $u(x)$ e quindi uguali alla cardinalità del codice duale, viene calcolato in funzione del peso della prima parola, lasciandolo invariato, oppure aggiungendo o sottraendo 1. Vediamo più praticamente:

- estraiamo $c^{(0)} = (c_0, \ldots, c_{n-1}) \in C$ la prima parola e ne calcoliamo il peso $wt(c^{(0)})$;
- la seconda parola ottenuta è $c^{(1)} = (c_1, \ldots, c_n) \in C$ quindi:
  1. se $c_0 = 0$ e $c_n \neq 0$, allora $wt(c^{(1)}) = wt(c^{(0)}) + 1$;
  2. se $c_0 \neq 0$ e $c_n = 0$, allora $wt(c^{(1)}) = wt(c^{(0)}) - 1$;
  3. altrimenti $wt(c^{(1)}) = wt(c^{(0)})$.

Continuiamo con questo procedimento per tutte le parole successive fino a che non raggiungiamo il periodo della successione. A questo punto otteniamo la distribuzione dei pesi del codice duale.

Il grande risparmio computazionale non sta nel trovare le parole, bensì nel metodo di calcolo del loro peso. Infatti grazie al calcolo del peso di una sola parola siamo riusciti ad ottenere l'intera distribuzione del codice duale.

Questo è il caso più semplice, cioè quello tramite il quale tutte le parole del codice duale possono essere calcolate a partire da una sola successione.

Generalizziamo i ragionamenti fatti. Quello che dobbiamo fare è riuscire a determinare le successioni necessarie e sufficienti per trovare tutte le parole del codice duale. Una volta trovate tali successioni, applicando il metodo spiegato per il caso primitivo, arriveremo a calcolare la distribuzione dei pesi del codice.

Grazie alla corrispondenza che c'è tra l'anello $\mathcal{R}_g^q$ e le successioni a ricorrenza lineare, invece di ragionare sull'insieme delle successioni a ricorrenza lineare, lavoriamo direttamente sull'anello quoziente $\mathcal{R}_g^q$. Le x-orbite formano una partizione in classi d'equivalenza di $\mathcal{R}_g^q$. Sia quindi la successione $(c_i)_{i\in\mathbb{N}}$ definita tramite lo sviluppo in serie di potenze formali della divisione $u(x)/g(x)$. Abbiamo dimostrato l'esistenza di una corrispondenza tra l'x-orbita $\mathfrak{C}_u$ e l'insieme di tutte le parole del duale calcolabili tramite la successione $(c_i)_{i\in\mathbb{N}}$. Il nostro lavoro si concentra quindi nella partizione in orbite di $\mathcal{R}_g^q$. Più specialmente cerchiamo un elemento per ogni orbita grazie al quale calcoliamo il peso delle parole del codice duale da esso ottenute. Una volta trovati tutti questi elementi e calcolato il peso delle parole ad essi correlate, otterremo la distribuzione dei pesi del codice duale.

Grazie all'insieme di generatori di $\mathcal{R}_g^q$ trovato nel secondo capitolo possiamo esplicitare i rappresentanti delle orbite.

Diamo nella prossima sezione, in modo dettagliato, lo schema dell'algoritmo.

4.1 Algoritmo

INPUT:

- $g(x) \in \mathbb{F}_q[x]$ polinomio generatore del codice CRC;
- $n \in \mathbb{N}$, con $n < q^{degg(x)} - 1$, lunghezza del codice;
- $0 \leq \epsilon \leq \frac{q-1}{q}$ probabilità di errore di una singola componente.

OUTPUT:

- $\{A_0, \ldots, A_n\}$ distribuzione dei pesi del codice CRC;
- $P_{ue}(\epsilon)$, *Undetected Error Probability*;
- d_{min}^C, distanza minima del codice CRC;
- una delle seguenti affermazioni: codice proprio, codice probabilmente proprio o codice non proprio.

PASSO 1: Fattorizzazione in irriducibili del polinomio $g(x)$:

$$g(x) = \prod_{h=1}^{m} g_h(x)^{e_h}.$$

PASSO 2: Calcolo dell'ordine del polinomio $g(x)$.

- Al variare di i tra 0 e m calcolo l'ordine dei polinomi $g_h(x)$ della fattorizzazione del PASSO 1:
  $m_h = \deg g_h(x)$ e fattorizziamo $q^{m_h} - 1$:

  $$q^{m_h} - 1 = \prod_{j=0}^{s} p_j^{r_j};$$

  Al variare di j tra 0 e s procediamo in modo iterativo come segue:
  - $n_h := \dfrac{q^{m_h} - 1}{p_j}$;
  - Calcoliamo $h(x) \equiv x^{n_h} - 1 \pmod{g_h(x)}$, ci sono due possibilità:
    1. $h(x) \neq 0$: in questo caso passiamo al j successivo;
    2. $h(x) = 0$: in questo caso $\mathrm{ord}(g_h(x))$ non è multiplo di $p_j^{r_j}$ e quindi $\mathrm{ord}(g_h(x)) \mid \dfrac{q^{m_h} - 1}{p_j}$. Calcoliamo allora di nuovo $h(x)$, sostituendo n_h con $\dfrac{n_h}{p_j}$, fino a quando il nuovo $h(x)$ soddisfa 1;
    3. una volta finito il passo $j = s$, n_h sarà l'ordine di $g_h(x)$.

  Finito questo passaggio otteniamo tutti gli $n_h = \mathrm{ord}(g_h(x))$;
- calcoliamo $e = \mathrm{lcm}(n_1, \ldots, n_m)$;
- cerchiamo $t = \max(e_1, \ldots, e_m)$;
- L'ordine del polinomio $g(x)$ è:

  $$\mathrm{ord}(f(x)) = ep^t,$$

  dove p è la caratteristica del campo $\mathbb{F}_q$.

(*si veda sezione 1.2*).

PASSO 3: Calcolo dei rappresentanti delle orbite dell'anello $\mathcal{R}_g^q$.
Al variare di h tra 1 e m, procediamo con i seguenti sottopassi.

PASSO 3.1: Calcolo dei generatori di $M_{g_h}^q$.
Siccome $g_h(x)$ è irriducibile, $M_{g_h}^q$ è ciclico perchè gruppo moltiplicativo di un campo. Abbiamo la suddivisione in due casi:

1. se $n_h = q^{m_h} - 1$: il polinomio $g_h(x)$ è primitivo. Allora il polinomio x è un generatore di $M_{g_h}^q$ per la definizione di polinomio primitivo,

$$M_{g_h}^q = (\mathrm{x});$$

2. se $n_h < q^{m_h} - 1$: il polinomio $g_h(x)$ non è primitivo. procediamo con la ricerca di un elemento primitivo $h(x)$ di $M_{g_h}^q$:

$$M_{g_h}^q = (h(x)) \,.$$

(*si veda sezione 2.2*)

PASSO 3.2: Calcolo dei rappresentanti delle orbite di $M_{g_h}^q$.
Anche questo passo si divide in due.

1. $g_h(x)$ primitivo. Il numero di elementi dell'orbita di un qualsiasi elemento $u(x) \in M_{g_h}^q$ è

$$M = \mathrm{ord}\left(\frac{g_h(x)}{\gcd(g_h(x), u(x))}\right) = \mathrm{ord}(g_h(x)) = q^{m_h} - 1.$$

   Quindi $M_{g_h}^q$ è costituito da un'unica orbita indipendente dalla scelta del rappresentante. Per semplicità prendiamo come rappresentante il polinomio x.

2. $g_h(x)$ non primitivo.

   (a) calcoliamo dell'ordine di un'orbita

$$M = \mathrm{ord}\left(\frac{g_h(x)}{\gcd(g_h(x), u(x))}\right) = \mathrm{ord}(g_h(x)) := o_{g_h} \mid q^{m_h} - 1,$$

indipendente quindi dalla scelta del rappresentante. Il gruppo moltiplicativo $M^q_{g_h}$ si divide quindi in $(q^{m_h}-1)/o_{g_h}$ orbite disgiunte.

(b) calcoliamo l'insieme dei rappresentanti delle orbite. Per quanto mostrato nella sezione 3.3.2, l'insieme dei rappresentanti delle orbite di $M^q_{g_h}$ è

$$\left\{ h(x)^k \pmod{g_h(x)} \mid 0 \leq i \leq \frac{q^{m_h}-1}{o_{g_h}} \right\}.$$

(*si veda sottosezione 3.3.2*)

PASSO 3.3: Calcolo dei rappresentanti di $\mathcal{R}^q_{g_i^{e_h}}$.

Per il teorema 2.8 sappiamo che

$$\mathcal{R}^q_{g_i^{e_h}} = \{0\} \cup \bigsqcup_{s=2}^{e_h} g_h(x)^{e_h-s} \cdot M^q_{g_h^s}.$$

Al variare di s tra 2 e e_h procediamo con i seguenti quattro passi.

PASSO 3.3.1: Calcolo dei generatori di $M^q_{g_h^s}$.

Il gruppo moltiplicativo $M^q_{g_h^s}$ si divide nel seguente modo

$$M^q_{g_h^s} \approx M^q_{g_h} \times S_p.$$

1. generatore di $M^q_{g_h}$: la situazione è la stessa del PASSO 3.1;
2. generatori di S_p:
   - il gruppo S_p è un gruppo abeliano finito e quindi può essere scomposto in prodotto di gruppi ciclici, vedi Teorema di classificazione dei Gruppi Abeliani Finiti 1.1
   - ricerca di un elemento $\alpha \in \mathbb{F}_q$, tale che il gruppo costituito dalle sue potenze nel gruppo moltiplicativo $\mathbb{F}_q^*$ abbia ordine esattamente δ dove $q = p^\delta$. Le potenze di questo elemento costituiranno una base per il campo $\mathbb{F}_q$ visto come spazio vettoriale su $\mathbb{F}_p$;
   - grazie al teorema 2.11 i generatori di S_p, il p-gruppo di Sylow,

sono tutti quelli della forma

$$a_{i,j,k}(x) = 1 + \alpha^i x^j g_h(x)^k$$

dove

- $0 \leq i \leq \delta - 1$;
- $0 \leq j \leq m_h - 1$;
- $1 \leq k \leq s - 1$, con $p \nmid k$.

(*si veda teorema 2.11*)

PASSO 3.3.2: Calcolo dei rappresentanti delle orbite di $M^q_{g_h^s}$.

- Per la eplicitare i rappresentanti delle orbite di $M^q_{g_h^s}$ dobbiamo prima trovare la rappresentazione di $x \in M^q_{g_h^s}$ nella base del PASSO 3.3.1:
  - per il teorema 3.18 sappiamo che

    $$\mathrm{ord}(x) = \mathrm{ord}(g_h(x))p^{\lceil \log_p s \rceil}.$$

    Il numero $\mathrm{ord}(x)$ è il numero di elementi delle orbite da esso generate tramite l'azione su un polinomo di $M^q_{g_h^s}$.
    Il gruppo da lui generato si decompone nel prodotto

    $$(\mathrm{x}) \approx (\mathrm{x}_p(x)) \times (\mathrm{x}_{o_{g_h}}(x)).$$

  - calcoliamo l'elemento $a_{i_0,j_0,1} \in M^q_{g_h^s}$ come da teorema 3.19. Il procedimento per il calcolo di questo elemento è:
    - calcoliamo $\mathrm{x}_p(x) \equiv 1 + f(x)g_h(x) \pmod{g_h(x)^2}$;
    - se $f(x) = \sum_{i,j} \alpha^i x^j$, allora

      $$1 + f(x)g_h(x) \equiv (1 + \alpha^i x^j g_h(x)) \pmod{g_h(x)^2}.$$

      Scegliamo $i_0,\ j_0$, tra i monomi che formano $f(x)$.

    (*si veda teorema 3.19*)
  - considerato $h(x) \in M^q_{g_h}$ un elemento primitivo e come da PASSO

3.1, allora:

$$\mathrm{x}_{o_{g_h}}(x) = h(x)^{\frac{q^{m_h}-1}{o_{g_h}}}.$$

(*si veda sottosezione 3.3.2*)

- Calcoliamo l'ordine degli elementi $a_{i,j,k} \in M^q_{g_h^s}$:

$$\mathrm{ord}(a_{i,j,k}) = p^{\lceil \log_p \frac{s}{k} \rceil} =: \eta_{ijk}.$$

(*si veda teorema 3.17*)

- Calcoliamo i rappresentanti delle orbite di $M^q_{g_h^s}$.

$$u_{i,j,k,l} = h(x)^l \prod_{\substack{i,j,k \\ (i,j,k)\neq(i_0,j_0,1)}} a_{i,j,k}^{c_{(ijk)}} \pmod{g_h(x)^s}$$

dove $0 \leq l \leq \frac{q^{m_h}-1}{o_g} - 1$ e $0 \leq c_{(ijk)} \leq \eta_{ijk}$.
(*si veda sottosezione 3.3.2*)

PASSO 3.3.3: Calcolo dei rappresentanti di $\mathcal{R}^q_{g_i^{e_h}}$.
Finito di elencare tutti i rappresentanti dei gruppi moltiplicativi $M^q_{g_h^s}$ con $0 \leq 1 \leq e_h$, li moltiplichiamo per l'opportuna potenza di $g_h(x)$, cioè $e_h - s$, per trasformarli in rappresentanti di orbite di $\mathcal{R}^q_{g_i^{e_h}}$.

PASSO 3.4: Calcolo dei rappresentanti delle orbite di $\prod_{h=1}^m \mathcal{R}^q_{g_i^{e_h}}$ tramite il Teorema Cinese del Resto.

- come da teorema 3.14, calcoliamo i numeri $\mathcal{K}_h$ dove il numero degli elementi delle orbite d_h sono stati registati al PASSO 3.3.2;
- calcoliamo i rappresentanti partendo da quelli trovati per ogni singolo irriducibile $g_h(x)$ che fa parte della fattorizzazione di $g(x)$. La relazione per il calcolo dei rappresentanti delle orbite di $\mathcal{R}^q_g$ è

$$\prod_{h=1}^{m} u_h^{k_h} \in \mathcal{R}^q_g$$

con $k_1 = 1$, $0 \leq k_h < \mathcal{K}_h$ e $u_i \in \mathcal{R}^q_{g_i^{e_h}}$ rappresentanti di orbite.

(*si veda corollario 3.14*)

PASSO 4 Calcolo della distribuzione dei pesi del codice duale.

Consideriamo, uno alla volta, i rappresentanti calcolati nel PASSO 3.4.

1. Calcoliamo i primi n termini di $c(1/x)$ grazie alla relazione (3.4). Otteniamo una parola del codice duale di cui calcoliamo il peso. Il procedimento iniziato per il calcolo degli elementi di $c(1/x)$ lo indichiamo con $LFSR_1$ (da *Linear Feedback Shift Register*) e l'i-esimo stato lo indichiamo con $LFSR_1(i)$.
2. A questo punto $LFSR_1$ si trova al suo $n-1$-esimo stato. Faccio partire in parallelo un secondo procedimento $LSFR_2$ che calcola anch'esso i coefficienti della successione $c(1/x)$.
3. Le successive parole del codice duale si ottengono *shiftando* consecutivamente gli n elementi della successione. Quindi se

   $$c^{(i)} = (c_i, \ldots, c_{i+n-1})$$

   è l'i-esima parola estratta da $c(1/x)$, quella successiva sarà

   $$c^{(i+1)} = (c_{i+1}, \ldots, c_{i+n}).$$

4. i coefficienti dell'i-esima parola differiscono dalla $(i+1)$-esima nella perdita di $c_i = LFSR_2(i)$ a favore di $c_{i+n} = LFSR(i+n)$. Quindi il calcolo del peso della $(i+1)$-esima parola si divide in tre casi:
   (a) se $LFSR_1(i+n) \neq 0$ e $LFSR_2(i) = 0$: $wt(c^{(i+1)}) = wt(c^{(i)}) - 1$;
   (b) se $LFSR_1(i+n) = 0$ e $LFSR_2(i) \neq 0$: $wt(c^{(i+1)}) = wt(c^{(i)}) + 1$;
   (c) altrimenti $wt(c^{(i+1)}) = wt(c^{(i)})$.

(*si veda sezione 3.2*)

OUTPUT: Distribuzione del codice duale.

PASSO 5: Verifica della condizione sufficiente affinché C sia un codice proprio. Questo passo lo verifichiamo calcolando i B_i^*. Se questi numeri verificano la relazione del teorema 1.51 allora il codice è proprio.
(*si veda teorema 1.51*)

OUTPUT: codice proprio o meno.

PASSO 6: Calcolo di $P_{ue}(\epsilon)$ basato sulla distribuzione dei pesi del codice duale. (*si veda sottosezione 1.3.3*)

OUTPUT: $P_{ue}(\epsilon)$.

PASSO 7: Applicazione del Teorema di MacWilliams 1.36 per il calcolo della distribuzione del codice CRC a partire da quella del codice duale.
Per il calcolo della distribuzione del codice sfruttiamo i polinomi di Krawtchouk e la relazione per ricorrenza dimostrata per calcolarli. (*si veda sottosezione 1.3.1*)

OUTPUT: distribuzione del codice e d^C_{min}.

Appendice A

Implementazione dell'Algoritmo

A.1 La Libreria NTL

NTL fornisce un'implementazione di alta qualità di algoritmi allo stato dell'arte per:

- aritmetica per interi di lunghezza arbitraria ed aritmetica in virgola mobile a precisione arbitraria;
- aritmetica per polinomi a coefficienti negli interi ed in campi finiti, per esempio: aritmetica di base, fattorizzazione di polinomi, test per l'irriducibilità, calcolo di polinomi minimi, tracce, norme;
- riduzione a base di reticoli, inclusa un'implementazione di Schnorr-Euchner molto stabile e veloce, riduzione a blocchi di Korkin-Zolotarev;
- algebra lineare di base su interi, campi finiti e numeri a virgola mobile di precisione arbitraria.

L'aritmetica polinomiale NTL permette di ottenere sia fattorizzazioni polinomiali sia l'ordine di curve ellittiche in modo più veloce possibile tra quelli disponibili.

Il codice per la riduzione a base di reticoli è inoltre uno dei migliori disponibili in termini di velocità e stabilità ed è stato usato per "rompere" alcuni criptosistemi.

NTL fornisce un'interfaccia coerente e pulita per una larga varietà di classi di oggetti matematici. È particolarmente indicato per lo sviluppo di nuovi algoritmi per la teoria dei numeri.

NTL è scritto e mantenuto principalmente da Victor Shoup.

La versione utilizzata nella seguente implementazione è la 5.3.2 del 21 maggio 2004.

A.2 Implementazione dell'Algoritmo

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <ctime>
#include <NTL/GF2XFactoring.h>
#include <NTL/vec_GF2.h>
#include <NTL/vector.h>
#include <NTL/RR.h>

using namespace std;

int n;
ZZ order_change;
GF2X pol_gen;
GF2X unity;
GF2X identity;
double dif_com;

//////////  POWER  //////////

ZZ power(int n, int k)
{
  ZZ power = to_ZZ(1);
  ZZ s = to_ZZ(n);
  ZZ r = to_ZZ(k);
  while (r!=0)
    {
      if (IsOdd(r))
        power *= s;
      r>>=1;
      s*=s;
    }
  return power;
}

//////////  POWER GF2X  //////////
```

```
GF2X powerGF2X(GF2X pol, int n)
{
  GF2X power = unity;
  if (!n)
    return power;
  else if (n==1)
    {
      return pol;
    }
  else
    {
      do
        {
          if (n%2)
            {
              power *= pol;
              n = (n-1)/2;
            }
          else n /= 2;
          pol *= pol;
        }
      while(n!=1);
      power *= pol;
      return power;
    }
}

//////////  LCM IN ZZ  //////////

ZZ Lcm_ZZ(ZZ* numeri)
{
  ZZ lcm = numeri[0];
  for (; !(IsZero(*numeri)); numeri++)
    lcm = (lcm * (*numeri))/GCD(lcm,(*numeri));
  return lcm;
}

//////////  QSORT FUNCTION  //////////

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}


//////////  POL CONVERSION  //////////

void inputConversion(int n, char pol[])
{
  int l;
```

```
l = strlen(pol);
int k = 0;

//CICLO FOR PER ELIMINARE GLI SPAZI
for (int i = 0; i < l; i++)
  {
    if (pol[i] != 32)
      pol[i-k] = pol[i];
    else
      {
        pol[i-k] = pol[i+1];
        k++;
      }
  }

l -= k;
pol[l+1] = '\0';

//CREAZIONE DI UN VETTORE DI INTERI CON GLI ESPONENTI DI x
int potenze[500];
int g = 0;

for (int i = 0; pol[i] != '\0' ; i++)
  {
    if (pol[i] == 'x' && pol[i+1] == '^')
      {
        int j = i+2, k = 0;
        while (pol[j] != '+' && pol[j] != '-' && pol[j] != '\0')
          {
            k++;
            j++;
          }

        j=i+2;
        int z = 0;
        char in_int[3];
        for (int t = k-1 ; t>=0; t--)
          {
            in_int[z] = pol[j];
            z++;
            j++;
          }
        in_int[z] = '\0';
        potenze[g] = atoi(in_int);
        g++;
      }
    else if (pol[i] == 'x' && pol[i+1] != '^')
      {
        potenze[g] = 1;
        g++;
      }
```

```
      else if (pol[i]=='1' && pol[i+1]=='+' || pol[i+1]=='-'
               || pol[i+1]=='\0')
        {
          if (i==0)
            {
              potenze[g] = 0;
              g++;
            }
          else if (pol[i-1]=='+' || pol[i-1]=='-')
            {
              potenze[g] = 0;
              g++;
            }
        }
      else if (pol[i]!='0' && pol[i]!='1' && pol[i]!='2' && pol[i]!='3'
               && pol[i]!='4' && pol[i]!='5' && pol[i]!='6' && pol[i]!='7'
               && pol[i]!='8' && pol[i]!='9' && pol[i]!='+' && pol[i]!='-'
               && pol[i]!='\0' && pol[i]!='^' && pol[i] != 'x')
        exit (EXIT_FAILURE);
    }
  potenze[g] = '\0';

  //ORDINIAMO IL VETTORE
  qsort (potenze, g, sizeof(int), compare);

  // PASSAGGIO A GF2X
  int p = 0;
  vec_GF2 pol_out (INIT_SIZE,n+1);

  for (int i = 0; i<n+1; i++)
    {
      if (i == potenze[p])
        {
          pol_out[i] = 1;
          p++;
        }
      else pol_out[i] = 0;
    }

  conv (pol_gen,pol_out);

  return;
}


//////// FACTORIZATION IN GF2X //////////

vec_pair_GF2X_long factorization_GF2X(GF2X f)
{
  vec_pair_GF2X_long factors;
  CanZass(factors, f);
```

```
  return factors;
}

//////////  FACTORIAL  //////////

RR factorial(int n)
{
  RR factor= to_RR(1);
  if (n==0 || n==1)
    return factor;
  else
    {
      for(int i=1; i<=n; i++)
        factor *= i;
      return factor;
    }
}

//////////  WEIGHT DISTRIBUTION  //////////

class weightDistribution
{
public:

  ZZ *weight_dist;
  ZZ *weight_dist_ZZ;

  void compute_distribution(GF2X pol_den, GF2X pol_num, int n, int l);
  void McWilliams(int n, int, ZZ* dual_weight);
  int testProper(int length, int dmin, GF2X pol_gen, ZZ* dual_weight);
};


void weightDistribution::compute_distribution(GF2X pol_den, GF2X pol_num,
                                              int n, int l)
{
  if (weight_dist)
    delete[] weight_dist;

  weight_dist = new ZZ[l+1];

  for (int i = 0; i < l+1; i++)
    weight_dist[i] = to_ZZ(0);

  GF2X pol_shift,pol_shift_2;

  pol_shift = pol_num;
  int w = 0;

  for (int i = 0; i!= l; i++)
    {
```

```
      if (deg(pol_shift) == n-1)
        w += 1;
      MulByXMod(pol_shift, pol_shift, pol_den);
    }

  pol_shift_2 = pol_num;

  do
    {
      int deg1=deg(pol_shift);
      int deg2=deg(pol_shift_2);

      if (deg1 == n-1 && deg2 == n-1 || deg1 != n-1 && deg2 != n-1)
        weight_dist[w]++;
      else
        {
          if (deg1 == n-1 && deg2 != n-1)
            {
              w++;
              weight_dist[w]++;
            }
          else
            {
              w--;
              weight_dist[w]++;
            }
        }

      MulByXMod(pol_shift, pol_shift, pol_den);
      MulByXMod(pol_shift_2, pol_shift_2, pol_den);
    }
  while (pol_shift_2 != pol_num);
}

void weightDistribution::McWilliams(int n, int l, ZZ* dual_weight)
{
  if (weight_dist_ZZ)
    delete[] weight_dist_ZZ;

  weight_dist_ZZ = new ZZ[l+1];

  for (int i = 0; i < l+1; i++)
    weight_dist_ZZ[i] = to_ZZ(0);

  for (int i=0; i <= l; i++)
    {
      ZZ kraw, kraw0, kraw1;

      kraw0 = to_ZZ(1);
      kraw1 = l-2*i;
```

```
      for (int k = 0; k <= l; k++)
        {
          if (!k)
            kraw = kraw0;
          else if (k==1)
            kraw = kraw1;
          else
            {
              kraw = ((l-2*i) * kraw1 - (l-k+2) * kraw0)/k;
              kraw0 = kraw1;
              kraw1 = kraw;
            }

          weight_dist_ZZ[k] += dual_weight[i]*kraw;
        }
    }

  for (int i=0; i<=l; i++)
    weight_dist_ZZ[i] /= power_ZZ(2,n);
}

int weightDistribution::testProper(int length, int dmin, GF2X pol_gen,
                                   ZZ* dual_weight)
{
  int t=2;
  RR temp = to_RR(0), pue=to_RR(0);
  for (int v=1; v<=500; v++)
    {
      RR epsilon = to_RR(v)*.001;
      pue=to_RR(1);
      for (int i=1; i<length+1; i++)
        {
          pue += to_RR(dual_weight[i])*power(1-2*epsilon,i);
        }
      pue /= power(to_RR(2),n);
      pue -= power(1-epsilon,length);

      if (pue < temp)
        {
          t = 3;
          break;
        }
      else
        temp = pue;
    }

  if (t==2)
    {
      t=1;
      int k = length-deg(pol_gen);
      int s = length-dmin-1;
```

```
      RR tempor = to_RR(0), tempor1 = to_RR(0), tempor2;

      for (int i=1; i<=s; i++)
        tempor+=factorial(length-i)*to_RR(dual_weight[i])/factorial(s-i);
      tempor *= factorial(s);
      tempor /= factorial(length);

      for (int j=s+1; j<length+1; j++)
        {
          for (int i=1; i<=j; i++)
            tempor1 += factorial(length-i)*to_RR(dual_weight[i])
                       /factorial(j-i);
          tempor1 *= factorial(s);
          tempor1 /= factorial(length);

          tempor2 =tempor1-power2_RR(j-1-k);
          if (tempor < tempor2)
            {
              t=2;
              break;
            }
          else
            tempor=tempor1;
        }
    }

  return t;
}


weightDistribution wd;


//////// TRUE IF A POL IS PRIMITIVE /////////

int ifPrimitive(GF2X polinomio, GF2X modulo, ZZ n, int* lista)
{
  for (; *lista; lista++)
    {
      order_change = n / *lista;
      if (PowerMod(polinomio,order_change,modulo) == unity)
        break;
    }
  return !*lista;
}


//////// FACTORIZATION IN N /////////

void factorInt(ZZ n, int* lista)
{
```

```
  int i=0;

  if (!(n%2))
    {
      n /=2;
      lista[i++]=2;

      while (!(n%2))
        n /= 2;
    }
  for (int m = 3; m*m <= n; m += 2)
    {
      if (!(n%m))
        {
          n /=m;
          lista[i++]=m;

          while (!(n%m))
            n /= m;
        }
    }

  if (n!=1)
    lista[i++]=to_int(n);

  lista[i]=0;
}


//////////  FIND ORDER OF A POL IRR  ///////////

ZZ findOrder(GF2X pol,ZZ n,int* lista)
{
  ZZ order = n;
  for (int i=0; lista[i]; i++)
    {
      while(PowerXMod(order,pol) == unity)
        {
          if (!(order%lista[i]))
            {
              order /= lista[i];
            }
          else break;
        }
      if (!(PowerXMod(order,pol) == unity))
        order *= lista[i];
    }

  return order;
}
```

```
// ///////// FIND ORDER OF A GENERIC POL /////////

ZZ findOrder(vec_pair_GF2X_long factors)
{
  int lista_fatt[20];
  ZZ ordine_irr[factors.length()+1], order=to_ZZ(1);

  ordine_irr[factors.length()]=to_ZZ(0);

  for (int i=0; i<factors.length(); i++)
    {
      if (deg(factors[i].a)==1)
        ordine_irr[i]=to_ZZ(1);
      else
      {
        ordine_irr[i] = power2_ZZ(deg(factors[i].a))-1;
        for (int k=0 ; k<20 ; k++)
          lista_fatt[k]=0;

        factorInt(ordine_irr[i],lista_fatt);

        for (int j=0; lista_fatt[j]; j++)
          {
            while(PowerXMod(ordine_irr[i],factors[i].a) == unity)
              {
                if (!(ordine_irr[i]%lista_fatt[j]))
                  {
                    ordine_irr[i] /= lista_fatt[j];
                  }
                else break;
              }
            if (!(PowerXMod(ordine_irr[i],factors[i].a) == unity))
              ordine_irr[i] *= lista_fatt[j];
          }
      }
    }

  ZZ max=to_ZZ(1);
  int cont=0;

  for (int i=0; i<factors.length(); i++)
    {
      if (factors[i].b>max)
        {
          for (int k=cont+1; power2_ZZ(k) < factors[i].b; k++)
            cont=k;
          max=power2_ZZ(cont+1);
        }
    }
```

```
  order*=Lcm_ZZ(ordine_irr)*max;

  return order;
}


//////////  FIND ORDER MOD  ///////////

ZZ findOrderMod(GF2X pol,ZZ n,int* lista, GF2X mod)
{
  ZZ order = n;
  for (int i=0; lista[i]; i++)
    {
      while(PowerMod(pol,order,mod) == unity)
        {
          if (!(order%lista[i]))
            {
              order /= lista[i];
            }
          else break;
        }
      if (!(PowerMod(pol,order,mod) == unity))
        order *= lista[i];
    }

  return order;
}


//////  FIND A PRIMITIVE ELEMENT OF A FINITE FIELD  /////

GF2X findPrimitive(GF2X pol_gen,int n, ZZ m,int* lista)
{
  vec_GF2 pol_prim (INIT_SIZE,n+1);
  pol_prim[n]=0;

  GF2X pol_primitive;
  ZZ a = power_ZZ(2,n)-2;

  do
    {
      ZZ b = a;
      for (int c = n-1; c >=0 ; c--)
        {
          if (b < power_ZZ(2,c))
            pol_prim[c] = 1;
          else
            {
              pol_prim[c]=0;
              b -= power_ZZ(2,c);
            }
```

```
        }
      pol_primitive = to_GF2X(pol_prim);
      a--;
    }
  while (!ifPrimitive(pol_primitive,pol_gen,m,lista));

  return(pol_primitive);
}


//////////  RANDOM GF2X  //////////

GF2X randomGF2X(int n)
{
  vec_GF2 vec_pol(INIT_SIZE,n+1);
  srand ( time(NULL) );
  for (int i=0; i<n+1; i++)
    vec_pol[i] = rand()%2;

  GF2X pol = to_GF2X(vec_pol);
  return pol;
}



//////////  CALCOLO DISTRIBUZIONE CODICE DUALE  //////////


int computeDualDist(GF2X pol_gen, int length, ZZ* dual_weight_dist,
                    double dif_com)
{
  vec_pair_GF2X_long factors = factorization_GF2X(pol_gen);

  time_t start_com,end_com;

  ZZ n_provv;
  for (int i=0; i<factors.length(); i++)
    {
      n_provv += deg(factors[i].a)*factors[i].b;
    }
  n = to_int(n_provv);

  if (length > power_ZZ(2,n)-1)
    {
      cout << "Length to large, write a number less or equal to " <<
        power_ZZ(2,n)-1 << "." <<  endl;
      return EXIT_FAILURE;
    }

  cout << "The polynomial factorizaton is: " << pol_gen << " = "
       << factors << endl;
```

```
cout << "The polynomial order is: " << findOrder(factors) << endl;

ZZ weight_dist_prod[length+1];
for (int i=0; i < length+1; i++)
  weight_dist_prod[i]=to_ZZ(0);

ZZ dim= to_ZZ(0);
for (int c=0; c < factors.length(); c++)
  {

    int lista_pow_irr[20];
    factorInt(power_ZZ(2,deg(factors[c].a))-1,lista_pow_irr);
    ZZ order;
    order = findOrder(factors[c].a,power_ZZ(2,deg(factors[c].a))-1,
                      lista_pow_irr);

    ZZ prod2=to_ZZ(0);
    for (int t=2; t <= factors[c].b; t++)
      {
        ZZ prod = to_ZZ(1);
        int test4=1;
        for (int k=0; k < ((t - t%2)/2); k++)
          {
            for (int s=1; ; s++)
              {
                if (power_ZZ(2,s) >= (t/(1+2*k)))
                  {
                    if (test4)
                      {
                        prod *= power(power_ZZ(2,s),deg(factors[c].a)-1);
                        test4--;
                      }
                    else prod *= power(power_ZZ(2,s),deg(factors[c].a));
                    break;
                  }
              }
          }
        prod *= (power_ZZ(2,deg(factors[c].a))-1)/order;
        prod2 = prod;
      }
    prod2 += (power_ZZ(2,deg(factors[c].a))-1)/order+1;

    if (prod2 > dim)
      dim = prod2;
  }

GF2X matrice_gen[factors.length()][to_int(dim)+1];
ZZ ordine_gen[factors.length()][to_int(dim)+1];
GF2X pol_inv[factors.length()];
```

```
for (int i=0; i<factors.length(); i++)
  {
    GF2X pol_da_inv = (pol_gen/powerGF2X(factors[i].a,factors[i].b))
                          %(powerGF2X(factors[i].a,factors[i].b));

    pol_inv[i]=InvMod(pol_da_inv,powerGF2X(factors[i].a,factors[i].b));
  }

for (int c=0; c < factors.length(); c++)
  {

    cout << "---" << endl << "The polynomial " << factors[c].a << " is";

    int s=0;

    int lista_pow_irr[20];
    int degc = deg(factors[c].a);
    factorInt(power_ZZ(2,degc)-1,lista_pow_irr);
    ZZ order;
    order = findOrder(factors[c].a,power_ZZ(2,degc)-1,lista_pow_irr);

    GF2X pol_shift;


    if (ifPrimitive(identity,factors[c].a,power_ZZ(2,degc),lista_pow_irr))
      {
        /////// CASO POTENZA DI IRRIDUCIBILE PRIMITIVO
        cout << " primitive." << endl;

        matrice_gen[c][s] = powerGF2X(factors[c].a,factors[c].b-1);
        ordine_gen[c][s] = power_ZZ(2,degc)-1;
        s++;

      }
    else
      {

        ///////CASO POTENZA DI IRRIDUCIBILE NON PRIMITIVO

        cout << "n't primitive and has order: " << order <<"." << endl;

        pol_shift = findPrimitive(factors[c].a,degc,
                        power_ZZ(2,deg(factors[c].a))-1,lista_pow_irr);
        cout << "A primitive element is " << pol_shift << endl;
        GF2X pol_shift_2 = pol_shift;
        for (int i = 1; i * order <= power_ZZ(2,degc)-1; i++)
          {

            matrice_gen[c][s]= pol_shift_2
                              *powerGF2X(factors[c].a,factors[c].b-1);
```

```
            ordine_gen[c][s]= order;
            s++;

            pol_shift_2 = MulMod(pol_shift_2 ,pol_shift ,factors[c].a);
          }


      }

    for (int t=2; t <= factors[c].b ; t++)
      {
        int lista_fatt[20];
        factorInt(power_ZZ(2,degc*(t-1))*(power_ZZ(2,degc)-1),
                  lista_fatt);

        ////// CALCOLO DI 2^(LOG_2 t)

        ZZ potenza = to_ZZ(1);
        for (int m=1; ; m++)
          {
            if (power_ZZ(2,m) >= t)
              {
                potenza *= power_ZZ(2,m);
                break;
              }
          }

        ////////////// CALCOLO DEL GENERATORE a_p

        srand(time(NULL));
        GF2X a_p;
        ZZ order_a_p;
        if (degc==1 && factors[c].a!=identity)
          a_p = identity;
        else
          {
            for (;;)
              {
                a_p = random_GF2X(t*degc-1);
                if (!(IsOne(a_p)) && !(IsZero(a_p%factors[0].a)) &&
                    !(IsZero(a_p)))
                  {
                    order_a_p = findOrderMod(a_p,
                              power_ZZ(2,degc*(t-1))*(power_ZZ(2,degc)-1),
                              lista_fatt,powerGF2X(factors[c].a,t));
                    if (IsZero(order_a_p % (power_ZZ(2,degc)-1)))
                      {
                        order_a_p /= (power_ZZ(2,degc)-1);
                        if (!(IsOne(order_a_p)))
                          {
                            a_p = PowerMod(a_p,order_a_p,
```

```
                                        powerGF2X(factors[c].a,t));
                    if (!(IsOne(a_p)) && !(IsZero(a_p)))
                      break;
                  }
              }
          }
      }
  }

GF2X lista_gen[degc*((t - t%2) / 2)];
for (int k = 0; k < ((t - t%2) / 2); k++)
  {
    for (int j = 0; j < degc; j++)
      {
        lista_gen[k*degc+j] = unity+(powerGF2X(identity,j)*
                                     powerGF2X(factors[c].a,1+2*k));
      }
  }

/////// RICERCA DEL GENERATORE DA ELIMINARE

int cancel=0;

int comb[degc];
for (int i=0; i<degc; i++)
  comb[i] = 0;

for (;;)
  {
    for (int i=degc-1; i>=0; i--)
      {
        int sum = 0;
        for (int j=0; j<i; j++)
          sum += comb[j];

        if (sum == i*(2-1))
          {
            comb[i]++;
            comb[i]%=2;
          }
      }

    GF2X prod = unity;

    for (int i=0; i<degc; i++)
      prod *= PowerMod(lista_gen[i],comb[i],
                       powerGF2X(factors[c].a,2));
    prod %= powerGF2X(factors[c].a,2);

    if (prod == PowerXMod(order,powerGF2X(factors[c].a,2)))
      {
```

```
        while(!(comb[cancel]))
          cancel++;
        break;
      }

    int test = 1;

    for (int i=0 ; i<degc;i++)
      if (comb[i]!=1)
        test = 0;

    if (test)
      break;
  }

for (int i=cancel; i<degc*((t - t%2) / 2)-1; i++)
  lista_gen[i] = lista_gen[i+1];

for (int i=0; i<degc*((t - t%2) / 2)-1; i++)
  cout << lista_gen[i] << endl;

GF2X prod = unity;
for (int i_p=0; i_p <((power_ZZ(2,degc)-1)/order); i_p++)
  {
    int comb[degc*((t - t%2) / 2)-1], mod[degc*((t - t%2) / 2)];
    for (int i=0; i<degc*((t - t%2) / 2)-1; i++)
      {
        comb[i] = 0;
      }

    for (int i=0; i<degc*((t - t%2) / 2); i++)
      {
        int k = ((i-i%degc)/degc);
        for (int s=1; ; s++)
          {
            if (power_ZZ(2,s) >= (t/(1+2*k)))
              {
                mod[i] = to_int(power_ZZ(2,s));
                break;
              }
          }
      }

    for (int i=cancel; i<degc*((t - t%2) / 2)-1; i++)
      mod[i] = mod[i+1];

    int test2 =1;

    for (;;)
      {
        prod = powerGF2X(a_p,i_p);
```

```
                    if (test2)
                      test2--;
                    else
                      {
                        for (int i=degc*((t - t%2) / 2)-2; i>=0; i--)
                          {
                            int sum = 0,maxsum = 0;
                            for (int j=0; j<i; j++)
                              {
                                sum += comb[j];
                                maxsum += mod[j]-1;
                              }

                            if (sum == maxsum)
                              {
                                comb[i]++;
                                comb[i]%=mod[i];
                              }
                          }
                      }

                    for (int i=0; i<degc*((t - t%2) / 2)-1;i++)
                      prod *= powerGF2X(lista_gen[i],comb[i]);
                    prod %= powerGF2X(factors[c].a,t);

                    matrice_gen[c][s] = prod*powerGF2X(factors[c].a,
                                                       factors[c].b-t);
                    ordine_gen[c][s] = order*potenza;
                    s++;

                    int test = 1;

                    for (int i=0 ; i<degc*((t - t%2) / 2)-1;i++)
                      if (comb[i]!=(mod[i]-1))
                        test = 0;

                    if (test)

                      break;
                  }
              }
          }
        ordine_gen[c][s]=1;
      }

    cout << endl;
    for (int i=0; i<factors.length(); i++)
      {
        for (int j=0; j<to_int(dim); j++)
          cout << matrice_gen[i][j] << endl;
```

```
    cout << endl;
  }

int quanti_gen[factors.length()], comb[factors.length()];
for (int c=0; c < factors.length(); c++)
  {
    int m = 1;
    while (!(IsZero(ordine_gen[c][m])))
      m++;
    quanti_gen[c]=m;
    comb[c]=0;
  }

int test2 =1;
for (;;)
  {
    if (test2)
      test2--;
    else
      {
        for (int s=factors.length()-1; s>=0; s--)
          {
            int sum = 0,maxsum = 0;
            for (int r=0; r<s; r++)
              {
                sum += comb[r];
                maxsum += quanti_gen[r]-1;
              }

            if (sum == maxsum)
              {
                comb[s]++;
                comb[s]%=quanti_gen[s];
              }
          }
      }


    for (int f=0; f<factors.length(); f++)
      {
        cout << matrice_gen[f][comb[f]] << " - ";
      }
    cout << endl;


    ZZ d_i[factors.length()+1];
    for (int i=0; i < factors.length(); i++)
      d_i[i] = ordine_gen[i][comb[i]];
    d_i[factors.length()] = to_ZZ(0);

    int K_i[factors.length()];
```

```
K_i[0]=2;

for (int i=1; i < factors.length(); i++)
  {
    ZZ lista_lcm[factors.length()];
    for (int j=0; j < i; j++)
      lista_lcm[j]=d_i[j];

    ZZ lcm = Lcm_ZZ(lista_lcm);
    K_i[i]=to_int(GCD(d_i[i],lcm));
  }

int combin[factors.length()];
combin[0]=1;
for (int i=1; i<factors.length(); i++)
  combin[i] = 0;

int test5=1;
for (;;)
  {
    if (test5)
      test5--;
    else
      {
        for (int i=factors.length()-1; i>0; i--)
          {
            int sum = 0,maxsum = 0;
            for (int j=1; j<i; j++)
              {
                sum += combin[j];
                maxsum += K_i[j]-1;
              }

            if (sum == maxsum)
              {
                combin[i]++;
                combin[i]%=K_i[i];
              }
          }
      }

    GF2X generatore = to_GF2X(0);
    for (int i=0; i<factors.length(); i++)
      {
        GF2X prodotto = to_GF2X(1);
        for (int g=0; g<factors.length(); g++)
          {
            if (g!=i)
              prodotto *= powerGF2X(factors[g].a,factors[g].b);
          }
        generatore += ((matrice_gen[i][comb[i]]*
```

```
                                powerGF2X(identity,combin[i]))%
                               powerGF2X(factors[i].a,factors[i].b))*
                               pol_inv[i]*prodotto;
            }
          generatore%=pol_gen;
          cout << generatore << endl;

          time(&start_com);
          wd.compute_distribution(pol_gen, generatore, n, length);
          time(&end_com);
          dif_com += difftime (end_com,start_com);

          for (int j = 0; j < length+1; j++)
            {
              weight_dist_prod[j] += wd.weight_dist[j];
              cout << weight_dist_prod[j] << " ";
            }
          cout << endl << endl;

          int test = 1;

          for (int i=0 ; i<factors.length();i++)
            if (combin[i]!=(K_i[i]-1))
              test = 0;

          if (test)
            break;
        }

      int test = 1;

      for (int s=0 ; s<factors.length(); s++)
        if (comb[s]!=(quanti_gen[s]-1))
          test = 0;

      if (test)
        break;
    }

  cout << endl;

  ZZ card_cod= to_ZZ(0);
  for (int i = 0; i <= length; i++)
    {
      card_cod += weight_dist_prod[i];
    }
  cout << "Verifica:  " << power_ZZ(2,n) << " = " << card_cod << endl;


  for (int i = 0; i < length+1; i++)
    dual_weight_dist[i] = weight_dist_prod[i];
```

```
  return 0;
}



//////////  MAIN  //////////

int main()
{
  int deg,length;

  //Generazione del polinomio unita'
  unity = to_GF2X(1);

  //generazione del polinomio x
  vec_GF2 ics(INIT_SIZE, 2);
  ics[0] = 0;
  ics[1] = 1;
  identity = to_GF2X(ics);

  bool test;

  cout << endl << "You have 2 different way to write the generator
    polynomial: " << endl << "  type  0  to write the polynomial in
    the extended formula (e.g. x^3+x+1)" << endl << "  type  1  to
    write the polynomial in the NTL formula (e.g. [1 1 0 1] := 1+x+x^3)."
     << endl << "Type now: ";
  cin >> test;

  if (test)
    {
      cout << endl << "Write the generator polynomial: ";
      cin >> pol_gen;
    }
  else
    {
      int m;
      cout << "Write the degree of the generator polynomial: ";
      cin >> m;
      cout << endl;

      char pol[12*m];
      cout << "Write the generator polynomial: ";
      while (!getchar());
      cin.getline (pol , 12*m);

      inputConversion(m,pol);

      cout << pol_gen << endl;
    }
```

```
  cout << endl << "Write the lenght of the code: ";
  cin >> length;


  RR epsilon;
  cout << endl << "Write the parameter of error probability of the
       symmetric binary channel: ";
  cin >> epsilon;
  cout << endl;

  time_t start,end;
  time_t start_mac,end_mac;
  double dif,dif_mac;
  time (&start);

  if (IsZero(pol_gen%identity))
    {
      cout << "You have to write a polynomial not divisible by x!" << endl;
      return EXIT_FAILURE;
    }


  ZZ dual_weight_dist[length+1];

  computeDualDist(pol_gen,length,dual_weight_dist,dif_com);

  cout << endl << "DISTRIBUTION OF THE DUAL CODE: " << endl;
  for (int i=0; i<length+1; i++)
    cout << dual_weight_dist[i] << " ";
  cout << endl;

  time(&start_mac);
  wd.McWilliams(n,length,dual_weight_dist);
  time(&end_mac);

  cout << endl;
  int dmin_codice;
  test=true;
  cout << endl << "DISTRIBUTION OF THE CODE: " << endl;
  for (int i=0; i<length+1; i++)
    {
      cout << wd.weight_dist_ZZ[i] << " ";
      if (i > 0 && test)
        {
          if (wd.weight_dist_ZZ[i] != 0)
            {
              dmin_codice = i;
              test=false;
            }
```

```
        }
      }

    ZZ card_cod = to_ZZ(0);
    for (int i = 0; i <= length; i++)
      card_cod += wd.weight_dist_ZZ[i];
    cout << endl << "Verifica:  " << power_ZZ(2,length-n)
         << " = " << card_cod << endl;

    RR pue_du=to_RR(1);

    for (int i=1; i<length+1; i++)
      {
        pue_du += to_RR(dual_weight_dist[i])*power(1-2*epsilon,i);
      }
    pue_du /= power(to_RR(2),n);
    pue_du -= power(1-epsilon,length);

    cout << endl << "Pue: " << pue_du << endl;

    int proper=wd.testProper(length,dmin_codice,pol_gen,dual_weight_dist);

    if (proper == 1)
      cout << endl << "The code is PROPER." << endl;
    else if (proper == 2)
      cout << endl << "The code is PROBABLY PROPER." << endl;
    else
      cout << endl << "The code is NOT PROPER." << endl;

    time (&end);
    dif = difftime (end,start);
    dif_mac = difftime (end_mac,start_mac);
    cout << "Time spent: " << dif << endl;
    cout << "Time spent MacWilliams: " << dif_mac << endl;
    cout << "Time spent compute_distribution: " << dif_com << endl;

    return(EXIT_SUCCESS);
}
```

Bibliografia

[CBH] Castagnoli, G.; Bräuer, S.; Hermann M.: *Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits*, IEEE Trans. on Communication, Vol 41, no. 6, Giugno 1993, pp. 883-892.

[CW] Chang, S. C.; Wolf, J. K.: *A Simple Derivation of the MacWilliams'Identity for Linear Codes*, IEEE Trans. on Inform. Theory, Vol IT-26, no. 4, Luglio 1980, pp. 476-477.

[Ch] Childs, L.: *Algebra, un'introduzione concreta*, ETS Editrice, 1989.

[Do] Dodunekova, R.: *The Duals of MMD Codes Are Proper for Error Detection*, IEEE Trans. on Inform. Theory, Vol 49, no. 8, Agosto 2003, pp. 2034-2037.

[FKKL] Fujiwara, T.; Kasami, T.; Kitai, A.; Lin, S.: *On the Undetected Error Probability for Shortened Hamming Codes*, IEEE Trans. on Communication, Vol COM-33, pp. 570-574, Giugno 1985.

[He] Herstein, I. N.: *Algebra*, editori Riuniti, Giugno 1999.

[HLL] Hoffman, D.G., Leonard, D.A.; Lindner,C.C., Phelps,K.T., Rodger, C.A., Wall, J.R.; *Coding Theory: The Essential*, Marcel Dekker, Inc., 1991.

[La] Lang, S.: *Algebra*, Addison-Wesley Publishing Company, Inc., 1965.

[LN] Lidl, R.; Niederreiter, H.: *Finite Fields*, Encyclopedia of Mathematics and its Application, Gian-Carlo Rota, 1993.

[MS] MacWilliams, F. J.; Sloane, N. J. A.: *The Theory of Error-Correcting Codes*, North Holland, 1988.

[VL] Van Lint, J.H.; *Introduction to Coding Theory, Third Edition*, Springer-Verlag, 1999.